**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра ИС**

**ОТЧЕТ**

**по заданию №2-1**

**по дисциплине «Методы и средства проектирования информационных систем»**

**Тема: Задача 2-1. Приемы моделирования Спецификации изделий**

|  |  |  |
| --- | --- | --- |
| Студены гр. 1374 |  | Архипов Д.А.  Дюков Н.В.  Хлебников А. |
| Преподаватель |  | Дубенецкий В.А. |

Санкт-Петербург

2024 г.

**Содержание**

[**1. Задание на выполнение работы 3**](#_3znysh7)

[1.1 Функциональные требования 3](#_2et92p0)

[1.2 Содержание работы 3](#_17dp8vu)

[1.3 Исходные данные для задачи 4](#_3rdcrjn)

[1.4 Используемые инструменты 4](#_26in1rg)

[1.5 Используемые документы 4](#_lnxbz9)

[**2. Описание выполненной работы 5**](#_35nkun2)

[2.1 Анализ исходных данных 5](#_1ksv4uv)

[2.2 Разработка функциональных требований к подсистеме 6](#_44sinio)

[2.3 Разработка проектной модели классов 7](#_2jxsxqh)

[2.4 Разработка модели хранения в среде СУБД 7](#_z337ya)

[2.5 Создание базы данных 8](#_3j2qqm3)

[2.6 Разработка основных процедур 11](#_1y810tw)

[2.7 Тестирование разработки 20](#_4i7ojhp)

[**Вывод 24**](#_1ci93xb)

**Задача №1**

**Приемы моделирования классификатора изделий**

# Задание на выполнение работы

Разработать проект каркаса для работы со спецификациями изделий

## 1.1 Функциональные требования

Разработать проект каркаса для работы со спецификациями изделий

## Обеспечить:

## Ведение справочника изделий различных типов, используемых в составе изделий;

## Формирование строк спецификаций с указанием норм расхода;

## Поиск всех строк спецификации изделия на всю глубину вложенности;

## Расчет сводных норм расхода компонентов изделия по ресурсам заданного класса.

## 1.2 Содержание работы

|  |  |  |
| --- | --- | --- |
| **№** | **Этап выполнения** | **Результат** |
|  | Разработать функциональные требования к проектируемой подсистеме | Диаграммы вариантов использования. Описание компонентов диаграмм |
|  | Разработать модель классов для выделенных процессов | Диаграммы классов с операциями и атрибутами. Описание компонентов диаграмм. |
|  | Разработать модель хранения в среде СУБД для решения задач управления материальной спецификацией | Диаграммы ER. Скрипты метаданных с комментариями. |
|  | Разработка основных SQL-процедур для работы с классификатором | Скрипты SQL-процедур с комментариями. |
|  | Тестирование процедур | Описание тестов. Скрипты исходных данных для тестов.  Скриншоты результатов тестирования. |

## 1.3 Используемые инструменты

РедакторVisio. МодельDatabase.Entity relationship

IDE Datagrip

СУБД - Microsoft SQL Server

Текстовый редактор Word

## 1.4 Используемые документы

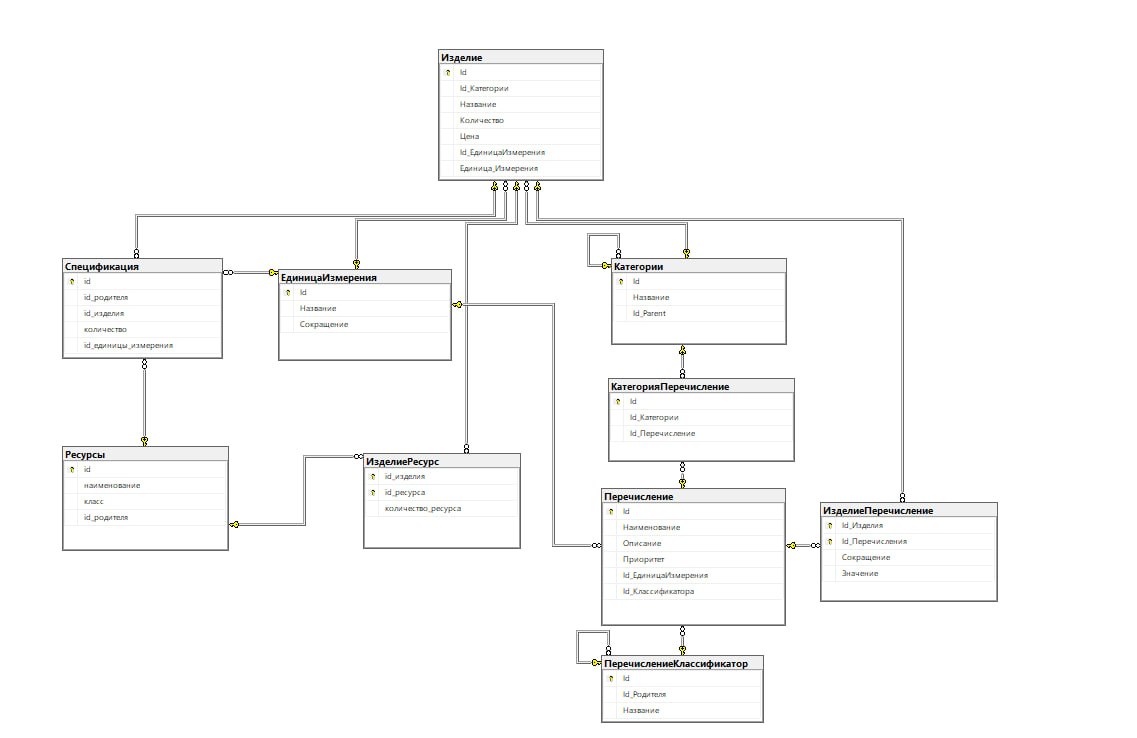
1. Data Definition Guide : файл DataDef.pdf
2. Language Reference Guide: файл LangRef.pdf
3. Архитектура информационных систем. Учебник. М.,Издательский центр «Академия», 2012
4. Информационное обеспечение жизненного цикла изделий. Учебное пособие: СПб, Издательство Политехнического университета, 2012
5. Приемы объектно-ориентированного проектирования. Паттерны проектирования: Изд-во «ПИТЕР», 2008
6. Проектирование корпоративных информационных систем.: СПб, Изд-во СПбГЭТУ «ЛЭТИ», 2013

# Описание выполненной работы

## 2.1 Анализ исходных данных

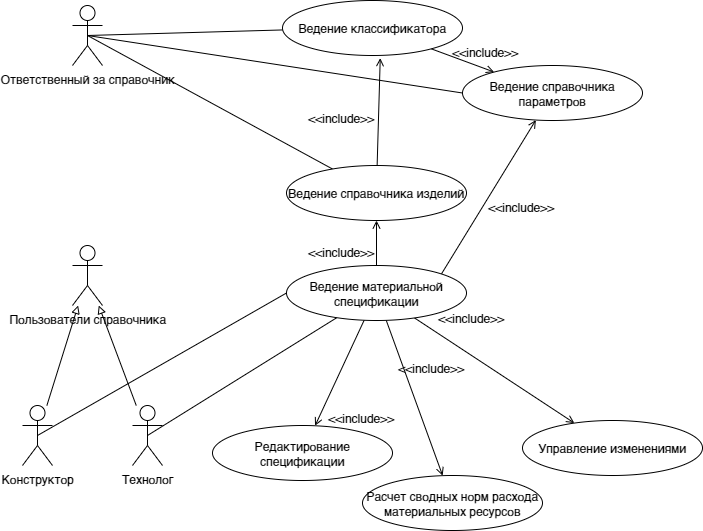
Для разработки концептуального проекта будем использовать редактор VISIO.

Модель классов этапа анализа представлена на рисунке 1.



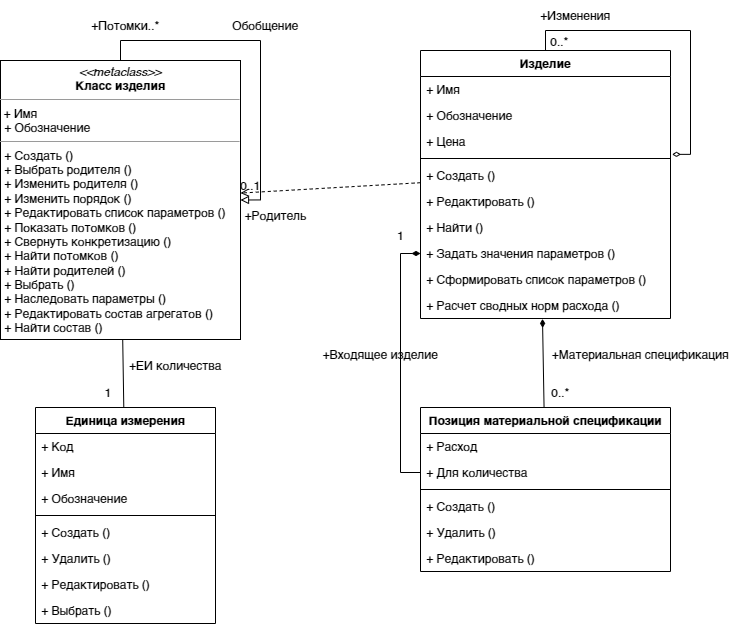
*Рисунок 1 – Модель классов для ведения спецификации материальных ресурсов*

## 2.2 Разработка функциональных требований к подсистеме

****

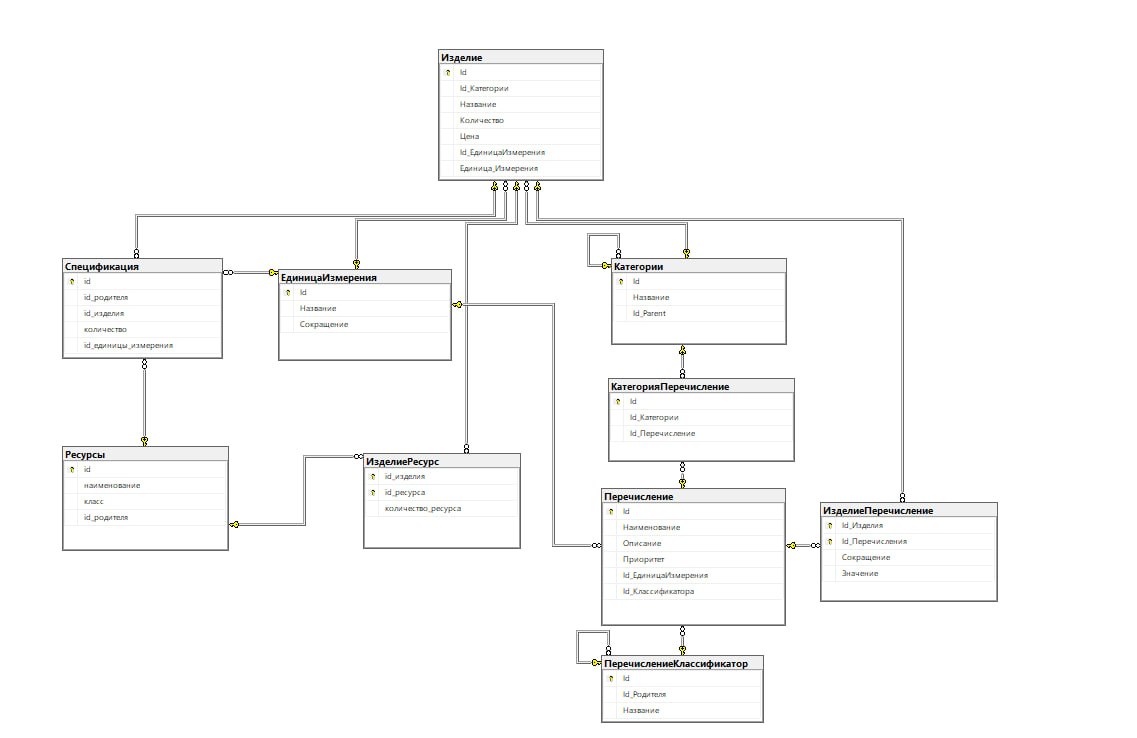
*Рисунок 2 – Диаграмма прецедентов Материальная спецификация изделий*

## Разработка проектной модели классов

****

*Рисунок 3 - Проектная модель классов для прецедента Управление спецификацией изделий (Billofmaterials)*

## Разработка модели хранения в среде СУБД



*Рисунок 4 - ERD для проекта Материальная спецификация изделий*

## Создание базы данных

1. **Создание таблицы *Единицы измерения***

CREATE TABLE ЕдиницаИзмерения (

Id INT IDENTITY(1,1) NOT NULL PRIMARY KEY,

Название NVARCHAR(50) NOT NULL,

Сокращение NVARCHAR(10) NOT NULL

);

1. **Создание таблицы *Категории***

CREATE TABLE Категории (

Id INT IDENTITY(1,1) NOT NULL PRIMARY KEY,

Название NVARCHAR(100) NOT NULL,

Id\_Parent INT NULL REFERENCES Категории (Id)

);

1. **Создание таблицы *Изделие***

CREATE TABLE Изделие (

Id INT IDENTITY(1,1) NOT NULL PRIMARY KEY,

Id\_Категории INT NOT NULL REFERENCES Категории (Id),

Название NVARCHAR(100) NOT NULL,

Количество INT NOT NULL,

Цена DECIMAL(18,2) NOT NULL,

Id\_ЕдиницаИзмерения INT NOT NULL REFERENCES ЕдиницаИзмерения (Id)

);

1. **Создание таблицы *Перечисление Классификатор***

CREATE TABLE ПеречислениеКлассификатор (

Id INT IDENTITY(1,1) NOT NULL PRIMARY KEY,

Id\_Родителя INT NULL REFERENCES ПеречислениеКлассификатор (Id),

Название NVARCHAR(100) NOT NULL

);

1. **Создание таблицы *Перечисление***

CREATE TABLE Перечисление (

Id INT IDENTITY(1,1) NOT NULL PRIMARY KEY,

Наименование NVARCHAR(100) NOT NULL,

Описание NVARCHAR(255) NULL,

Приоритет INT NOT NULL,

Id\_ЕдиницаИзмерения INT NULL REFERENCES ЕдиницаИзмерения (Id),

Id\_Классификатора INT NOT NULL REFERENCES ПеречислениеКлассификатор (Id)

);

1. **Создание таблицы Категория *Перечисление***

CREATE TABLE КатегорияПеречисление (

Id INT IDENTITY(1,1) NOT NULL PRIMARY KEY,

Id\_Категории INT NOT NULL REFERENCES Категории (Id),

Id\_Перечисление INT NOT NULL REFERENCES Перечисление (Id)

);

1. **Создание таблицы *Изделие Перечисление***

CREATE TABLE ИзделиеПеречисление (

Id\_Изделия INT NOT NULL REFERENCES Изделие (Id),

Id\_Перечисления INT NOT NULL REFERENCES Перечисление (Id),

PRIMARY KEY (Id\_Изделия, Id\_Перечисления)

);

1. **Создание таблицы *Ресурсы***

CREATE TABLE Ресурсы (

Id INT IDENTITY(1,1) NOT NULL PRIMARY KEY,

Наименование NVARCHAR(255) NOT NULL,

Класс NVARCHAR(255) NULL,

Id\_Родителя INT NULL REFERENCES Ресурсы (Id)

);

1. **Создание таблицы *Изделие ресурс***

CREATE TABLE ИзделиеРесурс (

Id\_Изделия INT NOT NULL REFERENCES Изделие (Id),

Id\_Ресурса INT NOT NULL REFERENCES Ресурсы (Id),

Количество\_Ресурса INT NOT NULL,

PRIMARY KEY (Id\_Изделия, Id\_Ресурса)

);

1. **Создание таблицы *Спецификация***

CREATE TABLE Спецификация (

Id INT IDENTITY(1,1) NOT NULL PRIMARY KEY,

Id\_Родителя INT NULL REFERENCES Ресурсы (Id),

Id\_Изделия INT NULL REFERENCES Изделие (Id),

Количество INT NULL,

Id\_Единицы\_Измерения INT NULL REFERENCES ЕдиницаИзмерения (Id)

);

## Разработка основных процедур

1. **Добавление дупликата продукта**

ALTER PROCEDURE [dbo].[DuplicateProduct]

*Вход:*

*- @id\_изделия INT: ID изделия, которое нужно дублировать*

*Выход:*

*- НовыйIdИзделия INT: Новый ID дублированного изделия (по желанию)*

*Эффекты по функции:*

*- Создает новую запись в таблице "Изделие" с данными исходного изделия и измененным названием (добавляет суффикс '-копия')*

*- Создает записи в таблице "ИзделиеРесурс" для нового изделия, дублируя ресурсы исходного изделия*

AS

BEGIN

SET NOCOUNT ON;

DECLARE @NewProductId INT,

@NewName NVARCHAR(100),

@Id\_Категории INT,

@Количество INT,

@Цена DECIMAL(18, 2),

@Id\_ЕдиницаИзмерения INT,

@Единица\_Измерения NVARCHAR(10);

-- *Получаем информацию о изделии, которое нужно дублировать*

SELECT

@NewName = Название,

@Id\_Категории = Id\_Категории,

@Количество = Количество,

@Цена = Цена,

@Id\_ЕдиницаИзмерения = Id\_ЕдиницаИзмерения,

@Единица\_Измерения = Единица\_Измерения

FROM Изделие

WHERE Id = @id\_изделия;

-- *Формируем новое название изделия*

SET @NewName = @NewName + '-копия';

-- *Вставляем новое изделие в таблицу "Изделие" с измененным названием*

INSERT INTO Изделие (Id\_Категории, Название, Количество, Цена, Id\_ЕдиницаИзмерения, Единица\_Измерения)

VALUES (@Id\_Категории, @NewName, @Количество, @Цена, @Id\_ЕдиницаИзмерения, @Единица\_Измерения);

-- *Получаем новый ID для вставленного изделия*

SET @NewProductId = SCOPE\_IDENTITY();

-- *Дублируем все записи из таблицы "ИзделиеРесурс" для нового изделия*

INSERT INTO ИзделиеРесурс (id\_изделия, id\_ресурса, количество\_ресурса)

SELECT @NewProductId, id\_ресурса, количество\_ресурса

FROM ИзделиеРесурс

WHERE id\_изделия = @id\_изделия;

-- *Возвращаем новый ID дублированного изделия (по желанию)*

SELECT @NewProductId AS НовыйIdИзделия;

END;

1. **Изменение ресурса у продукта**

ALTER PROCEDURE [dbo].[UpdateResourceForProduct]

*Вход:*

*- @id\_изделия INT: ID изделия, для которого обновляем ресурс*

*- @old\_id\_ресурса INT: Старый ID ресурса, который нужно заменить*

*- @new\_id\_ресурса INT: Новый ID ресурса*

*Выход:*

*- Нет*

*Эффекты по функции:*

*- Проверяет наличие нового ресурса с указанным ID; если не существует, генерирует ошибку*

*- Проверяет наличие старого ресурса для указанного изделия; если не найден, генерирует ошибку*

*- Обновляет ID ресурса в таблице "ИзделиеРесурс" для указанного изделия*

*- Выводит сообщение об успешном обновлении ресурса*

AS

BEGIN

SET NOCOUNT ON;

-- *Проверка на существование нового ресурса*

IF NOT EXISTS (SELECT 1 FROM Ресурсы WHERE Id = @new\_id\_ресурса)

BEGIN

RAISERROR('Новый ресурс с таким ID не существует.', 16, 1);

RETURN;

END

-- *Проверка на существование старого ресурса для данного изделия*

IF NOT EXISTS (SELECT 1 FROM ИзделиеРесурс WHERE id\_изделия = @id\_изделия AND id\_ресурса = @old\_id\_ресурса)

BEGIN

RAISERROR('Указанный ресурс для данного изделия не найден.', 16, 1);

RETURN;

END

-- *Обновление ID ресурса для указанного изделия*

UPDATE ИзделиеРесурс

SET id\_ресурса = @new\_id\_ресурса

WHERE id\_изделия = @id\_изделия AND id\_ресурса = @old\_id\_ресурса;

-- *Возврат сообщения об успешном завершении*

PRINT 'Ресурс успешно обновлен.';

END;

1. **Получить ресурсы продукта**

CREATE PROCEDURE [dbo].[GetProductResources]

*Вход:*

*- @Product INT: ID изделия, для которого необходимо получить ресурсы*

*Выход:*

*- Таблица с информацией о ресурсах изделия, включающая:*

*- ID\_Ресурса: Идентификатор ресурса*

*- Ресурс: Наименование ресурса*

*- Класс\_Ресурса: Класс ресурса*

*- Количество\_Ресурса: Количество ресурса*

*- Перечисление: Название единицы измерения (в данном случае по умолчанию отображается как "штуки")*

*Эффекты по функции:*

*- Извлекает и возвращает информацию о ресурсах, связанных с указанным изделием, из таблиц "ИзделиеРесурс", "Ресурсы" и "ЕдиницаИзмерения"*

AS

BEGIN

SET NOCOUNT ON;

-- *Получаем состав изделия (велосипеда)*

SELECT

r.id AS ID\_Ресурса,

r.наименование AS Ресурс,

r.класс AS Класс\_Ресурса,

ir.количество\_ресурса AS Количество\_Ресурса,

ei.Название AS Перечисление

FROM

ИзделиеРесурс ir

INNER JOIN

Ресурсы r ON ir.id\_ресурса = r.id

INNER JOIN

ЕдиницаИзмерения ei ON ei.id = 1 -- По умолчанию "штуки", при необходимости можно доработать

WHERE

ir.id\_изделия = @Product;

END;

1. **Добавление ресурса**

CREATE PROCEDURE [dbo].[AddResource]

*Вход:*

*- @Наименование NVARCHAR(255): Наименование ресурса*

*- @Класс NVARCHAR(255): Класс ресурса (значение по умолчанию – NULL)*

*- @IdРодителя INT: Идентификатор родительского ресурса (значение по умолчанию – NULL)*

*Выход:*

*- Нет*

*Эффекты по функции:*

*- Вставляет новую запись в таблицу "Ресурсы" с заданными параметрами*

*- В случае успешной вставки выводит сообщение 'Запись успешно добавлена.'*

*- Обрабатывает возможные ошибки при вставке и выводит сообщение об ошибке с соответствующей серьезностью и состоянием*

AS

BEGIN

BEGIN TRY

-- *Вставка новой записи в таблицу "Ресурсы"*

INSERT INTO Ресурсы (наименование, класс, id\_родителя)

VALUES (@Наименование, @Класс, @IdРодителя);

PRINT 'Запись успешно добавлена.';

END TRY

BEGIN CATCH

-- *Обработка ошибок*

DECLARE @ErrorMessage NVARCHAR(4000) = ERROR\_MESSAGE();

DECLARE @ErrorSeverity INT = ERROR\_SEVERITY();

DECLARE @ErrorState INT = ERROR\_STATE();

RAISERROR(@ErrorMessage, @ErrorSeverity, @ErrorState);

END CATCH

END;

1. **Добавить ресурс продукту**

CREATE PROCEDURE [dbo].[AddResourceToProduct]

*Вход:*

*- @ProductId INT: ID изделия*

*- @ResourceId INT: ID ресурса*

*- @ResourceQuantity INT: Количество ресурса*

*Выход:*

*- Нет*

*Эффекты по функции:*

*- Вставляет новую запись в таблицу "ИзделиеРесурс", связывая изделие с ресурсом в указанном количестве*

*- В случае успешной вставки выводит сообщение 'Ресурс успешно добавлен к изделию.'*

*- В случае ошибки при добавлении ресурса выводит сообщение об ошибке с подробной информацией*

AS

BEGIN

BEGIN TRY

-- *Добавляем запись в таблицу ИзделиеРесурс*

INSERT INTO ИзделиеРесурс (id\_изделия, id\_ресурса, количество\_ресурса)

VALUES (@ProductId, @ResourceId, @ResourceQuantity);

PRINT 'Ресурс успешно добавлен к изделию.';

END TRY

BEGIN CATCH

-- *Обрабатываем ошибки*

DECLARE @ErrorMessage NVARCHAR(4000) = ERROR\_MESSAGE();

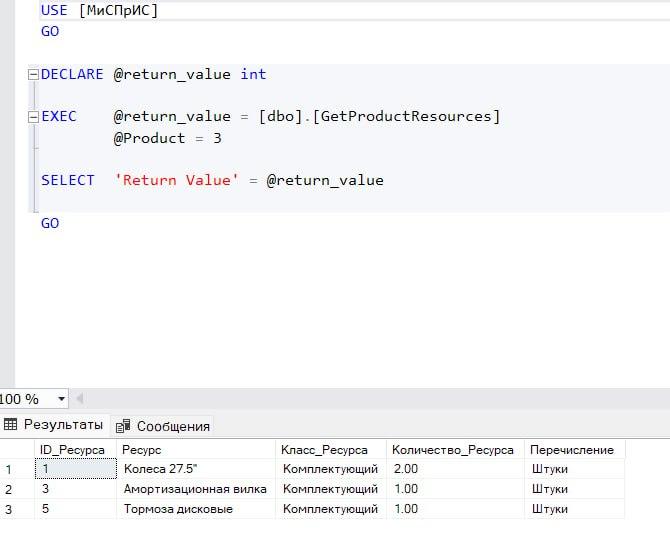
PRINT 'Ошибка при добавлении ресурса: ' + @ErrorMessage;

END CATCH

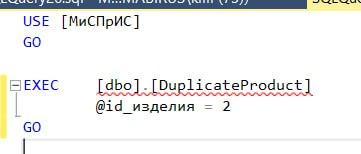
END;

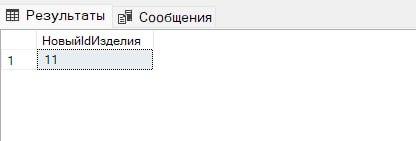
## Тестирование разработки

1. **Получение ресурсов у продукта**

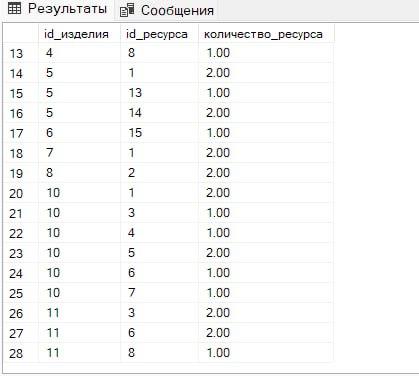


1. **Дублировать продукт**

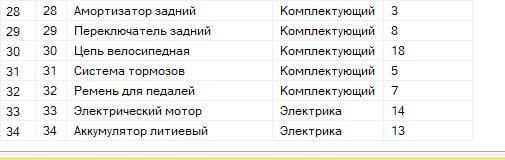


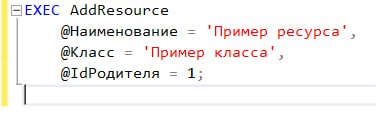


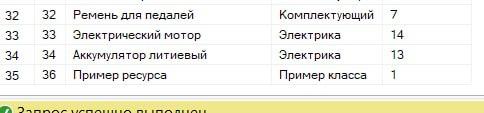
1. **Изменение ресурса у изделия**



1. **Добавить ресурс**

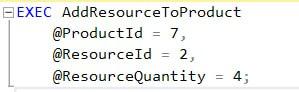


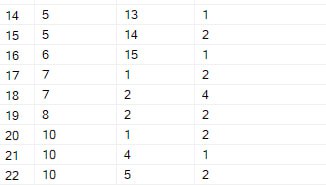




1. **Добавить ресурс в продукт**







# Вывод

В данной работе с помощью классификатора была создана и реализована модель предметной области «Велосипеды». Реализация произведена в базе данных Microsoft SQL Server. Проведено тестирование, показывающее корректную работу реализованных функций для работы с данными.