**Тема 29. Хранимые процедуры**

1. **Необходимость применения хранимых процедур**

Если необходимо выполнить несколько инструкций для сложного запроса.

Допустим,

1. выяснить, есть ли товары на складе;
2. если они есть, зарезервировать их и количество, доступное другим пользователям на заказ;
3. запросить отсутствующие товары;
4. сообщить клиенту, какие товары будут отгружены, а какие запрашиваются под заказ.

Чтобы не писать по каждой операции отдельную инструкцию и выполнять ее каждый раз по ситуации необходимо создать **хранимую процедуру** (ХП)(набор из нескольких инструкций, сохраненный для последующего выполнения). Не поддерживаются в Access и SQLite.

Применяют ХП:

1. для упрощения запросов;
2. для обеспечения непротиворечивости данных за счет выполнения единого кода (повышает целостность данных);
3. при обновлении данных изменяется только код хранимой процедуры (повышает безопасность);
4. СУБД обрабатывает уже скомпилированные процедуры (повышение производительности);
5. хранимые процедуры можно применять для написания более гибкого и мощного кода.

Достоинства: простота, безопасность, производительность.

Проблемы:

1. синтаксис сильно зависит от СУБД. Вызовы хранимых процедур (имена и способы передачи аргументов могут быть переносимыми).
2. сложные в написании, поэтому многие администраторы БД ограничивают права на создание ХП в качестве меры безопасности.

Если не получается написать, используйте готовые.

1. **Выполнение хранимых процедур**

Команда EXECUTE имя процедуры (аргументы).

EXECUTE AddNewProduct (‘JTS01’,’Stuffed Eiffel Tower’, 6.49,’Plush staffed toy with the text La Tour Eiffel in red, white and blue’)

Добавляем новый товар в таблицу, у функции четыре аргумента: идентификатор поставщика, название товара, цена, описание.

ХП выполняет следующие действия:

1. подтверждает правильность передаваемых данных всем четырем аргументам;
2. генерирует у них идентификатор (первичный ключ);
3. добавляет данные о новом товаре в таблицу Products с первичным ключом в соответствующие столбцы.

Могут быть и другие варианты выполнения: опции по умолчанию, нестандартные параметры, входные параметры обновления переменной, данные, извлекаемые Select, возвращаемые коды.

1. **Создание хранимых процедур**

Создадим ХП, которая подсчитывает число клиентов, имеющих адрес почты.

CREATE PROCEDURE MailListCount AS

DECLARE @cnt INT

SELECT @cnt= COUNT (\*)

FROM Customers

WHERE NOT Cust\_email IS NULL;

RETURN @cnt;

Эта хранимая процедура не принимает аргументов.

DECLARE объявляет локальную переменную @cnt.

Затем она используется в инструкции SELECT, принимая значение, возвращаемое функцией COUNT (\*).

RETURN передает результат подсчета в вызываемое приложение.

Для запуска примера

DECLARE @ReturnValue INTEGER

EXECUTE

@ReturnValue=MailListCount

SELECT @ReturnValue;

Например,

CREATE PROCEDURE NewOrder

@Cust\_id CHAR (10) AS

--объявление переменной для номера заказа

DECLARE @Order\_num INTEGER

--получение текущего наибольшего номера заказа

SELECT @Order\_num=MAX (Order\_num)

FROM Orders

--определение следующего номера заказа

SELECT @ Order\_num= @Order\_num+1

--добавление нового заказа

INSERT INTO Orders (Order\_num, Order\_date, Cust\_id)

VALUES (@Order\_num, GetDATE (), @Cust\_id)

--возвращение номера заказа

RETURN @ Order\_num;

*--* Это универсальный синтаксис комментария. *ХП рекомендуется комментировать!*

Хранимая процедура создает новый заказ в таблице Orders и принимает один аргумент: классификатор клиента, сделавшего заказ. Номер и дата заказа генерируются автоматически в самой хранимой процедуре. Вначале объявляется локальная переменная для хранения номера заказа, затем запрашивается текущий максимальный номер заказа, увеличивающийся на один с помощью SELECT.

Затем с помощью INSERT добавляется новый заказ с использованием сгенерированного номера заказа, текущей системной даты, которая определяется с помощью RETURN @Order\_num.

В SQL Server можно реализовать эту хранимую процедуру и таким образом:

CREATE PROCEDURE New\_Order @cust\_id CHAR (10)

AS

--Добавление нового заказа

INSERT INTO Orders (Cust\_id)

VALUES (@Cust\_id)

--Возвращение номера заказа

SELECT Order\_num=@@IDENTITY;

Данная хранимая процедура создает новый заказ в таблице Заказы (Orders). СУБД сама генерирует №заказа (поле автогенерации). Процедуре передается только id клиента. № и дата\_заказа не указываются вообще, так как СУБД использует значение по умолчанию, для даты и № оно генерируется автоматически.

Какой же id был сгенерирован? В SQL Server для этого есть глобальная переменная @@IDENTTY, возвращаемая в вызывающее приложение с помощью SELECT.

Выходные параметры позволяют возвратить из процедуры некоторый результат. Выходные параметры определяются с помощью ключевого слова OUTPUT. Например, определим еще одну процедуру:

USE productsdb;

GO

CREATE PROCEDURE GetPriceStats

@minPrice MONEY OUTPUT,

@maxPrice MONEY OUTPUT

AS

SELECT @minPrice = MIN(Price), @maxPrice = MAX(Price)

FROM Products

При вызове процедуры для выходных параметров передаются переменные с ключевым словом OUTPUT:

USE productsdb;

DECLARE @minPrice MONEY, @maxPrice MONEY

EXEC GetPriceStats @minPrice OUTPUT, @maxPrice OUTPUT

PRINT 'Минимальная цена ' + CONVERT(VARCHAR, @minPrice)

PRINT 'Максимальная цена ' + CONVERT(VARCHAR, @maxPrice)

Также можно сочетать входные и выходные параметры. Например, определим процедуру, которая добавляет новую строку в таблицу и возвращает ее id:

USE productsdb;

GO

CREATE PROCEDURE CreateProduct

@name NVARCHAR(20),

@manufacturer NVARCHAR(20),

@count INT,

@price MONEY,

@id INT OUTPUT

AS

INSERT INTO Products(ProductName, Manufacturer, ProductCount, Price)

VALUES(@name, @manufacturer, @count, @price)

SET @id = @@IDENTITY

С помощью глобальной переменной @@IDENTITY можно получить идентификатор добавленной записи.

При вызове этой процедуры ей также по позиции передаются все входные и выходные параметры:

USE productsdb;

DECLARE @id INT

EXEC CreateProduct 'LG V30', 'LG', 3, 28000, @id OUTPUT

PRINT @id

Возвращение значения

Кроме передачи результата выполнения через выходные параметры хранимая процедура также может возвращать какое-либо значение типа INT с помощью оператора RETURN. Хотя данная возможность во многом нивелирована использованием выходных параметров, через которые можно возвращать результат, тем не менее, если надо возвратить из процедуры одно значение, то вполне можно использовать оператор RETURN.

Например, возвратим среднюю цену на товары:

USE productsdb;

GO

CREATE PROCEDURE GetAvgPrice AS

DECLARE @avgPrice MONEY

SELECT @avgPrice = AVG(Price)

FROM Products

RETURN @avgPrice;

После оператора RETURN указывается возвращаемое значение. В данном случае это значение переменной @avgPrice.

Вызовем данную процедуру:

USE productsdb;

DECLARE @result MONEY

EXEC @result = GetAvgPrice

PRINT @result

Для получения результата процедуры ее значение сохраняется в переменную (в данном случае в переменную @result):

Стоит отметить, что RETURN возвращает только целочисленные значения.