МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРОЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

"ПОВОЛЖСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ"

Кафедра ИБ

**Пояснительная записка**

**к курсовой работе по дисциплине**

**«БЕЗОПАСНОСТЬ СИСТЕМ БАЗ ДАННЫХ»**

**«Кондитерская фирма»**

**Выполнили:** студенты гр. Бис-31

Фомин В.И.

Шаунина В.О.

Соловьев И.С.

Курицына Т.В.

Черепанов П.В.

Казаков Д.С.

**Проверил**: Сучков Д.С.

г. Йошкар-Ола

2015 г.

Содержание

[Теоретическая часть 3](#_Toc405497992)

[Описание основных операций MSSQL. 3](#_Toc405497993)

[1. Создание базы данных 3](#_Toc405497994)

[2. Создание таблицы 3](#_Toc405497995)

[3. Ограничение целостности в MSSQL 3](#_Toc405497996)

[4. Описание типов данных 4](#_Toc405497997)

[Описание команд MSSQL 4](#_Toc405497998)

[1. Добавление данных командой INSERT 4](#_Toc405497999)

[2. Модификация записей командой UPDATE 5](#_Toc405498000)

[3. Удаление записей командой DELETE 5](#_Toc405498001)

[4. Выборка данных командой SELECT 5](#_Toc405498002)

[Описание триггеров и хранимые процедуры. 5](#_Toc405498003)

[Режимы аутентификации 6](#_Toc405498004)

[Практическая часть 7](#_Toc405498005)

[Создание базы данных 7](#_Toc405498006)

[Доказательство соответствия третьей нормальной форме 7](#_Toc405498007)

[API 7](#_Toc405498008)

[1. Аутентификация пользователя (создание сессии) 7](#_Toc405498009)

[2. Добавление данных в таблицы 8](#_Toc405498010)

[3. Удаление данных в таблицах 9](#_Toc405498011)

[4. Изменение данных в таблицах 9](#_Toc405498012)

[5. Выборка данных из ключевых таблиц по запросам 10](#_Toc405498013)

[6. Выборка данных из таблиц с объединением результатов 10](#_Toc405498014)

[Триггеры 10](#_Toc405498015)

[Просмотр результатов запросов 13](#_Toc405498016)

# Теоретическая часть

## Описание основных операций MSSQL.

### Создание базы данных

Первый тест, с помощью которого будет видно, что можно получить доступ к серверу баз данных - это попытка создать базу данных. Запущенный сервер MSSQL может управлять множеством баз данных. Обычно, для каждого проекта или каждого пользователя используется отдельная база данных.

Новая база данных создается при помощи операции CREATE DATABASE.

### Создание таблицы

Таблицы являются основными блоками хранения данных в базе. Перед любыми операциями создания, выборки или модификации записей необходимо сначала создать таблицу, в которой эти записи будут храниться.

В языке SQL таблицы создаются командой CREATE TABLE. Минимальный набор параметров включает имя таблицы и описания полей, состоящие из имени поля и типа данных. Команде CREATE TABLE также могут передаваться некоторые необязательные параметры: ограничения полей (правила, которые определяют, какие данные могут или не могут присутствовать в поле) и ограничения таблицы (общие ограничения и связи, определенные для самой таблицы).

### Ограничение целостности в MSSQL

Ограничения целостности, вводят некоторые ограничения для данных пользователя, тем самым помогая предотвратить ввод неправильных данных в базу. Когда мы назначаем для какой-либо колонки в таблице определённый тип данных, мы вводим ограничение на эту колонку.

Ограничение NOT NULL  просто говорит, что колонка не должна содержать значение null.  Ограничение NOT NULL не допускает появление значений NULL в колонке. Попытка вставить значение NULL, или выполнение оператора INSERT приведёт к ошибке выполнения оператора INSERT.

Ограничения уникальности дают уверенность в том, что данные, содержащиеся в колонке или группе колонок являются уникальными по отношению к другим строкам в той же таблице. Ограничение UNIQUE предотвращает появление в колонке одинаковых значений. Это достигается созданием уникального индекса для этой колонки.

Ограничение PRIMARY KEY, является комбинацией ограничений UNIQUE и NOT NULL. Накладывание этого ограничения на колонку означает, что её значения будут являться уникальными идентификаторами для каждой строки таблицы.

### Описание типов данных

SQL относится к категории языков с сильной типизацией. Это означает, что с любым объектом данных, представленным в MSSQL, связывается определенный тип, даже если на первый взгляд это и не очевидно. Тип данных одновременно определяет и ограничивает разновидности операций, которые могут выполняться с этими данными.

Типы не только ассоциируются со всеми данными, но и играют важную роль при создании таблиц. Как упоминалось в разделе «Знакомство с реляционными базами данных», таблицы состоят из одного или нескольких полей. При создании таблицы каждому полю, помимо имени, назначается определенный тип данных.

MSSQL имеет богатый список доступных пользователю встроенных типов. Кроме того, с помощью команды CREATE TYPE пользователи могут добавлять новые типы данных.

Каждый тип данных имеет внешнее представление, определяемое с помощью функций ввода и вывода. Многие из встроенных типов имеют понятные внешние форматы. Однако, некоторые типы являются либо уникальными в MSSQL, такие как геометрические, либо имеют несколько возможных форматов, такие как типы даты и времени. Некоторые функции ввода и вывода являются неинвертирумыми. Например, к ним относятся такие функции, результат вывода которых может привести к потере точности по сравнению с введённым первоначальным значением.

## Описание команд MSSQL

### Добавление данных командой INSERT

После создания таблицы с заданной структурой наступает следующий этап – заполнение таблицы данными. В MSSQL имеются три общих способа заполнения таблиц данными:

* вставка новых группированных данных командой INSERT INTO;
* вставка существующих данных из другой таблицы командой INSERT INTO в сочетании с командой SELECT;
* вставка данных из внешнего файла командой COPY (или \copy).

Тип каждого значения в секции VALUES должен соответствовать типу поля, которому оно присваивается. Если необязательный список полей отсутствует, MSSQL предполагает, что секция VALUES содержит значения всех полей в структуре таблицы в порядке их определения. Если количество значений меньше количества полей, MSSQL пытается использовать значение по умолчанию (или NULL при его отсутствии) для каждого пропущенного элемента.

### Модификация записей командой UPDATE

После того как записи сохранены в базе данных, вы можете обновить их поля командой SQL UPDATE. Новые значения полей задаются в виде констант, идентификаторов других баз данных или выражений. Допускается обновление как поля в целом, так и подмножества его значений в соответствии с заданными условиями.

### Удаление записей командой DELETE

Удаление записей из таблиц производится стандартной командой SQL DELETE. Вызов DELETE приводит к необратимым последствиям (исключение составляют тщательно спланированные транзакционные блоки), поэтому удаление данных из базы требует крайней осторожности.

Команда удаления одной или нескольких записей из базы имеет следующий синтаксис:

Секция WHERE почти всегда присутствует в команде DELETE.

### Выборка данных командой SELECT

Центральное место в SQL занимает команда SELECT, предназначенная для построения запросов и выборки данных из таблиц и представлений. Данные, возвращаемые в результате запроса, называются итоговым набором; как и таблицы, они состоят из записей и полей.

Данные итогового набора не хранятся на диске в какой-либо постоянной форме. Итоговый набор является лишь временным представлением данных, полученных в результате запроса. Структура полей итогового набора может соответствовать структуре исходной таблицы, но может и радикально отличаться от нее. Итоговые наборы даже могут содержать поля, выбранные из других таблиц.

Из-за своей особой роли в MSSQL команда SELECT также является самой сложной командой, обладающей многочисленными секциями и параметрами. Ниже приведено общее определение синтаксиса SELECT, а отдельные компоненты рассматриваются в следующих разделах. Термин выражение соответствует имени поля или общему выражению (например, результату операции, в которой участвует значение поля и константа или значения двух полей).

## Описание триггеров и хранимые процедуры.

Триггер – это набор инструкций SQLServer, который выглядит и действует подобно хранимой процедуре, но триггер нельзя вызвать с помощью команды EXEC. Триггеры активизируются при выполнении пользователем определенной инструкции Transact-SQL. Существует два вида триггеров :

- DML запускаются с помощью инструкций INSERT, UPDATE или DELETE.

- DDL с помощью инструкций CREATE, ALTER, DROP.

Если вы не хотите, чтобы триггеру передавались записи, нужно добавить команду ROLLBACK в соответствующее место в коде. Команда ROLLBACK  указывает серверу остановить обработку модификации и запретить транзакцию.

Существует также команда RAISEERROR с помощью которой вы можете отправить сообщение об ошибке пользователю.

Триггеры бывают трех типов INSERT,UPDATE,DELETE.

Триггеры INSERT запускаются при каждой попытке создать новую запись в таблице с помощью команды INSERT. При попытке вставить новую запись в таблицу SQLServer копирует эту запись в таблицу триггеров БД и специальную таблицу **INSERTED.**

Триггеры DELETE используются для предотвращения удаления данных пользователями из БД. При наличии триггера DeleteSQLServer переносит удаляемую запись в логическую таблицу **DELETED,** т.е. записи не исчезают полностью, и вы можете ссылаться на них в коде.

Триггер UPDATE – для перехвата и сверки модификации данных.

Триггер UPDATE использует обе таблицы – **INSERTED и DELETED**. Это объясняется тем, что триггер UPDATE использует два действия - удаление и вставку.

## Режимы аутентификации

**MS SQL Server** поддерживает два режима аутентификации:

* Windows аутентификация (**Windows Authentication mode**) — безопасность задается на уровне операционной системы, т.е. метод, при помощи которого пользователи соединяются с операционной системой, используя учетную запись пользователя Windows. SQL Server принимает учетную запись пользователя, предполагая, что она уже проверена на уровне операционной системы (доверительное соединение).
* Смешанный режим аутентификации (**SQL Server and Windows Authentication mode**) — Смешанный режим дает пользователю возможность соединяться с SQL Server, используя аутентификацию Windows или аутентификацию SQL Server.

Сама **Microsoft** рекомендует (в случаях, когда это возможно) использовать для проверки пользователей аутентификацию **Windows**.

Практическая часть

# Практическая часть

## Создание базы данных

Была разработана структура базы данных «СonfectioneryСompany»

База данных содержит 7 таблиц. Каждая таблица содержит минимум 3 поля, одно из которых является ключевым.

Осуществлено правомерное использование типов.

Созданная база данных находится в третьей нормальной форме.

### Доказательство соответствия третьей нормальной форме

Каждый столбец содержит одно неделимое значение, таблица не содержит повторяющихся строк или групп данных. Таким образом, таблицы находятся в первой нормальной форме.

Любой столбец, который не является ключом (в том числе внешним), зависит от первичного ключа, и база данных находится в первой нормальной форме, поэтому, данная база соответствует второй нормальной форме.

Поскольку база приведена ко второй нормальной форме и каждый не ключевой столбец независим друг от друга, то база данных находится в третьей нормальной форме.

## API

В ходе выполнения курсовой работы, был разработан API для базы данных «СonfectioneryСompany» - было создано оконное приложение на языке C#.

Осуществлены следующие методы:

### Аутентификация пользователя (создание сессии)

Для формы аутентификации (Рис.1) было создано несколько запросов:

**Для проверки наличия введенного логина и верности введенного пароля:**

SELECT count(\*) FROM users WHERE loginn = @username and passwd= @password;

**Для запоминания id пользователя:**

SELECT userr\_code FROM users WHERE loginn = @login

**Для открытия сессии:**

INSERT into sessionns(userr\_id,created\_time) VALUES(@id,getdate());

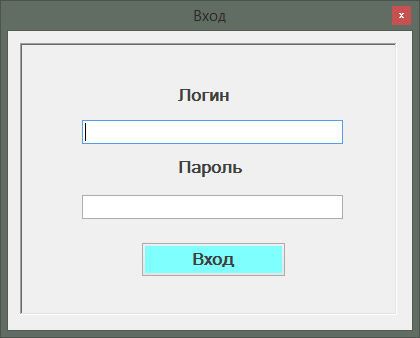


Рис 1. Аутентификация пользователя

### Добавление данных в таблицы

**Хранимая процедура добавления данных на склад**

CREATE PROCEDURE InsertWarehouse @name nvarchar(255),@confectionCount int,@bestBefore date

AS

insert into [Warehouse]([ConfectionId],[ConfectionCount],[BestBefore])

values ((select top 1 [id] from [Confections] where [Name] = @name),@confectionCount,@bestBefore);

GO

**Хранимая процедура добавления данных заказа**

CREATE PROCEDURE InsertOrders @name nvarchar(255),@confectionCount int,@orderDate datetime

AS

insert into [Orders]([ConfectionId],[ConfectionCount],[OrderDate])

values ((select top 1 [id] from [Confections] where [Name] = @name),@confectionCount,@orderDate);

GO

**Хранимая процедура добавления данных состава изделия**

CREATE PROCEDURE InsertConfectionProducts @confectionName nvarchar(255),@productName nvarchar(255)

AS

insert into [ConfectionProducts]([ConfectionId],[ProductId])

values ((select top 1 [id] from [Confections] where [Name] = @confectionName),(select top 1 [id] from [Products] where [Name] = @productName));

GO

### Удаление данных в таблицах

**Хранимая процедура удаления со склада**

CREATE PROCEDURE DeleteWarehouse @id int

AS

delete from [Warehouse] where id = @id

GO

**Хранимая процедура удаления заказа**

CREATE PROCEDURE DeleteOrders @id int

AS

delete from [Orders] where id = @id

GO

**Хранимая процедура удаления состава изделия**

CREATE PROCEDURE DeleteConfectionProducts @id int

AS

delete from [ConfectionProducts] where id = @id

GO

### Изменение данных в таблицах

**Хранимая процедура изменения данных на складе**

CREATE PROCEDURE UpdateWarehouse @confectionCount int,@bestBefore date,@id int

AS

update [Warehouse]

set

[ConfectionCount] = @confectionCount,

[BestBefore] = @bestBefore

where [id] = @id

GO

**Хранимой процедуры для изменения данных заказа не предусмотрено поскольку, обычно никто не меняет заказы, только если админ в базе.**

**Хранимая процедура изменения данных состава изделия**

CREATE PROCEDURE UpdateConfectionProducts @confectionName nvarchar(255),@productName nvarchar(255),@id int

AS

update [ConfectionProducts]

set

[ConfectionId] =(select top 1 [id] from [Confections] where [Name] = @confectionName),

[ProductId] = (select top 1 [id] from [Products] where [Name] = @productName)

where [id] = @id

GO

### Выборка данных из ключевых таблиц по запросам

**Выборка по изделиям. Процедуры на delete update insert VisualStudio сгенерировала автоматически**

select [Name] as 'Название изделия',[Price] as 'Цена(руб.)',[Info] as 'Информация' from [Confections];

**Выборка по продуктам. Процедуры на delete update insert VisualStudio сгенерировала автоматически**

select [Name] as 'Название продукта',[Info] as 'Информация' from [Products];

**Выборка по складу.**

select [Confections].[Name] as 'Название изделия',[ConfectionCount] as 'Количество',[BestBefore] as 'Годен до',[Warehouse].[id] from [Warehouse]

join [Confections] on [Confections].[id] = [Warehouse].[ConfectionId];

**Выборка по заказам**

select [Confections].[Name] as 'Название изделия',[ConfectionCount] as 'Количество',[OrderDate] as 'Дата заказа',[Orders].[id] from [Orders]

join [Confections] on [Confections].[id] = [Orders].[ConfectionId];

**Выборка по составу изделий**

select [Confections].[Name] as 'Название изделия',[Products].[Name] as 'Название продукта',[ConfectionProducts].[id] from [ConfectionProducts]

join [Confections] on [Confections].[id] = [ConfectionProducts].[ConfectionId]

join [Products] on [Products].[id] = [ConfectionProducts].[ProductId];

### Выборка данных из таблиц с объединением результатов

**Хранимая процедура подсчета выручи за период**

CREATE PROCEDURE CountMoneyByPeriod @startDate datetime,@endDate datetime

AS

select sum([Confections].[Price] \* [Orders].[ConfectionCount]) from [Orders]

join [Confections] on [Confections].[id] = [Orders].[ConfectionId]

where [Orders].[OrderDate] between @startDate and @endDate

GO

**Хранимая процедура выборки заказов за период**

CREATE PROCEDURE GetOrdersByPeriod @startDate datetime,@endDate datetime,@confectionName nvarchar(255)

AS

select Confections.Name as 'Название изделия',Confections.Price as 'Цена',Orders.OrderDate as 'Дата заказа' from Orders

join Confections on Confections.id = Orders.ConfectionId

where Orders.OrderDate between @startDate and @endDate and CHARINDEX(LOWER(@confectionName),LOWER(Confections.Name)) > 0

order by Confections.Name

GO

## Триггеры

## Триггер проверки наличия изделий на складе при вставка нового заказа

## CREATE TRIGGER [dbo].[CheckConfectionCount]

## ON [dbo].[Orders]

## INSTEAD OF INSERT

## AS

## BEGIN

## SET NOCOUNT ON;

## if (select ConfectionCount from [Warehouse]

## where [ConfectionId] = (select [id] FROM inserted)) < (select [ConfectionCount] FROM inserted)

## rollback

## print'Недостаточно изделий на складе!'

## END

## Триггер проверки наличия изделий на складе при обновлении заказа(только для бд, поскольку в программе не реализовано обновление данных в таблице заказов)

## CREATE TRIGGER [dbo].[CheckConfectionCountUpdate]

## ON [dbo].[Orders]

## AFTER Update

## AS

## BEGIN

## SET NOCOUNT ON;

## if (select [ConfectionCount] from [Warehouse]

## where [ConfectionId] = (select [ConfectionId] FROM inserted)) < (select [ConfectionCount] FROM inserted)

## rollback

## print'Недостаточно изделий на складе!'

## END

## Триггер уменьшения количества изделий на складе после вставки в заказы

## CREATE TRIGGER [dbo].[UpdateConfectionCountInWarehouse]

## ON [dbo].[Orders]

## AFTER INSERT

## AS

## BEGIN

## SET NOCOUNT ON;

## Update [Warehouse] set [ConfectionCount] =([Warehouse].[ConfectionCount] - (select [ConfectionCount] FROM inserted))

## where [ConfectionId] = (select [ConfectionId] FROM inserted)

## END

## Просмотр результатов запросов

Все исходные таблицы, таблицы-результаты запросов выводятся с помощью DataGridView, результаты запросов можно увидеть при обновлении таблиц.

Пример вывода таблиц-результатов на экран представлен на Рис.2

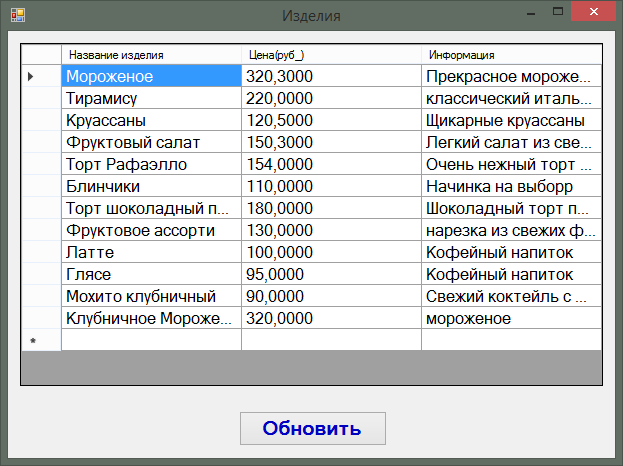
****

Рис. 2. Результат вывода таблиц

На Рис.3. представлено основное окно API (меню)

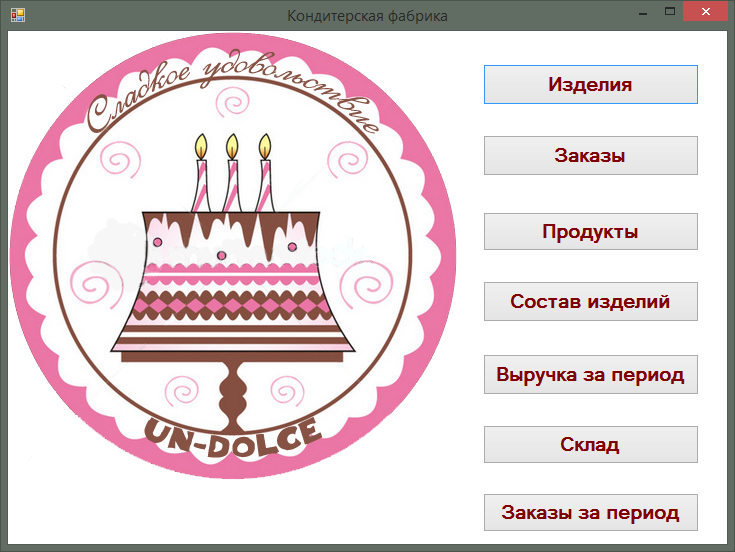


Рис.3. Меню