**Реляционная модель БД**



**Скрипт создания БД**

USE master

GO

CREATE DATABASE NewMart

ON

(

NAME = NewMart,

FILENAME = 'C:\DB\NewMart.mdf',

SIZE = 512MB,

MAXSIZE = Unlimited,

FILEGROWTH = 256MB

)

GO

USE NewMart

GO

create table Manufacturer

(

ManufacturerID int primary key not null,

Name varchar(50) not null,

)

GO

create table Category

(

CategoryID int primary key not null,

Name varchar(50) not null,

)

GO

create table Product

(

ProductID int primary key not null,

ManufacturerID int not null,

CategoryID int not null,

Name varchar(50) not null,

InStock int,

Cost money,

Info text,

Picture image,

constraint FK\_ID1 foreign key(ManufacturerID) references Manufacturer(ManufacturerID),

constraint FK\_ID2 foreign key(CategoryID) references Category(CategoryID)

)

GO

create table Location

(

LocationID int primary key not null,

LocationType varchar(10) not null,

LocationAddress varchar(50) not null,

)

GO

create table Worker

(

WorkerID int primary key not null,

LocationID int not null,

Position varchar(20) not null,

Salary money,

Name varchar(20),

FamilyName varchar(20),

Gender char,

BirthDate datetime not null,

EmployDate datetime not null,

FireDate datetime,

constraint FK\_ID3 foreign key(LocationID) references Location(LocationID),

)

GO

create table [Order]

(

OrderID int primary key not null,

LocationID int not null,

Cost money not null,

DoneTime datetime,

ArrivalTime datetime not null,

constraint FK\_ID4 foreign key(LocationID) references Location(LocationID),

)

create table OrderProducts

(

ProductID int not null,

OrderID int not null,

Ammount int not null,

constraint FK\_ID5 foreign key(ProductID) references Product(ProductID),

constraint FK\_ID6 foreign key(OrderID) references [Order](OrderID)

)

**Триггеры**

**1)**

Create Trigger InsertOrder ON OrderProducts

AFTER INSERT

AS

BEGIN

DECLARE @InsertedCheck Cursor

SET @InsertedCheck = CURSOR FORWARD\_ONLY FOR

select OrderID, ProductID, ammount from OrderProducts with(index(POIndx))

open @InsertedCheck

DECLARE @OID INT

DECLARE @PID INT

DECLARE @PA INT

DECLARE @Summary MONEY

FETCH NEXT FROM @InsertedCheck into @OID, @PID, @PA

WHILE @@FETCH\_STATUS = 0 BEGIN

SET @Summary = dbo.Fun2(@PID, @PA)

UPDATE [ORDER]

SET Cost = Cost + @Summary where OrderID = @OID

SET @SUMMARY = 0

FETCH NEXT FROM @InsertedCheck into @OID, @PID, @PA

END

Close @InsertedCheck

END

**2)**

Create Trigger UpdateAmmount ON [Order]

AFTER UPDATE

AS

BEGIN

Declare @ProductsSelect Cursor

Declare @DoneCheck Cursor

set @DoneCheck = CURSOR FORWARD\_ONLY FOR

select OrderID, DoneTime from inserted

open @DoneCheck

Declare @OID int

Declare @DTC datetime

Declare @PID int

Declare @PA int

FETCH NEXT FROM @DoneCheck INTO @OID, @DTC

WHILE @@FETCH\_STATUS = 0 BEGIN

IF ((@DTC is not null) and

((select DoneTime from deleted where OrderID = @OID) is null))

BEGIN

set @ProductsSelect = CURSOR FORWARD\_ONLY FOR

select ProductID, ammount from OrderProducts where OrderID=@OID

open @ProductsSelect

FETCH NEXT FROM @ProductsSelect INTO @PID, @PA

WHILE @@FETCH\_STATUS = 0 BEGIN

UPDATE Product

Set InStock = InStock - @PA where ProductID = @PID

FETCH NEXT FROM @ProductsSelect INTO @PID, @PA

END

Close @ProductsSelect

END

FETCH NEXT FROM @DoneCheck INTO @OID, @DTC

END

Close @DoneCheck

END

**3)**

Create Trigger RewindOrder ON OrderProducts

AFTER INSERT

AS

BEGIN

DECLARE @DeletedCheck Cursor

SET @DeletedCheck = CURSOR FORWARD\_ONLY FOR

select OrderID, ProductID, ammount from OrderProducts with(index(POIndx))

open @DeletedCheck

DECLARE @OID INT

DECLARE @PID INT

DECLARE @PA INT

DECLARE @Summary MONEY

FETCH NEXT FROM @DeletedCheck into @OID, @PID, @PA

WHILE @@FETCH\_STATUS = 0 BEGIN

SET @Summary = dbo.Fun2(@PID, @PA)

UPDATE [ORDER]

SET Cost = Cost - @Summary where OrderID = @OID

SET @SUMMARY = 0

FETCH NEXT FROM @DeletedCheck into @OID, @PID, @PA

END

Close @DeletedCheck

END

**9) Функции**

**1)**

create Function Fun1 (@MN varchar(50))

RETURNS TABLE

AS

RETURN

(Select ProductID from Product p

inner join Manufacturer m

on p.ManufacturerID = m.ManufacturerID

where m.Name = @MN

);

**2)**

create Function Fun2 (@PID int, @ammount int)

RETURNS INT

AS

BEGIN

RETURN

(@ammount \* (select Cost from Product with(index(PrdctIndx)) where ProductID = @PID));

END

Используется в триггерах на INSERT и на DELETE

**3)**

create Function Fun3 (@PID int)

RETURNS varchar(101)

AS

BEGIN

Return

(

(Select m.Name from Manufacturer m inner join Product p

on p.ManufacturerID = m.ManufacturerID where p.ProductID=@PID)

+ ‘ ‘ +

(Select Name from Product where ProductID=@PID)

);

END

**Курсоры**

**1)@DoneCheck (использован в триггере на UPDATE)**

Declare @DoneCheck Cursor

set @DoneCheck = CURSOR FORWARD\_ONLY FOR

select OrderID, DoneTime from inserted

open @DoneCheck

**2)@ProductsSelect (использован в триггере на UPDATE)**

Declare @ProductsSelect Cursor

set @ProductsSelect = CURSOR FORWARD\_ONLY FOR

select ProductID, ammount from OrderProducts where OrderID=@OID

open @ProductsSelect

**3)@InsertedCheck (использован в триггере на INSERT)**

DECLARE @InsertedCheck Cursor

SET @InsertedCheck = CURSOR FORWARD\_ONLY FOR

select OrderID, ProductID, ammount from OrderProducts with(index(POIndx))open @InsertedCheck

Точный аналог в лице @DeletedCheck использован в триггере на DELETE

**Индексы**

**1)Nonclustered on Product**

Create nonclustered Index PrdctIndx on Product(ProductID, Cost)

Используется в функции Fun2

**2)Clustered on OrderProducts**

Create Clustered Index POIndx on OrderProducts(OrderID,ProductID,ammount)

Использован в курсоре @InsertedCheck

**3)Unique on Order**

Create Unique Index OrdrIndx on [Order](OrderID,DoneTime)