Министерство науки и высшего образования РФ  
ФГАОУ ВПО

Национальный исследовательский технологический университет «МИСиС»

Институт Информационных технологий и компьютерных наук (ИТКН)

Кафедра Инфокоммуникационных технологий (ИКТ)

**Контрольное домашнее задание**

по дисциплине «Базы данных»

на тему «Продуктовый супермаркет.»

Выполнила:

Кирилина В.А.,

Уч. г. БИВТ-20-3

Проверила:   
Валова А.А.

Москва, 2021г.

[Постановка задачи 3](#_Toc91110517)

[Описание структуры БД 4](#_Toc91110518)

[Вербальная модель 4](#_Toc91110519)

[Реляционная модель 5](#_Toc91110520)

[Анализ функциональных зависимостей 5](#_Toc91110521)

[Заполнение БД информацией 7](#_Toc91110522)

[Описание представлений 8](#_Toc91110523)

[Описание функций 9](#_Toc91110524)

[Описание хранимых процедур 12](#_Toc91110525)

[Описание триггеров 15](#_Toc91110526)

[Пример работы БД с использованием созданных объектов 15](#_Toc91110527)

[Список литературы 20](#_Toc91110528)

# Постановка задачи

Данная база данных создается в рамках учебного курса «Базы данных».

По заданию КДЗ (контрольное домашнее задание) необходимо создать базу данных, состоящую не менее чем из 10 таблиц. База данных должна осуществлять:

1. Учет поставщиков и поставок;
2. Учет продаж по отделам и по кассам;
3. Учет сотрудников;
4. Подсчет остатков товара;
5. Оформление заказов на товары, запасы которых подходят к концу;
6. Подведение финансовых итогов дня (по отделам и по кассам);
7. Учет продукции, у которых срок годности подходит к концу.

Кроме того, в базе данных должны быть реализованы представления, функции, хранимые процедуры, и триггеры (минимум по три объекта каждого типа)

# Описание структуры БД

## Вербальная модель

База данных состоит из 13 таблиц:

* Product - основная информация о продуктах, продаваемых в магазине. Атрибуты таблицы: ProductId, Name, Price, DepartmentId, ManufacturerId, DateOfManufacture, ExpirationId, Photo, SupplyId
* Storage - склад магазина. Атрибуты таблицы: ProductId, ShelfId, Quantity
* Manufacturer - производитель товара. Атрибуты таблицы: ManufacturerId, Name, Phone, Address, Email
* Supplier - поставщик продукции. Атрибуты таблицы: SupplierId, Name, Phone, Address, Email
* Supply - поставка продукции. Атрибуты таблицы: SupplyId, SupplierId, Date of Supply
* Department - отдел в магазине. Атрибуты таблицы: DepartmentId, Name, HeadworkerId
* Employer - учет сотрудников магазина. Атрибуты таблицы: EmployeeId, FirstName, SecondName, Phone, Salary, DateOfBirth, Position, DepartmentId, DateOfEmployment
* Vacation - учет отпусков сотрудников. Атрибуты таблицы: VacationId, EmployeeId, Start, End, NumberOfVacation
* Purchase - покупки, совершенные в магазине. Атрибуты таблицы: ID, CustomerId, Total Expenses, Cash\_registerId
* Details - дополнительные сведения о покупке в магазине. Атрибуты таблицы: ID, ProductId, Quantity, Date
* Cash\_register - сведения о кассах в магазине. Атрибуты таблицы: Cash\_registerId, EmployeeId, Revenue
* Customer - данные о покупателях в магазине. Атрибуты таблицы: CustomerId, FirstName, LastName, Phone, Email, DiscountCardId
* DiscountCard - данные о дисконтных картах покупателей (программа лояльности). Атрибуты таблицы: ID, TotalExpenses, DiscountPercent

## **Image**Реляционная модель

Рис.1. Диаграмма базы данных

База данных нормализована (то есть данные организованы, устранены избыточные функциональные зависимости между атрибутами таблиц) и приведена к третьей нормальной форме. Это означает, что отношения находится во 2НФ и каждый не ключевой атрибут не транзитивно зависит от первичного ключа.

## Анализ функциональных зависимостей

Проанализируем функциональные зависимости в некоторых таблицах, которые входят в базу данных:

1. Таблица Product имеет простой первичный ключ ProductId, который определяет атрибуты Name, Price, DepartmentId, и Photo. Между ними однозначная зависимость.
2. Таблица Storage имеет составной первичный ключ, который состоит из ProductId и ShelfId. Частичная зависимость, связь «один-ко многим»
3. Таблица Manufacturer имеет простой первичный ключ ManufacturerId, который однозначно определяет остальные атрибуты таблицы. Cвязь «один-ко многим»
4. Таблица DiscountCard содержит простой первичный ключ ID, он однозначно определяет остальные определяет. Связь «один-ко многим»
5. Таблица Supply имеет простой первичный ключ SupplyId, который однозначно определяет остальные атрибуты таблицы. Cвязь «один-ко многим»
6. Таблица Department имеет простой первичный ключ DepartmentId. Связь с остальными атрибутами таблицы «один-ко многим»

# Заполнение БД информацией

Базу данных можно заполнить несколькими способами: импортировать данные из открытых источников или заполнить в ручном режиме, использовать механизмы генерации тестовых данных.

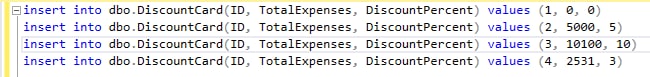
Для заполнения БД информацией я использовала команду INSERT 

Рис 2. Пример заполнения таблицы DiscountCard

После применения команды INSERT таблица выглядит так:

Изображение выглядит как стол

Автоматически созданное описание

Рис.3. Пример заполненной таблицы DiscountCard

# Описание представлений

Представление (или Viewers) – это виртуальная таблица, которая динамически извлекает данные с помощью запросов. Для создания представления используется CREATE VIEW, а для вызова уже созданного представления - ALTER VIEW.

В моей базе данных три представления:

1. Учет товаров, запасы которых подходят к концу

SELECT dbo.Storage.ProductId, dbo.Product.Name

FROM dbo.Storage INNER JOIN

dbo.Product ON dbo.Storage.ProductId = dbo.Product.ProductId

WHERE (dbo.Storage.Quantity < 10)

1. Учет покупок, которые были совершены сегодня

SELECT dbo.Purchase.ID, dbo.Purchase.CustomerID, dbo.Purchase.TotalExcpenses, dbo.Purchase.Cash\_registerID, dbo.Details.ID AS Expr1, dbo.Details.ProductId, dbo.Details.Quantity, dbo.Details.Date

FROM dbo.Purchase INNER JOIN

dbo.Details ON dbo.Purchase.ID = dbo.Details.ID

WHERE (dbo.Details.Date = CAST(GETDATE() AS Date))

1. Подсчет прибыли по кассам (сортировка от менее прибыльных до самых прибыльных)

SELECT TOP (100) PERCENT dbo.Cash\_register.Cash\_registerID, dbo.Cash\_register.EmployeeId, dbo.Cash\_register.Revenue, { fn CONCAT(dbo.Employee.FirstName, ' ' + dbo.Employee.SecondName) } AS Name

FROM dbo.Cash\_register INNER JOIN

dbo.Employee ON dbo.Cash\_register.EmployeeId dbo.Employee.EmplyeeId

ORDER BY dbo.Cash\_register.Revenue

# Описание функций

Функции в SQL Server бывают встроенными и определяемые пользователем.

Встроенные функции разделяются на группы: агрегатные (count, max, min и др.), ранжирующие (rank и др.), функции для работы с данными (getdate, day, month, year и др.) и многие другие.

Если пользователь хочет создать свою функцию, то для этого используется инструкция CREATE FUNCTION

В моей базе данных 3 встроенные функции:

1. В одном из представлений я использовала функцию CONCAT(str1, str2) для конкатенации (сложения) двух строк
2. В процедуре CustomerStatistic для нахождения максимальной стоимости покупки конкретного покупателя использована агрегатная функция MAX
3. В процедуре CustomerStatistic для нахождения средней стоимости покупок конкретного покупателя я использовала агрегатную функцию AVG

Также я написала 3 пользовательских функции:

1. Функция dbo.error используется для обработок ошибок и выводит какая произошла ошибка

use [KirilinaVA\_bivt-20-3\_Shop]

go

alter function dbo.error(@code varchar(50))

returns varchar(50)

as

begin

declare @err varchar(50)

set @err = concat('Произошла ошибка! ', @code)

return (@err);

end

go

1. Функция dbo.SupplierProduct принимает id продукта, возвращает id поставщика этого продукта

alter function dbo.SupplierProduct (@id int)

returns varchar(max)

begin

declare @supId int

if (@id <= 0)

begin

return (dbo.error('ProductId не может быть меньше 0!'))

end

if @id = 1 or @id = 2

set @supId = 1

else if @id = 3 or @id = 5

set @supId = 2

else if @id = 4

set @supId = 3

else if @id > (select max (ProductId) from Product)

return (dbo.error('Товара с таким Id нет в ассортименте магазина!'))

return cast (@supId as varchar(max))

end

1. Функция dbo.createSupply подсказывает кому из поставщиков нужно написать, чтобы оформить поставку

alter function dbo.createSupply (@id int)

returns varchar(max)

begin

declare @info varchar(max)

declare @supId int

--declare @today date = CAST(GETDATE() AS Date)

set @info = concat('Нужно заказать товар с id = ', @id)

set @supId = try\_convert(int, dbo.SupplierProduct(@id))

set @info = @info + '. Для этого позвоните ' + (select name from Supplier where Supplier.SupplierId = @supId) + ' по номеру ' + (select Phone from Supplier where Supplier.SupplierId = @supId)

---declare @newSupply int = (select max(SupplyId) from Supply) + 1

return @info

end

Также я использовала курсор, тело которого включало все эти функции:

declare @ProductId int

declare newOrder cursor local for

select ProductId from View\_ProductIsRunningOut

open newOrder

fetch next from newOrder into @ProductId

while @@FETCH\_STATUS = 0

begin

--declare @runningId int = @ProductId

declare @supId int = try\_convert(int, dbo.SupplierProduct(@ProductId))

declare @today date = CAST(GETDATE() AS Date)

declare @newSup int = (select max(SupplyId) from Supply) + 1

print(dbo.CreateSupply( @ProductId))

insert Supply (SupplyId, SupplierId, DateOfSupply)

values (@newSup, @supId, @today)

fetch next from newOrder into @ProductId

end

close newOrder

deallocate newOrder

go

# Описание хранимых процедур

Хранимые процедуры – это группа инструкций, которые выполняются как единое. Преимущества хранимых процедур в том, что они более безопасны и выполняются быстрее чем обычные sql- инструкции

В моей БД три хранимых процедуры:

1. Процедура для учета сотрудников, которые сейчас находятся в отпуске

USE [KirilinaVA\_bivt-20-3\_Shop]

GO

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

ALTER PROCEDURE [dbo].[EmployeeVacation]

@status int

AS

BEGIN

declare @dateToday date

set @dateToday = CAST(GETDATE() AS Date)

SET NOCOUNT ON;

if @status = 1

SELECT dbo.Employee.\*

FROM dbo.Employee INNER JOIN

dbo.Vacation ON dbo.Employee.EmplyeeId = dbo.Vacation.EmpoyeeId

WHERE @dateToday between Start and [End]

else print 'Неизвестная команда!'

END

1. Процедура для просмотра поставщиков, которые совершили поставку сегодня

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE PROCEDURE findSupplier

@date date

AS

BEGIN

SET NOCOUNT ON;

SELECT dbo.Supplier.\*

FROM dbo.Supplier INNER JOIN

dbo.Supply ON dbo.Supplier.SupplierId = dbo.Supply.SupplierId

WHERE DateOfSupply = @date

END

GO

1. Процедура для статистики покупателей магазинов

USE [KirilinaVA\_bivt-20-3\_Shop]

GO

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

alter PROCEDURE CustomerStatistic

@customerId int

AS

BEGIN

SET NOCOUNT ON;

declare @dateToday date

set @dateToday = CAST(GETDATE() AS Date)

declare @discount int --скидка покупателя

declare @purchaseToday int -- сегодняшняя покупка покупателя

declare @purchaseMax int -- максимальная покупка покупателя

declare @purchaseAvg float -- средняя стоимость покупки

set @discount = (select dbo.DiscountCard.DiscountPercent

FROM dbo.Customer INNER JOIN dbo.DiscountCard ON dbo.Customer.DiscountCardId = dbo.DiscountCard.ID

WHERE dbo.Customer.CustomerID = @customerId)

set @purchaseMax = (SELECT max(dbo.Purchase.TotalExcpenses)

FROM dbo.Customer INNER JOIN dbo.Purchase ON dbo.Customer.CustomerID = dbo.Purchase.CustomerID inner join dbo.Details on dbo.Purchase.ID = dbo.Details.ID

WHERE dbo.Customer.CustomerID = @customerId)

set @purchaseToday = (SELECT dbo.Purchase.TotalExcpenses

FROM dbo.Customer INNER JOIN dbo.Purchase ON dbo.Customer.CustomerID = dbo.Purchase.CustomerID inner join dbo.Details on dbo.Purchase.ID = dbo.Details.ID

--WHERE dbo.Customer.CustomerID = @customerId)

WHERE dbo.Details.Date = @dateToday and dbo.Customer.CustomerID = @customerId)

set @purchaseAvg = (SELECT avg(dbo.Purchase.TotalExcpenses) as AvgPerchase

FROM dbo.Customer INNER JOIN dbo.Purchase ON dbo.Customer.CustomerID = dbo.Purchase.CustomerID

WHERE dbo.Customer.CustomerID = @customerId)

if (@purchaseMax is null)

set @purchaseMax = 0

if (@purchaseAvg is null)

set @purchaseAvg = 0

if (@purchaseToday is null)

set @purchaseToday = 0

print(concat('Скидка покупателя: ', @discount)+'%')

print(concat('Максимальная покупка покупателя: ', @purchaseMax))

print(concat('Средняя стоимость покупок: ', @purchaseAvg))

print(concat('Cегодняшние покупки покупателя: ', @purchaseToday))

END

GO

# Описание триггеров

Триггеры – это специальный хранимые процедуры, которые автоматически запускаются при выполнении определенного действия над таблицей или представлением. Существует два типа триггеров: AFTER (после действия, только для таблиц) и INSTEAD OF (вместо действия)

В моей Базе данных содержится три триггера:

1. Триггер для автоматического добавления нового покупателя в программу лояльности

USE [KirilinaVA\_bivt-20-3\_Shop];

GO

CREATE TRIGGER addDiscountCard

ON Customer

AFTER INSERT

AS

declare @discount int

set @discount = (select DiscountCardId from inserted)

INSERT INTO DiscountCard (ID, TotalExpenses, DiscountPercent)

VALUES (@discount, 0, 0)

1. Триггер, который учитывает на складе, что товар удалили из ассортимента из магазина

USE [KirilinaVA\_bivt-20-3\_Shop];

GO

CREATE TRIGGER deleteProductFromStorage

ON Product

AFTER DELETE

AS

declare @id int

set @id = (select ProductId from inserted)

UPDATE Storage

set ShelfId = 0

where Storage.ProductId = @id

1. Триггер, который при совершении покупки, добавляет стоимость покупки к «прибыли» кассы, на которой была совершена покупка

USE [KirilinaVA\_bivt-20-3\_Shop]

GO

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

ALTER TRIGGER [dbo].[addRevenue]

ON [dbo].[Details]

AFTER INSERT

AS

declare @revenue int

declare @productPrice int

declare @cashredisterId int

set @productPrice = (select Price

from Product

where ProductId in (select ProductId from inserted))

set @revenue = (select Quantity from inserted)\*@productPrice

set @cashredisterId = (select Purchase.Cash\_registerID

from Purchase inner join Details on Purchase.ID=Details.ID

where Details.ID in (select ID from inserted))

UPDATE Cash\_register

set Revenue = Revenue + @revenue

where Cash\_registerID = @cashredisterId

# Пример работы БД с использованием созданных объектов

Как говорилось раннее, данная база данных используется в продуктовом магазине. Рассмотрим решение задачи «обслуживание покупателя» с помощью созданных объектов.

Если в магазин пришел новый покупатель, которого не было ранее, то триггер addDiscountCard автоматически внесет его в программу лояльности магазина. Суммарные траты покупателя в программе лояльности учитываются только после того, как покупатель зарегистрирован (то есть первая покупка нового покупателя не учитывается для определения его скидки).

По каждому покупателю можно получить статистику, которая выведет скидку покупателя, стоимость его максимальной покупки, среднюю стоимость покупок, а также стоимость покупки, которую покупатель совершил сегодня. Для этого используется процедура CustomerStatistic.

Изображение выглядит как текст

Автоматически созданное описание

Рис.4. Пример статистики для покупателя с id=3

Если товар убирают из ассортимента магазина, то триггер deleteProductFromStorage автоматически обновляет информацию в таблице Storage о том, что кол-во такого товара на складе теперь 0.

Кроме вышеописанных объектов, можно использовать и другие триггеры, процедуры и представления для других задач. Например, для удобства сотрудников можно использовать процедуру findSupplier, которая выводит информацию о поставщиках, которые сегодня поставляли товар, или процедуру EmployeeVacation, которая при аргументе 1 выводит список сотрудников, которые сейчас находятся в отпуске (как видно из рис.5. в данный момент ни один сотрудник не находится в отпуске).

Изображение выглядит как текст

Автоматически созданное описание

Рис.5 Результат выполнения процедур EmployeeVacation и findSupplier

Вероятно, сотрудникам будет удобно смотреть какие товары подходят к концу. Для этого они могут воспользоваться представлением View\_ProductIsRunningOut, которое выводит список товаров, количество которых на складе меньше 10. Также можно использовать курсор newOrder, который для каждого товара из представления View\_ProductIsRunningOut оформляет новую поставку товара. Для учета прибыли могут быть использованы представления View\_TodayPurchase (учет сегодняшних покупок) и View\_CashRevenueAsc, которое ранжирует список касс магазина по прибыльности (в порядке возрастания).

Изображение выглядит как текст

Автоматически созданное описание

Рис. 6. Пример работы представлений

В представлении View\_TodayPurchase пустая таблица, так как на момент 02:29 23.12.21 магазин еще не открылся, и сегодня (23.12.21) никто не совершил покупки.

**Изображение выглядит как текст

Автоматически созданное описание**

Рис.7. Результат работы курсора

# Список литературы

1. [Курс лекций В.А. Дударева по базам данных](https://www.youtube.com/watch?v=41DvlCcXYzk&list=PL29EQ0Er2RStUeOavl-6tTtJB0TJ6jVmz)
2. [Документация по SQL](https://docs.microsoft.com/ru-ru/sql/sql-server/what-s-new-in-sql-server-ver15?view=sql-server-ver15)
3. [Хранимые процедуры](https://docs.microsoft.com/ru-ru/sql/relational-databases/stored-procedures/stored-procedures-database-engine?view=sql-server-ver15)
4. [Таблицы](https://docs.microsoft.com/ru-ru/sql/relational-databases/tables/tables?view=sql-server-ver15)
5. [Отслеживание изменений](https://docs.microsoft.com/ru-ru/sql/relational-databases/track-changes/track-data-changes-sql-server?view=sql-server-ver15)
6. [Триггеры](https://docs.microsoft.com/ru-ru/sql/relational-databases/triggers/logon-triggers?view=sql-server-ver15)
7. [Определяемые пользователем функции](https://docs.microsoft.com/ru-ru/sql/relational-databases/user-defined-functions/user-defined-functions?view=sql-server-ver15)
8. [Представления](https://docs.microsoft.com/ru-ru/sql/relational-databases/views/views?view=sql-server-ver15)
9. [Функции, используемые в БД](https://docs.microsoft.com/ru-ru/sql/t-sql/functions/functions?view=sql-server-ver15)
10. [Курс «SQL for Data Science»](https://www.coursera.org/learn/sql-for-data-science/home/welcome)