**Лабораторная работа 8 – Макеева Настя ПИН-31**

**Контрольные вопросы**

1. Что представляет собой хранимая процедура?

Хранимая процедура представляет собой оформленный особым образом пакет, который хранится в базе данных.

1. Как создать хранимую процедуру?

Для создания хранимой процедуры применяется команда CREATE PROCEDURE или CREATE PROC.

1. Как выполнить хранимую процедуру?

По умолчанию выполнить хранимую процедуру может только ее владелец, которым является владелец БД, и создатель хранимой процедуры. Однако владелец хранимой процедуры может делегировать права на ее запуск другим пользователям. Имя хранимой процедуры является идентификатором в языке программирования, на котором она пишется, и должно удовлетворять всем требованиям, которые предъявляются к идентификаторам в данном языке.

В MS SQL Server хранимая процедура создается оператором:

CREATE PROC[EDURE] <имя\_процедуры> [;<версия>]

[{@параметр1 тип\_данных}

[VARYING] [= <значение\_по\_умолчанию>] [ OUTPUT ]]

[**,**.параметр N ...]

[ WITH

{ RECOMPILE

| ENCRYPTION

| RECOMPILE**,**ENCRYPTION }]

[ FOR REPLICATION ]

AS

Тело процедуры

1. Как передать данные в хранимую процедуру?

Для передачи входных данных в создаваемой хранимой процедуре имена параметров должны начинаться с символа @. В одной хранимой процедуре можно задать множество параметров, разделенных запятыми. В теле процедуры не должны применяться локальные переменные, чьи имена совпадают с именами параметров этой процедуры.

1. Как получить результаты из хранимой процедуры?

Существует три способа возврата данных из процедуры в вызывающую программу: результирующие наборы, параметры вывода и коды возврата.

Если включить инструкцию SELECT в тело хранимой процедуры (но не SELECT... INTO или INSERT... SELECT), 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. Такое поведение также применяется к вложенным пакетам TSQL, вложенным хранимым процедурам и пакетам TSQL верхнего уровня.

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.

Transact-SQL в процедурах только выходные (OUTPUT) параметры могут иметь тип данных **cursor**. Если тип данных **cursor** указан для параметра, то как ключевое слово VARYING, так и ключевое слов OUTPUT должны быть указаны для этого параметра в определении процедуры. Параметр может быть указан только как выходной, однако если в объявлении параметра указано ключевое слово VARYING, типом данных должен быть **cursor**, при этом также следует указать ключевое слово OUTPUT.

1. Каково назначение команды Return?

Служит для безусловного выхода из запроса или процедуры. Инструкция RETURN выполняется немедленно и полностью и может использоваться в любой точке для выхода из процедуры, пакета или блока инструкций. Инструкции, следующие после RETURN, не выполняются. При использовании в хранимой процедуре инструкция RETURN не может возвращать значение NULL. Если процедура пытается вернуть значение NULL (например, с помощью инструкции RETURN @status, если @status равен NULL), формируется предупредительное сообщение и возвращается значение 0. Возвращаемое значение состояния может быть включено в последующие инструкции Transact-SQL пакета или процедуры, выполняющей текущую процедуру, но должно вводиться в следующем формате: EXECUTE @return\_status = <procedure\_name>.

1. Как получить возвращаемое значение из хранимой процедуры?

Для получения результата процедуры ее значение сохраняется в переменную

1. Что представляет собой триггер?

Триггер — это хранимая процедура особого типа, вызываемая на выполнение в ответ на определенные события.

1. Перечислите отличия триггера от хранимой процедуры.

В отличие от хранимых процедур, при использовании которых требуется их явный вызов на выполнение, триггеры вызываются на выполнение автоматически при обнаружении события (или событий), связанного с выполнением операций над таблицей, за которой закреплен триггер. Триггеры не могут быть вызваны явно; единственный способ обеспечения вызова состоит в выполнении требуемого действия над таблицей, за которой закреплен триггер.

Различия между хранимыми процедурами и триггерами не ограничиваются тем, что невозможен явный вызов триггера. Хранимые процедуры не только вызываются явно, но и отличаются двумя особенностями, которыми не обладают триггеры: (1) принимают и передают параметры и (2) возвращают коды завершения.

Триггеры не имеют параметров, но в коде триггера может использоваться определенный механизм, позволяющий выяснить, на какие строки должно распространяться действие триггера. Еще одна особенность триггеров состоит в том, что в них допускается применение команды RETURN, но они не позволяют возвращать конкретные значения кода завершения (поскольку явный вызов триггера не предусмотрен, то не определена точка вызова, в которую можно было бы возвратить код завершения).

1. Какие классы триггеров предусмотрены?

Триггеры подразделяются на два основных класса: триггеры языка определения данных (Data Definition Language — DDL) и триггеры языка манипулирования данными (Data Manipulation Language — DML).

1. Какие типы триггеров DML существуют?

Триггеры INSERT.

Триггеры DELETE.

Триггеры UPDATE.

Триггеры, создаваемые с учетом одновременного возникновения и совпадения событий.

1. Какие подклассы триггеров DML существуют и каковы их особенности?

INSTEAD OF (триггер замены операции) и AFTER (триггер, выполняющийся сразу после операции), отличающиеся своим назначением, моментом выполнения и производимым эффектом

Сравнение характеристик подклассов триггеров DML

|  |  |  |
| --- | --- | --- |
| Характеристика | Триггер INSTEAD OF | Триггер AFTER |
| Эффект оператора DML | Автоматически откатывается | Выполняется, если триггер сам не откатит транзакцию |
| Момент выполне-ния триггера | Перед проверкой ограниче-ний первичного и внешнего ключей | После выполнения транзакции,  но перед ее подтверждением |
| Количество воз-можных событий таблицы | Одно | Несколько |
| Возможность при-менения  к пред-ставлениям | Есть | Отсутствует |
| Рекурсивное срабатывание | Отсутствует | Зависит от параметров настройки СУБД |

1. Как временно отключить срабатывание триггера, а затем снова включить?

триггер можно  временно отключить оператором ALTER TABLE с параметром DISABLE TRIGGER:

ALTER  TABLE  *имя\_таблицы* DISABLE TRIGGER  *имя\_триггера*

 Для включения триггера применяется тот же оператор, но с параметром ENABLE  TRIGGER:

ALTER  TABLE   *имя\_таблицы* ENABLE TRIGGER  *имя\_триггера*

1. Что содержат виртуальные таблицы Inserted и Deleted?
2. Содержимое виртуальных таблиц Inserted и Deleted

|  |  |  |
| --- | --- | --- |
| Оператор DML | Виртуальная таблица | |
| Inserted | Deleted |
| insert | Вставленные строки | Пустая |
| update | Строки базы данных после обновления | Строки базы данных до обновления |
| delete | Пустая | Строки, подлежащие удалению |

1. На Терминале 4100 запустить утилиту SSMS.
2. Дополнить выбранную главную таблицу столбцом NORR с нулевым значением по умолчанию и заполнить его значениями, равными числу подчиненных строк.
3. Создать хранимые процедуры и убедиться в их наличии в базе данных, выбрав ветви Programming и Stored Procedures в окне Обозревателя объектов.
4. Создать триггеры и убедиться в их наличии в базе данных, выбрав ветвь Triggers для таблиц базы данных в окне Обозревателя объектов.

CREATE TRIGGER trigDel ON dbo.Operation

AFTER DELETE

AS

BEGIN

UPDATE dbo.Music SET NORR=NORR - 1 WHERE dbo.Music.MusicName = (SELECT deleted.MusicName FROM deleted)

END

GO

CREATE TRIGGER trigMusic ON dbo.Operation

AFTER INSERT

AS

UPDATE dbo.Music SET NORR = NORR+1 WHERE dbo.Music.MusicName = (SELECT inserted.MusicName FROM inserted);

DELETE FROM dbo.Music WHERE dbo.Music.MusicName = (SELECT inserted.MusicName FROM inserted)AND dbo.Music.NORR=5;

CREATE TRIGGER trigDelAll ON dbo.Music

INSTEAD OF Delete

AS

DECLARE @x int;

SELECT @x=Deleted.MusicName from Deleted;

IF EXISTS (SELECT \* from dbo.Operation WHERE MusicName = @x)

DELETE FROM dbo.Operation WHERE MusicName = @x;

DELETE FROM dbo.Music WHERE MusicName = @x;