

# Хранимые процедуры в T-SQL

**Хранимая процедура в SQL Server** — это группа из одной инструкции Transact-SQL или нескольких или ссылка на метод среды CLR Microsoft .NET Framework. Хранимые процедуры используются для сохранения на сервере повторно используемого кода.

### **Назначение**

- обрабатывают входные параметры и возвращают вызывающей программе значения в виде выходных параметров;
- содержат программные инструкции, которые выполняют операции в базе данных, включая вызов других процедур;
- возвращают значение состояния вызывающей программе, таким образом передавая сведения об успешном или неуспешном завершении (и причины последнего).

## Создание/изменение хранимой процедуры

```
CREATE [ OR ALTER ] { PROC | PROCEDURE }  
    [schema_name.] procedure_name [ ; number ]  
    [ { @parameter [ type_schema_name. ] data_type }  
        [ VARYING ] [ = default ] [ OUT | OUTPUT | [READONLY]  
    ] [ ,...n ]  
    [ WITH <procedure_option> [ ,...n ] ]  
    [ FOR REPLICATION ]  
AS { [ BEGIN ] sql_statement [;] [ ...n ] [ END ] }  
[;]
```

```
<procedure_option> ::=  
    [ ENCRYPTION ]  
    [ RECOMPILE ]  
    [ EXECUTE AS Clause ]
```

## Упрощенный вариант

```
CREATE PROCEDURE procedure_name (  
    @parameter1, @parameter2, ... @parameterN )  
AS  
BEGIN  
    --Инструкции, реализующие алгоритм, sql_statement  
END
```

## Аргументы

*schema\_name* — имя схемы, к которой относится процедура.

; *число* — целочисленный аргумент, используемый для группирования одноименных процедур (компонент будет удален в следующих версиях).

@ *parameter* — параметр, объявленный в процедуре.

[ *type\_schema\_name* . ] *data\_type* — тип данных параметра и схема, к которой относится этот тип.

VARYING — указывает результирующий набор, поддерживаемый в качестве выходного параметра. Область применения — только параметры **cursor** .

*default* — значение параметра по умолчанию.

OUT | OUTPUT — показывает, что параметр является выходным.

READONLY — указывает, что параметр не может быть обновлен или изменен в тексте процедуры.

RECOMPILE — указывает, что Компонент Database Engine не кэширует план запроса для этой процедуры, что вызывает ее компиляцию при каждом выполнении.

ENCRYPTION - Показывает, что SQL Server выполняет запутывание исходного текста инструкции CREATE PROCEDURE.

EXECUTE AS *clause* — определяет контекст безопасности, в котором должна быть выполнена процедура.

FOR REPLICATION - Указывает, что процедура создается для репликации.

{ [ BEGIN ] *sql\_statement* [;] [ ... *n* ] [ END ] } — одна или несколько инструкций Transact-SQL, составляющих текст процедуры.

## SET NOCOUNT

Запрещает вывод количества строк, на которые влияет инструкция Transact-SQL или хранимая процедура, в составе результирующего набора.

Инструкция SET NOCOUNT ON запрещает всем инструкциям хранимой процедуры отправлять клиенту сообщения DONE\_IN\_PROC. Для хранимых процедур с несколькими инструкциями, не возвращающих большое количество фактических данных, или для процедур, содержащих циклы Transact-SQL, установка в инструкции SET NOCOUNT параметра ON может значительно повысить производительность за счет существенного снижения объема сетевого трафика.

## Выполнение хранимой процедуры

Существует два способа выполнения хранимой процедуры:

- вызов процедуры приложением или пользователем.
- настройка автоматического выполнения процедуры при запуске экземпляра SQL Server.

Если процедура вызывается приложением или пользователем, то в вызове явно указывается ключевое слово Transact-SQL **EXECUTE** или **EXEC**.

При выполнении определяемой пользователем процедуры рекомендуется дополнительно указывать имя схемы. Это позволяет немного увеличить производительность, поскольку компоненту Компонент Database Engine не нужно выполнять поиск в нескольких схемах. Также исключается выполнение неправильной процедуры в случае, если в нескольких схемах базы данных имеются процедуры с одним именем.

## Указание параметров

Путем указания параметров процедуры вызывающие программы могут передавать значения в тело процедуры. Эти значения могут использоваться для разных целей во время исполнения процедуры. Параметры процедуры могут также возвращать значения вызывающей программе, если параметр помечен признаком OUTPUT.

Хранимая процедура может иметь не более 2100 параметров, каждый из которых имеет имя, тип данных и направление. При необходимости параметрам можно задавать значения по умолчанию.

Значения параметра, переданные при вызове процедуры, должны быть константами или переменными. Имя функции не может быть значением параметра. Переменные могут быть пользовательскими или системными, например @@spid.

При создании процедуры и объявлении имени параметра, последнее должно начинаться с единичного символа @ и быть уникальным для всей процедуры.

Явные имена параметров и задание значений каждому параметру в процедуре позволяет передавать параметры в любом порядке.

## Указание параметров

**Вызов процедуры без указания параметров, значения параметров устанавливаются равным значениям по умолчанию**

EXECUTE procedure\_name

**Вызов процедуры с явным указанием значений параметров**

EXEC procedure\_name @param1 = значение1, @param2 = значение2,  
@paramN = значениеN

**Вызов процедуры без указания имен параметров**

EXEC procedure\_name значение1, значение2, значениеN



## Пример. Процедура возвращает имя базы данных

```
CREATE PROCEDURE What_DB_is_this  
AS  
SELECT DB_NAME() AS ThisDB;
```

Вызов хранимой процедуры: EXEC What\_DB\_is\_this;

**Примечание:** DB\_NAME ( [ database\_id ] ) - возвращает имя указанной базы данных. Если в вызове DB\_NAME аргумент *database\_id* не указан, функция DB\_NAME возвращает имя текущей базы данных.

```
CREATE PROC What_DB_is_that @ID INT  
AS  
SELECT DB_NAME(@ID) AS ThatDB;
```

Вызов хранимой процедуры: EXEC What\_DB\_is\_that 2  
-- возвращает tempdb.

## Пример

```
CREATE PROCEDURE TestProcedure (  
--Входящие параметры  
@CategoryId INT, @ProductName VARCHAR(100), @Price MONEY = 0  
)  
AS  
BEGIN  
    SET NOCOUNT ON  
    --Удаление лишних пробелов в начале и в конце символьной строки  
    SET @ProductName = LTRIM(RTRIM(@ProductName));  
    --Добавляем новую запись  
    INSERT INTO TestTable(CategoryId, ProductName, Price) VALUES  
    (@CategoryId, @ProductName, @Price)  
    --Возвращаем данные  
    SELECT * FROM TestTable WHERE CategoryId = @CategoryId  
END  
GO  
  
--Вызываем процедуру, не указывая имен параметров  
EXEC TestProcedure 1, 'Тестовый товар 3', 400
```

# Возврат данных из хранимой процедуры

Существует три способа возврата данных из процедуры в вызывающую программу:

- результатирующие наборы,
- параметры вывода,
- коды возврата.

## Возврат данных с помощью результирующих наборов

Если включить инструкцию SELECT в тело хранимой процедуры, то строки, указанные инструкцией SELECT, будут отправляться непосредственно клиенту.

Для больших результирующих наборов выполнение хранимой процедуры не перейдет к следующей инструкции, пока результирующий набор не будет полностью передан клиенту.

Для небольших результирующих наборов результаты будут буферизированы для возврата клиенту, а выполнение продолжится. Если при выполнении хранимой процедуры запускается несколько таких инструкций SELECT, клиенту отправляется несколько результирующих наборов.

```
CREATE PROCEDURE dbo.uspЭффективностьПродавцов
AS
SELECT ФИО, Сумма
FROM dbo.Продавцы
INNER JOIN dbo.Продажи
ON Продавцы.ID = Продажи.IDПродавца
RETURN
```

## Возврат данных с помощью выходного параметра

Процедура может возвращать текущее значение параметра в вызываемой программе при завершении работы при указании ключевого слова OUTPUT для параметра в определении процедуры. Чтобы сохранить значение параметра в переменной, которая может быть использована в вызываемой программе, при выполнении процедуры вызываемая программа должна использовать ключевое слово OUTPUT.

```
CREATE PROCEDURE dbo.uspЭффективностьПродавцаФИО
@ИмяПродавца nvarchar(50),
@СуммаПродаж money OUTPUT
AS
    SET NOCOUNT ON;
    SELECT @СуммаПродаж = Сумма FROM dbo.Продавцы
    INNER JOIN dbo.Продажи ON Продавцы.ID = Продажи.IDПродавца
    WHERE ФИО = @ИмяПродавца;
    RETURN
GO
```

Входные значения также могут быть указаны для параметров OUTPUT при выполнении процедуры.

## Возврат данных с использованием кода возврата

Процедура может возвращать целочисленное значение, называемое кодом возврата, чтобы указать состояние выполнения процедуры. Код возврата для процедуры указывается при помощи инструкции RETURN. Как и выходные параметры, при выполнении процедуры код возврата необходимо сохранить в переменной, чтобы использовать это значение в вызывающей программе.

**Пример.** Переменная @result типа данных *int* используется для хранения кода возврата из процедуры my\_proc:

```
DECLARE @result int;  
EXECUTE @result = my_proc;
```

**Пример:**

```
IF @ ИмяПродавца IS NULL  
    BEGIN  
        PRINT 'ERROR: Информация по продавцу не найдена.'  
        RETURN(1)  
    END
```

# Удаление хранимых процедур

```
DROP PROCEDURE <имя_процедуры>;
```

Пример:

```
USE AdventureWorks2012;
```

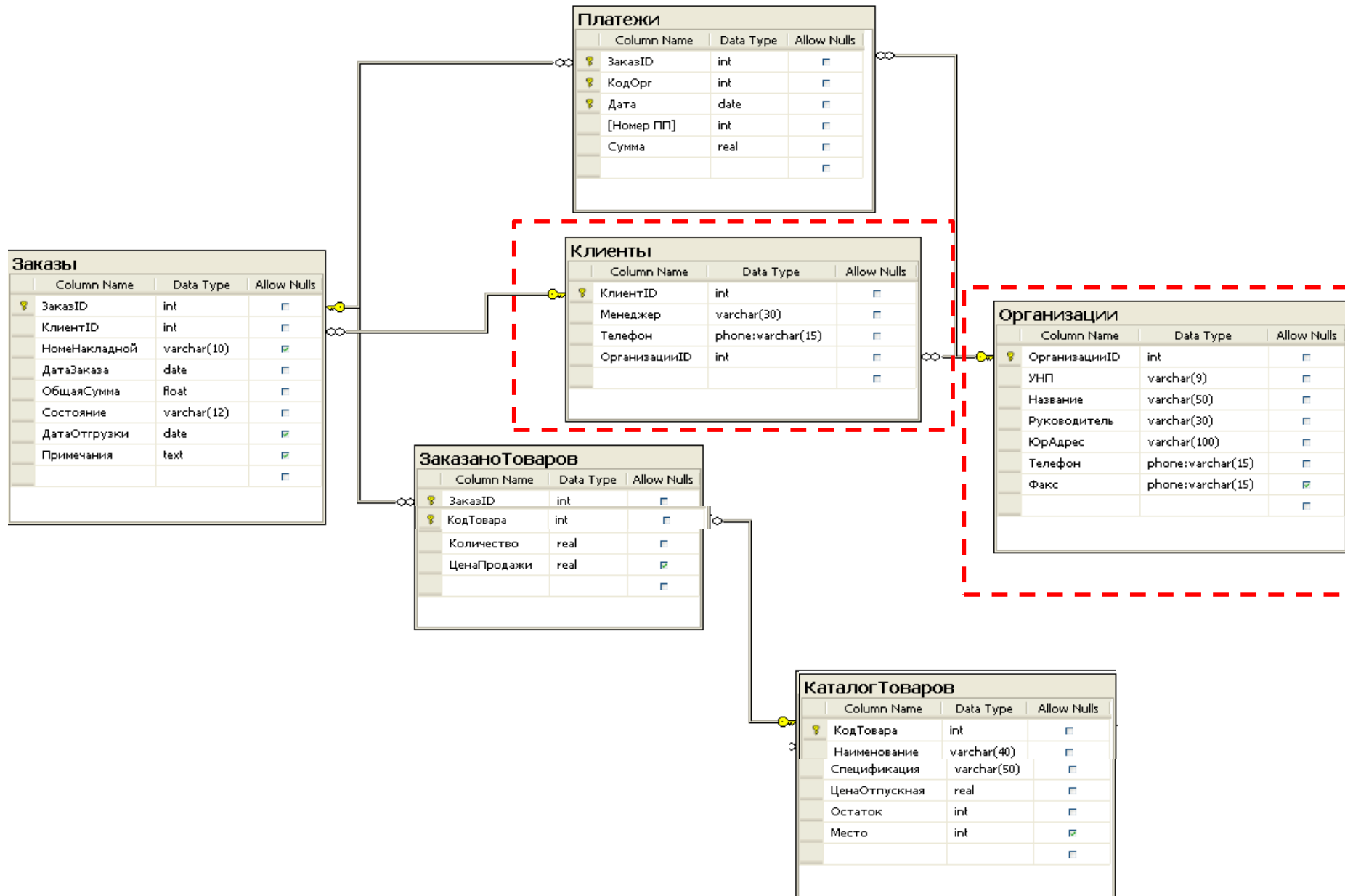
```
GO
```

```
IF OBJECT_ID('MyProc', 'P') IS NOT NULL
```

```
DROP PROCEDURE dbo.MyProc;
```

```
GO
```

# Схема БД «Заказы»

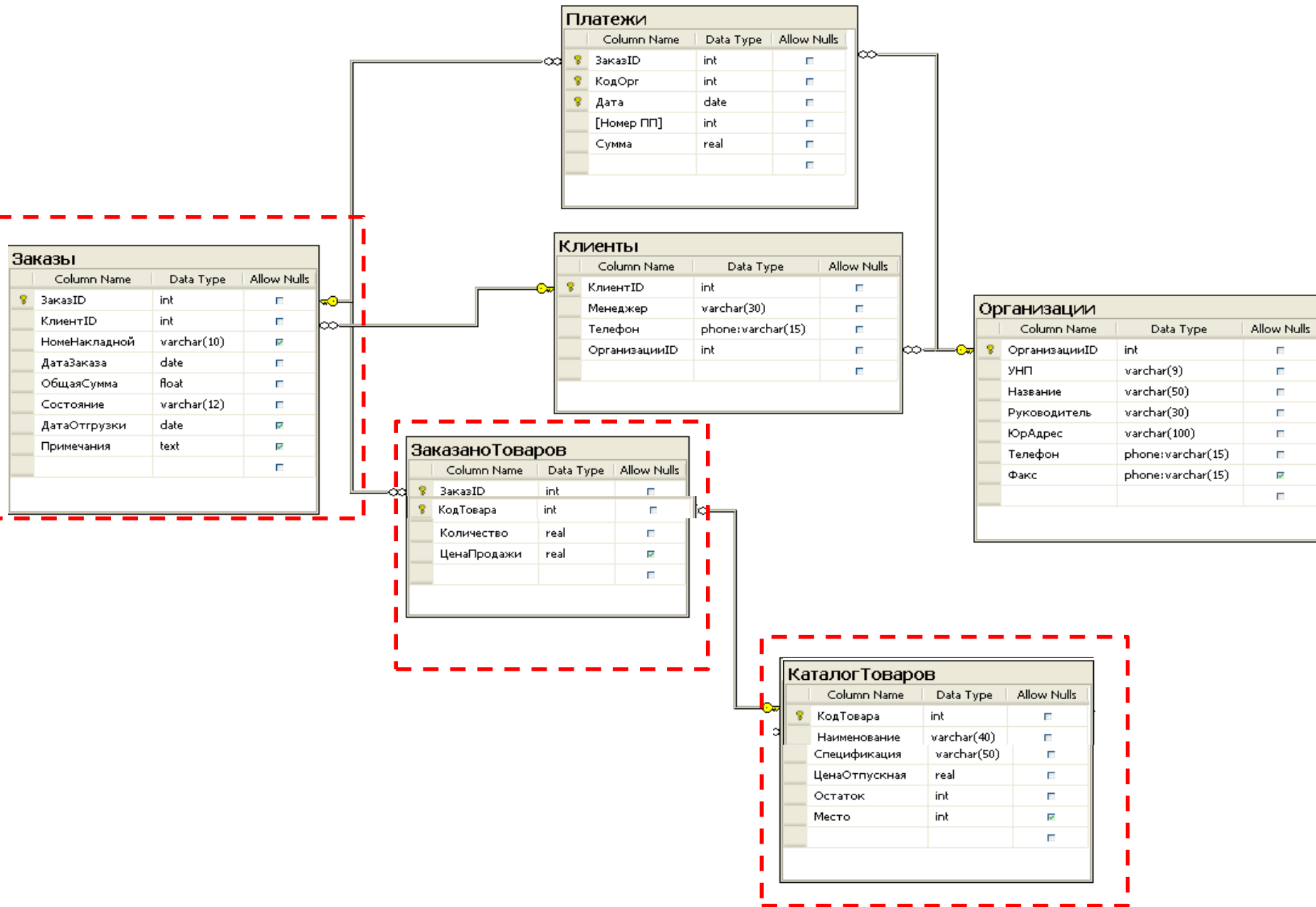




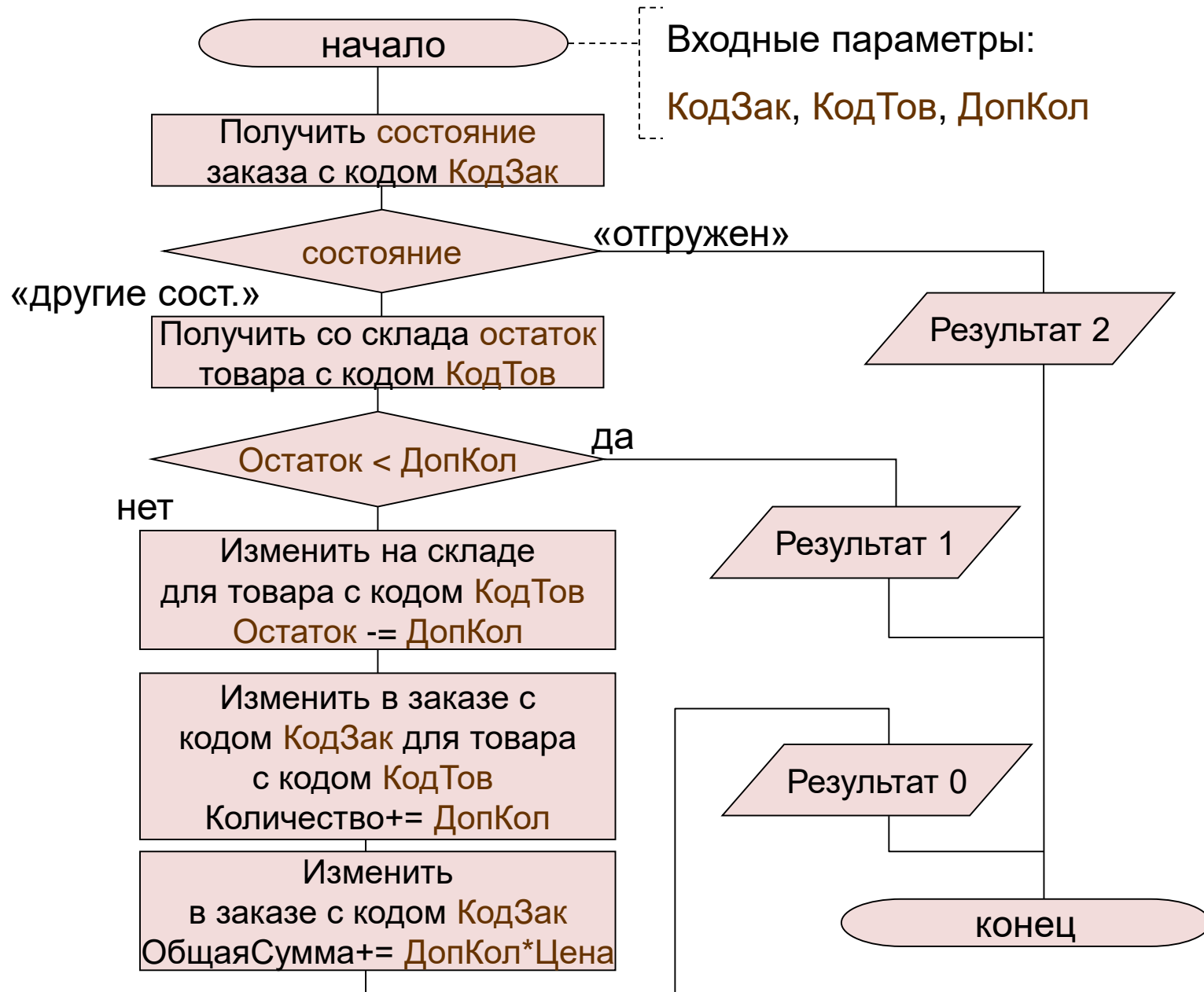
## ХП добавления нового клиента

```
CREATE PROCEDURE НовыйКлиент
@УНП varchar(9),
@НазваниеОрганизации varchar(50),
@Руководитель varchar(30),
@ЮрАдрес varchar(100),
@Телефон phone,
@Факс phone = NULL,
@Менеджер varchar(30),
@МТелефон phone = NULL
AS
begin
    DECLARE @ОрганизацияID int
    select @ОрганизацияID=ОрганизацииID from Организации
    where Название=@НазваниеОрганизации
    if @ОрганизацияID IS NULL
    begin
        INSERT INTO Организации
        (УНП, Название, Руководитель, ЮрАдрес, Телефон, Факс)
        VALUES (@УНП, @Наименование, @Руководитель, @ЮрАдрес, @Телефон,
        @Факс)
        SET @ОрганизацияID = IDENT_CURRENT('ОрганизацииID')
    end
    INSERT INTO Клиенты (Менеджер, Телефон, ОрганизацияID)
    VALUES (@Менеджер, @МТелефон, @ОрганизацияID)
end
```

# Схема БД «Заказы»



# Пример функции бизнес-логики



# ХП добавления товара в заказ

```
CREATE PROC ДобавитьЗаказКолТовар
@КодЗак INT, @КодТов INT, @ДопКол INT
AS
BEGIN
    DECLARE @Состояние VARCHAR(10), @Остаток INT, @Цена MONEY
    SELECT @Состояние = Состояние FROM Заказы WHERE ЗаказID = @КодЗак
    IF @Состояние IS NOT NULL AND @Состояние <> 'отгружен'
    BEGIN
        SELECT @Остаток = Остаток, @Цена= ЦенаОтпускная FROM КаталогТоваров
        WHERE КодТовара= @КодТов
        IF @Остаток >= @ДопКол
        BEGIN
            UPDATE КаталогТоваров SET Остаток = Остаток - @ДопКол
            WHERE КодТовара = @КодТов
            UPDATE ЗаказаноТоваров SET Количество = Количество + @ДопКол
            WHERE ЗаказID = @КодЗак AND КодТовара = @КодТов
            UPDATE Заказы SET ОбщаяСумма = ОбщаяСумма + @ДопКол* @Цена
            WHERE ЗаказID= @КодЗак
            RETURN 0
        END
    ELSE
        RETURN 1 -- остаток<@ДопКол
    ELSE
        RETURN 2 -- если отгружен
END
```

## Курсоры. Основные определения

Курсор – это особый временный объект SQL, предназначенный для использования в программах и хранимых процедурах. С его помощью можно в цикле пройти по результирующему набору строк запроса, по отдельности считывая и обрабатывая каждую его строку.

### Типы курсоров

- **Однонаправленный** (указывается как FORWARD\_ONLY и READ\_ONLY и не поддерживает прокрутку назад)
- **Статический** (отображает результирующий набор точно в том виде, в котором он был при открытии курсора)
- **Keyset** (Членство и порядок строк в курсоре, управляемом набором ключей, являются фиксированными при открытии курсора. Такие курсоры управляются с помощью набора уникальных идентификаторов)
- **Динамический** (отражают все изменения строк в результирующем наборе при прокрутке курсора)

## Объявление курсора

При помощи инструкции DECLARE можно объявлять переменные курсоров для использования в других инструкциях.

```
DECLARE cursor_name [ INSENSITIVE ] [ SCROLL ] CURSOR  
    FOR select_statement  
    [ FOR { READ ONLY | UPDATE [ OF column_name [ ,...n ] ] } ]  
[;]
```

Transact-SQL Extended Syntax

```
DECLARE cursor_name CURSOR [ LOCAL | GLOBAL ]  
    [ FORWARD_ONLY | SCROLL ]  
    [ STATIC | KEYSET | DYNAMIC | FAST_FORWARD ]  
    [ READ_ONLY | SCROLL_LOCKS | OPTIMISTIC ]  
    [ TYPE_WARNING ]  
    FOR select_statement  
    [ FOR UPDATE [ OF column_name [ ,...n ] ] ]  
[;]
```

## Открытие курсора

Открывает серверный курсор языка Transact-SQL и заполняет его с помощью инструкции языка Transact-SQL, определенной в инструкции DECLARE CURSOR или SET *cursor\_variable*.

```
OPEN { { [ GLOBAL ] cursor_name } | cursor_variable_name }
```

## Извлечение строки из курсора

Извлекает определенную строку из серверного курсора Transact-SQL.

FETCH

[ [ NEXT | PRIOR | FIRST | LAST  
| ABSOLUTE { n | @nvar }  
| RELATIVE { n | @nvar }

]

FROM

]

{ { [ GLOBAL ] cursor\_name } | @cursor\_variable\_name }

[ INTO @variable\_name [ ,...n ] ]



## Удаление курсора

Закрывает открытый курсор, высвобождая текущий результирующий набор и снимая блокировки курсоров для строк, на которых установлен курсор.

**CLOSE { { [ GLOBAL ] cursor\_name } | cursor\_variable\_name }**

Удаляет ссылку курсора. Когда удаляется последняя ссылка курсора, Microsoft SQL Server освобождает структуры данных, составляющие курсор.

**DEALLOCATE { { [ GLOBAL ] cursor\_name } | @cursor\_variable\_name }**

## Пример использования курсора


```
DECLARE @LastName VARCHAR(50), @FirstName VARCHAR(50);
DECLARE my_cur CURSOR FOR SELECT LastName, FirstName FROM Person

OPEN my_cur

FETCH NEXT FROM my_cur INTO @LastName, @FirstName

WHILE @@FETCH_STATUS = 0
BEGIN
exec dbo.my_proc @LastName, @FirstName
PRINT 'Contact Name: ' + @FirstName + ' ' + @LastName
FETCH NEXT FROM my_cur INTO @LastName, @FirstName
END

CLOSE my_cur
DEALLOCATE my_cur
```



Любые действия с переменными, в которые считаны данные из курсора

## Задание

1. Создать хранимую процедуру, в которую в качестве значения параметра передаётся фамилия\_студента. В окно сообщений вывести информацию о хобби данного студента.
2. Создать хранимую процедуру, которой в качестве значения параметра передаётся название\_группы и название\_дисциплины. Сформировать ведомость сдачи экзаменов по указанной дисциплине у указанной группы. Информацию записать в итоговую физическую/виртуальную таблицу Экзаменационная\_ведомость.

*Вид таблицы Экзаменационная\_ведомость :*

КодЭк зВедо м	Название Группы	НазваниеДисциплины	Дата	ФИО	Оценка
1	ИАБ-221	Высшая математика	12.01.2024	Артемов П.Р.	3
2	ИАБ-221	Высшая математика	12.01.2024	Баев А.Г.	2
3	ИАБ-221	Высшая математика	18.02.2024	Баев А.Г.	3
4	ИАБ-221	Высшая математика	12.01.2024	Гусев П.Т.	5
5	ИАБ-221	Высшая математика	12.01.2024	Краснова О.В.	5
6	ИАБ-221	Высшая математика	12.01.2024	Шпаков И.Н.	Null