Введение в программирование SQL Server

Хранимые процедуры

Хранимые процедуры

- **Хранимые процедуры** - это объекты базы данных, которые представляют собой небольшие программы, манипулирующие данными и выполняемые на сервере.
- **Хранимая процедура** – это набор инструкций T-SQL, выполняемый как единое целое.
- **Хранимая процедура** кроме команд языка **SQL**, могут использовать немногочисленные управляющие команды.

Хранимые процедуры

Создание

CREATE PROC[EDURE] имя_процедуры [параметры]

AS

операторы процедуры

Вызов

EXEC[UTE] имя_процедуры [список_формальных_параметров]

Переменные

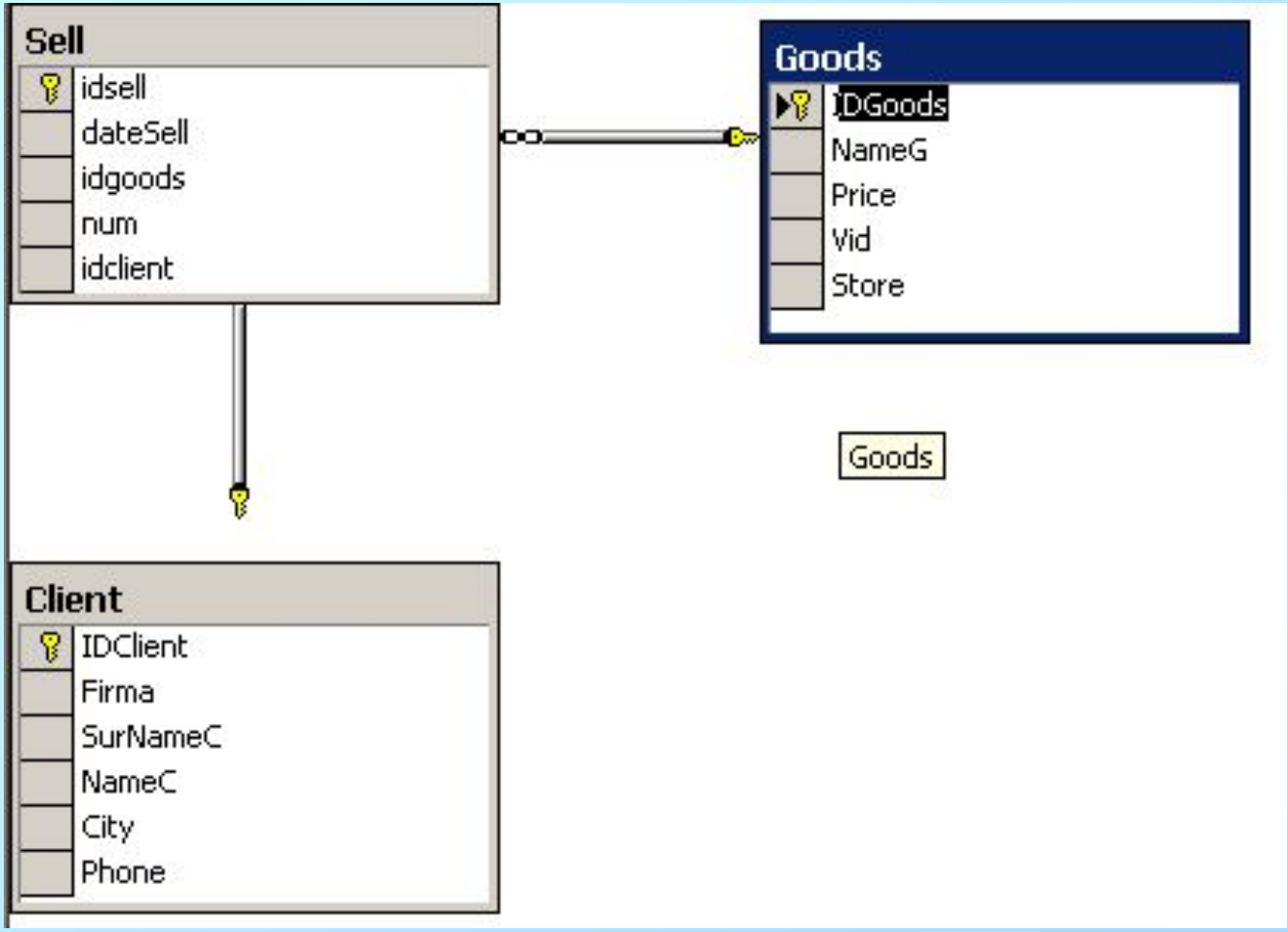
- Локальные переменные (начинаются с символа @)
- Глобальные переменные (начинаются с символов @@)
- Объявление переменных **DECLARE имя_переменной тип_переменной**

Операторы

- Блок операторов
  BEGIN … END

Оператор присвоения
  SELECT переменная=значение
  SET переменная=значение

- Условный оператор
  IF условие Оператор1 [ELSE Оператор2]
- Цикл
  WHILE условие Оператор
- Выбор
  CASE выражение
  WHEN вариант1 THEN выражение1
  WHEN вариант2 THEN выражение2 …
  ELSE выражениеN
  END


Sell
idsell
dateSell
idgoods
num
idclient
Goods
IDGoods
NameG
Price
Vid
Store
Goods
Client
IDClient
Firma
SurNameC
NameC
City
Phone

Хранимые процедуры

- Создание процедуры, использующий входной параметр

```
CREATE PROCEDURE show_goods @vid nVarchar(50)
AS
SELECT NameG, price, store
        FROM goods
        WHERE vid=@vid
```

- Вызов

```
EXEC show_goods 'конфеты'
```

Использование условия

- Проверка существования клиента. Если фирма существует вывести подробную информацию о фирме.

```
alter proc ex_begin_end
as
if exists (select * from client  where firma='Норд')
begin
        print 'customer found'
        select namec, surnamec, city, phone
        from client
        where firma='Норд'
        return
end
else
begin
        print 'customer not found'
        return
end
```

- Проверка существования клиента. Если фирма существует вывести подробную информацию о фирме. Наименование фирмы вводится с клавиатуры. Если фирмы не существует выводится соответствующее сообщение

```
create proc ex_begin_end3 @firma nvarchar(255)
as

if exists (select * from client  where firma= + @firma
begin

        select namec, surnamec, city, phone
        from client
        where firma= @firma
        return
end
else
begin
        print 'customer not found'
        return
end
```

Использование цикла

- Обновляется цена до тех пор пока средняя цена <80. Максимальная цена не должна превышать 200. В цикле выводится максимальная цена

```
create proc ex_while
as

while (select avg(price)  from goods) < 80
begin
        update goods
        set price=price+1
        select max(price) from goods
        if (select max(price) from goods) > 200
                break
        else
                continue
end
        print 'updated'
```

- Вывод содержимого, указанной таблицы или представления

```
create procedure ex_table1
@t varchar(50)
as
declare @text varchar(50)
set @text='select * from ' + @t
exec(@text)
go

exec ex_table1 'client'
```

Примеры хранимых процедур

- Вывести данные о продажах клиентов из определенного города или продажи определенного товара. Город и товар вводится с клавиатуры.

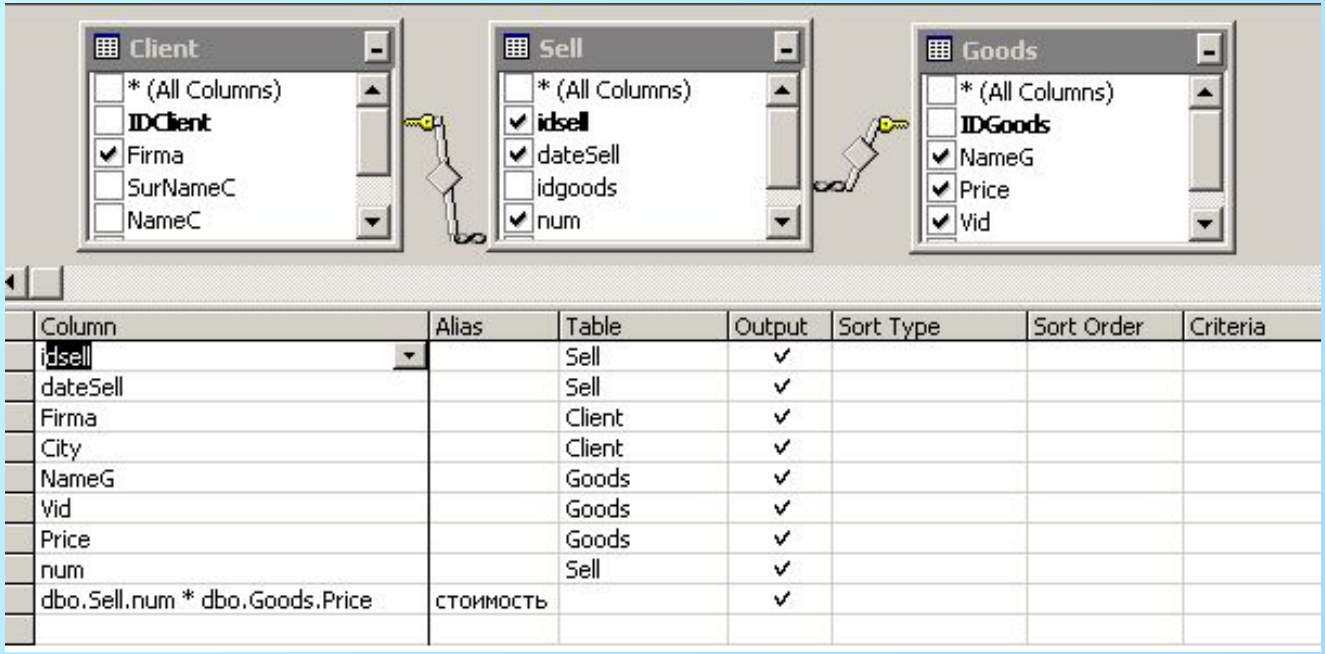
Способ 1.

Процедура создается в окне редактора запросов

```
CREATE proc ex6
@city nvarchar(50),
@vid  nvarchar(50)
AS
SELECT     Sell.idsell, Sell.dateSell, Client.Firma, Client.City, Goods.NameG,
Goods.Price, Goods.Vid, Sell.num
FROM         Client INNER JOIN
                      Sell ON Client.IDClient = Sell.idclient INNER JOIN
                      Goods ON Sell.idgoods = Goods.IDGoods
WHERE     (Client.City = @city) OR
                      (Goods.Vid = @city)
```

Способ 2

1. Создайте представление в Конструкторе



Column	Alias	Table	Output	Sort Type	Sort Order	Criteria
idsell		Sell	✓			
dateSell		Sell	✓			
Firma		Client	✓			
City		Client	✓			
NameG		Goods	✓			
Vid		Goods	✓			
Price		Goods	✓			
num		Sell	✓			
dbo.Sell.num * dbo.Goods.Price	стоимость		✓			

2. Сохраните представление, как V_all

Способ 2

3. Создайте процедуру в окне редактора запросов

```
create procedure ex_Or1
@city nvarchar(50),
@vid  nvarchar(50)
as
select * from V_All
where vid=@vid or city=@city


exec ex_or1 'Спб','конфеты'
```

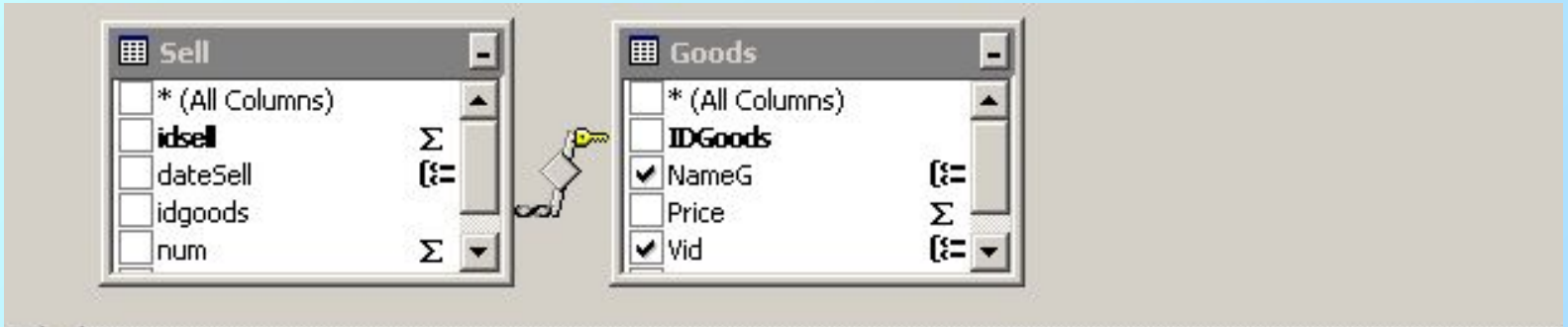
4. Сравните процедуры

Способ 1

- Итоги продажи товара за год

```
create procedure my5
@y int
as
SELECT      YEAR(Sell.dateSell) AS год, Goods.NameG, Goods.Vid,
 SUM(Sell.num) AS [кол продаж], COUNT(Sell.idsell) AS продано,
                        SUM(Sell.num * Goods.Price) AS сумма
FROM          Sell INNER JOIN
                        Goods ON Sell.idgoods = Goods.IDGoods
GROUP BY YEAR(Sell.dateSell), Goods.NameG, Goods.Vid
HAVING       (YEAR(Sell.dateSell) = @y)
```

Способ 2



Column	Alias	Table	Output	Sort Type	Sort Order	Group By	Crit
YEAR(dbo.Sell.dateSell)	год		✓			Group By	
NameG		Goods	✓			Group By	
Vid		Goods	✓			Group By	
num	[кол продаж]	Sell	✓			Sum	
idsell	продано	Sell	✓			Count	
dbo.Sell.num * dbo.Goods	сумма		✓			Sum	

```
create procedure ex_итоги
@y int

as
select * from итоги
where год=@y


exec ex_итоги 2005
```

Распродажа1. Уменьшить стоимость непродаваемых товаров в десять раз

1. Создать представление, отбирающее непродаваемые товары

```
create view V_not_Sell
as
select * from goods where
idgoods not in (select idgoods from sell)
```

2. Создать хранимую процедуру, на основе представления

```
create proc sale_1
as
update V_not_sell
Set price=price/10
```

3. Выполнить процедуру

Распродажа2. Уменьшить стоимость непродаваемых товаров на указанное количество процентов.

1. Создать представление, отбирающее непродаваемые товары

```
create view V_not_Sell
as
select * from goods where
idgoods not in (select idgoods from sell)
```

2. Создать хранимую процедуру, на основе представления

```
create proc sale_2
@percent int
as
update V_not_sell
SET price = price*(100-@percent)/100
select * from V_not_sell
go
```

3. Выполнить процедуру

```
exec sale_2 10
```

Процедура на добавление

```
create proc ins_good
@nameg nVarchar(50),
@price money,
@vid nVarchar(50),
@store int
as
/*set noncount on*/
declare @newgood nVarchar(50)
insert into goods(nameg,price,vid,store)
values(@nameg,@price,@vid,@store)
Set @newgood=@nameg
select @newgood as 'новый товар'

exec ins_good2 'графские развалины', 200, 'торт', 5
```

- Между

```
create procedure my6
@y1 int, @y2 int
as
SELECT      YEAR(Sell.dateSell) AS год, Goods.NameG, Goods.Vid,
 SUM(Sell.num) AS [кол продаж], COUNT(Sell.idsell) AS продано,
                        SUM(Sell.num * Goods.Price) AS сумма
FROM          Sell INNER JOIN
                        Goods ON Sell.idgoods = Goods.IDGoods
GROUP BY YEAR(Sell.dateSell), Goods.NameG, Goods.Vid
HAVING       (YEAR(Sell.dateSell) between @y1 and @y2)
```

```
create procedure my9
@y int, @m1 int, @m2 int
as
SELECT    YEAR(Sell.dateSell) AS год, month (Sell.dateSell) AS год,
          Goods.NameG, Goods.Vid,
          SUM(Sell.num) AS [кол продаж], COUNT(Sell.idsell) AS продано,
          SUM(Sell.num * Goods.Price) AS сумма
FROM  Sell INNER JOIN  Goods ON Sell.idgoods = Goods.IDGoods
GROUP BY YEAR(Sell.dateSell), month(Sell.dateSell), Goods.NameG, Goods.Vid
HAVING       (month(Sell.dateSell) between @m1 and @m2  and YEAR(Sell.dateSell)=@y)
```

- Триггер - это специальный тип хранимой процедуры, которая автоматически выполняется при каждой попытке изменить защищаемые его данные.
- Триггеры не имеют параметров и не выполняются явно. Это значит, что триггер запускается только при попытке изменения данных.

- По умолчанию все триггеры (INSERT, DELETE и UPDATE) срабатывают после выполнения оператора изменения данных. Эти триггеры, называющиеся триггерами AFTER (после)
- Кроме того в SQL Server 2000 используются триггеры INSTEAD OF (вместо), которые выполняются вместо оператора предполагаемого изменения данных.

CREATE TRIGGER имя_триггера ON имя_таблицы

FOR INSERT | UPDATE | DELETE AS Код_триггера

- При добавлении строки в таблицу ее копия помещается во временную таблицу с именем **Inserted**, при удалении - с именем **Deleted**.
- При обновлении старая версия строки помещается во временную таблицу с именем **Deleted**, новая - с именем **Inserted**.

```
CREATE TRIGGER ins_goods
    ON sell FOR INSERT
    AS
    DECLARE @IDGoods int, @kolvo int
-- Выбираем код товара и количество товара
SELECT @kolvo=I.num, @IDGoods=I.IDGoods
FROM Inserted I
INNER JOIN Goods  T ON I.IDGoods = T.IDGoods
UPDATE Goods
Set store=store - @kolvo
Where IDGoods=@IDGoods
```

```
CREATE TRIGGER ins_good
    ON number FOR INSERT
    AS
    DECLARE @codG int, @kolvo int
-- Выбираем код товара и количество товара
SELECT @kolvo=I.number, @codG=I.codG
FROM Inserted I
INNER JOIN Goods  T ON I.codG = T.codG
UPDATE Goods
Set store=store - @kolvo
Where codG=@codG
```

```
CREATE TRIGGER tr_good
ON Goods
FOR UPDATE
AS
DECLARE @Price MONEY, @CodG INT
-- Проверка того, что учетная запись изменилась
IF UPDATE (price)
BEGIN
-- Выбираем старую цену для товара, у которого она изменилась
     SELECT @CodG=D.CodG, @Price=D.Price
FROM DELETED D INNER JOIN Goods T ON D.CodG=T.CodG
-- Вставка записи в таблицу история цен (t_history)
INSERT INTO T_History (CodG, Date_price, Price)
VALUES (@CodG, GETDATE(),@Price)
END
```