Лабораторная работа № 13

Язык SQL. Создание хранимых процедур

4 часа

Цель работы – научиться создавать хранимые процедуры в СУБД MS SQL Server Management Studio. Реализовать хранимые процедуры для вставки, удаления, изменения данных и специальные хранимые процедуры в базе данных разрабатываемой информационной системы.

Хранимые процедуры имеют очень много общего с обычными процедурами, широко используемыми в различных языках программирования. Определение хранимой процедуры заключается в исполнении следующей команды:

```
CREATE PROCEDURE] procedure_name [;number]
[ {@parameter data_type} [VARYING] [= default] [OUTPUT] ] [,...n]
[WITH {    RECOMPILE
|  ENCRYPTION
|  RECOMPILE.  ENCRYPTION }  ]
[FOR REPLICATION]
AS sql_statement [...n]
```

Синтаксис имеет следующее назначение:

procedure_name — имя создаваемой процедуры. Используя префиксы sp_, # и ##, можно определить создаваемую процедуру как системную или временную. При этом, однако, необходимо позаботиться и о размещении процедуры в соответствующей системной базе данных, поскольку команда CREATE PROCEDURE создает хранимую процедуру в *текущей* базе данных. Поэтому перед созданием процедуры необходимо выполнить команду USE, чтобы сделать требуемую базу данных текущей;

number — параметр определяет идентификационный номер хранимой процедуры, однозначно определяющий ее в группе процедур;

(@parameter — определяет имя параметра, который будет использоваться создаваемой хранимой процедурой для передачи входных или выходных данных. Параметры, определяемые при создании хранимой процедуры, являются локальными переменными, поэтому несколько хранимых процедур могут иметь абсолютно идентичные параметры;

к data_type — определяет, к какому типу данных должны относиться значения параметра описываемой процедуры. Для определения параметров можно использовать любые типы данных;

OUTPUT — использование этого ключевого слова определяет указанный параметр как выходной;

VARYING — ключевое слово, которое используется совместно с параметром, относящимся к типу данных cursor. Определяет, что в качестве выходного параметра будет представлено результирующее множество;

default — позволяет определить для параметра значение по умолчанию, которое хранимая процедура будет использовать в случае, если при ее вызове указанный параметр был опущен.

RECOMPILE — ключевое слово, предписывающее системе создавать план выполнения процедуры при каждом ее вызове;

FOR REPLICATION — процедура, определенная с использованием данного ключевого слова, предназначена исключительно для осуществления процесса репликации. Вы не можете сочетать это ключевое слово с ключевым словом WITH RECOMPILE;

ENCRYPTON — если при определении процедуры было использовано данное ключевое слово, то текст процедуры непосредственно перед записью в системную таблицу syscomments будет зашифрован. Вы можете прибегнуть к шифрованию, если необходимо скрыть от пользователя особенности реализации хранимой процедуры;

AS — ключевое слово, определяющее начало кода хранимой процедуры. После этого ключевого слова следуют команды Transact-SQL, которые и составляют непосредственно *тело* процедуры (sql statement). Здесь можно использовать любые команды, включая вызов других хранимых процедур, за исключением команд, начинающихся с ключевого слова CREATE.

Примеры

1. Список выдач книг за текущий день.

```
CREATE PROCEDURE CpicokVidach
AS
SELECT   /*Перечисляем поля, которые будут выведены в результате запроса */
Пользование_библиотекой2.Табельный_номер,
COUNT(Пользование_библиотекой2.Дата_выдачи)
FROM /*указываем имя таблицы  из которых выбираются записи*/
Пользование_библиотекой2
WHERE  /*задаем условие отбора*/
Пользование_библиотекой2.Дата_выдачи=(SELECT GETDATE())
GROUP BY /*производится группировка по указанному полю*/
Табельный_номер
```

*/*SELECT GETDATE() позволяет получить текущую дату (год, месяц, число)*

COUNT (<поле>) возвращает количество записей какого-либо поля/*

2. Количество экземпляров какой-либо книги.

```
CREATE PROCEDURE KolExzemplarov
/*Объявляем необходимые переменные*/
@ISBN varchar(20)
AS
/* Следующая конструкция проверяет, существуют ли записи в таблице «Книги» с заданным ISBN*/
IF not EXISTS (SELECT * FROM Книга WHERE ISBN = @ISBN)
RETURN 0 /*Вызывает конец процедуры KolExzemplarov */
SELECT Экземпляр.ISBN
INTO TEMP1 /*Сохраняет выбранные поля во временной таблице Temp1*/
FROM Экземпляр
WHERE ISBN = @ISBN
SELECT COUNT(ISBN) /*Count подсчитывает количество неповторяющихся записей поля ISBN*/
FROM TEMP1
```

3. Список книг, которыми пользовался какой-либо студент

```
CREATE PROCEDURE CpicokKnigCtudenta
@Chit_nom int     /*Объявляем необходимые переменные*/
AS SELECT Студенты.Имя, Студенты.Фамилия, Пользование_библиотекой2.Шифр, Книга.Автор, Книга.Название
FROM /*указываем имена таблиц,  из которых выбираются записи*/
Книга, Экземпляр ,Студенты , Пользование_библиотекой2
WHERE (Студенты.Читательский_номер = Пользование_библиотекой2.Читательский_номер) AND (Экземпляр.Шифр = Пользование_библиотекой2.Шифр) AND (Экземпляр.ISBN =Книга.ISBN)
/* AND позволяет задать в операторе WHERE несколько условий, которые должны выполняться одновременно*/
```

4. Удаление из таблицы «Студенты». Допустимо, если в таблице «Пользование библиотекой2» нет ссылающихся записей.

```
CREATE  PROCEDURE DeleteStudent
@Chit_nom int          /*Объявляем необходимые переменные*/
```

AS */*Проверяем, если ссылающиеся записи, если записей нет, разрешается удаление.*/*

IF not EXISTS (SELECT * FROM Пользование_библиотекой2 WHERE Читательский_номер=@Chit_nom)

DELETE */*Оператор удаления*/*

FROM Студенты */*Имя таблицы, откуда нужно удалить*/*

WHERE */*Условие удаления – удаляем строку, для которой значение поля Читательский_номер совпадает с нужным*/*

Читательский_номер=@Chit_nom

5. Вставка в таблицу «Заказы». Разрешена, если в таблицах «Книга» и «Преподаватели» есть записи, на которые будет ссылаться новая запись.

PROCEDURE NewZakaz

@Kolvo int, */*Объявляем необходимые переменные*/*

@data_zakaza datetime,

@Chit_nomer int,

@ISBN varchar(20)

AS */*Проверяем, есть ли запись в таблице «Заказы» с такими же значениями ключевых полей, как у новой записи*/*

IF EXISTS (SELECT * FROM Заказы WHERE ISBN = @ISBN AND Читательский_номер=@Chit_nomer)

RETURN 0 */*Если есть, завершаем выполнение процедуры*/*

IF EXISTS (SELECT * FROM Преподаватели WHERE Читательский_номер = @Chit_nomer)

*/*Проверили, есть ли в «Преподаватели» соответствующая запись*/*

IF EXISTS (SELECT * FROM Книга WHERE ISBN = @ISBN)

*/*Проверили, есть ли в «Книга» соответствующая запись*/*

INSERT INTO Заказы */*Указываем таблицу, куда вставляем запись*/*

VALUES (@Kolvo,@data_zakaza,@Chit_nomer, @ISBN) /**Указываем какие значения*/*

6. Обновление таблицы «Студенты». Изменение фамилии студента.

CREATE PROCEDURE UpdateStudent

@Chit_nom int, */*Объявляем необходимые переменные*/*

@Fam varchar(20)

AS

IF EXISTS (SELECT * FROM Студенты /**Проверяем, существуют ли студенты,*/*

WHERE Читательский_номер = @Chit_nom) /*читательский номер которых равен искомому*/

UPDATE Студенты /*Если такие есть обновляем «Студенты»

SET Фамилия=@Fam /*полю фамилия присваиваем новое значение*/

WHERE Читательский_номер = @Chit_nom /*если читательский номер записи равен искомому*/

7. Вставка в таблицу «Пользование библиотекой2». Разрешается, если есть в таблицах «Студенты», «Сотрудники_библиотеки» и «Экземпляр» соответствующие записи.

```
ALTER  PROCEDURE NewPolzovanieStydentov
@Chit_nomer int, /*Объявляем необходимые переменные*/
@data_vidachi datetime,
@data_priema datetime,
@tab_nomer int,
@Shifr varchar(20)
AS
IF EXISTS (SELECT * FROM Пользование_библиотекой2 /*Проверяем, нет ли уже в таблице */
WHERE Шифр = @Shifr AND /*записи с таким же значением первичного ключа*/
Читательский_номер=@Chit_nomer)
RETURN 0 /*если есть, завершается процедура*/
IF EXISTS (SELECT * FROM Студенты /*проверяем наличие соответствующей записи в*/
WHERE Читательский_номер = @Chit_nomer) /*таблице «Студенты»*/
IF EXISTS (SELECT * FROM Экземпляр /*проверяем наличие соответствующей записи в*/
 WHERE Шифр = @Shifr) /*таблице «Экземпляр»*/
IF EXISTS (SELECT * FROM Сотрудники_библиотеки /*проверяем наличие соответствующей */
WHERE табельный_номер = @tab_nomer) /* записи в таблице «Сотрудники_библиотеки»*/
```

INSERT /*если условия выполняются, добавляем*/
INTO Пользование_библиотекой2 /*в таблицу новые значения*/
VALUES (@Chit_nomer,@data_vidachi,@data_priema,@tab_nomer,@Shifr)

8. Вставка в таблицу «Сотрудники_библиотеки». Проверяется , наличие соответствующей записи в поле номер-отдела в таблице «Отделы».

```
CREATE PROCEDURE NewSotrudnik
 @Tab_nom int, /*Объявляем необходимые переменные*/
 @Fam varchar(20),
 @Name varchar(20),
 @Sec_name varchar(20),
 @data_rogd datetime,
 @Dolgn varchar(20),
 @Nom_otd int
AS
IF EXISTS (SELECT * FROM Сотрудники_библиотеки
/*Проверяем, нет ли уже в таблице */
WHERE Табельный_номер = @Tab_nom) /*записи с таким же значением первичного ключа*/
RETURN 0 /*если есть, завершается процедура*/
IF EXISTS (SELECT * FROM Отделы /*Проверяем, есть ли уже в таблице «Отделы» */
WHERE Номер_отдела = @Nom_otd) /*записи с таким же значением поля номер_отдела*/
INSERT /*если условие выполняется, добавляем*/
INTO Сотрудники_библиотеки /*в таблицу новые значения*/
VALUES ( @Tab_nom ,@Fam , @Name, @Sec_name, @data_rogd, @Dolgn, @Nom_otd)
```

9. Сколько существует должностей в библиотеке.

```
CREATE PROCEDURE KolDolgn
AS  SELECT COUNT (DISTINCT Сотрудники_библиотеки.должность)
```

FROM Сотрудники_библиотеки */*COUNT (DISTINCT <поле>) подсчитывает количество разноименных значений какого-либо поля в таблице*/*

10. Обновление таблицы «Отделы». Изменился начальник отдела.

```
ALTER PROCEDURE UpdateOtdel
@Nom_otdela int,
@Fam_New_Nach_otd varchar(20),
@Tab_Nom_New_Nach_otd int
AS
IF not EXISTS (SELECT * FROM Сотрудники_библиотеки /*Условие проверяет, есть ли в */
WHERE табельный_номер = @Tab_Nom_New_Nach_otd /* библиотеке сотрудник с искомыми*/
AND фамилия=@Fam_New_Nach_otd) /*фамилией и табельным номером*/
RETURN 0 /*если нет, завершается процедура*/
IF EXISTS (SELECT * FROM Отделы /*Условие проверяет, есть ли в библиотеке*/
WHERE Номер_отдела = @Nom_otdela)
/*искомый номер отдела*/
UPDATE Отделы
SET фамилия_начальника_отдела=@Fam_New_Nach_otd /*меняем значение поля «фамилия начальника отдела» на новое*/
UPDATE Отделы
SET табельный_номер_начальника=@Tab_Nom_New_Nach_otd /*меняем значение поля «табельный номер начальника отдела» на новое*/
WHERE Номер_отдела = @Nom_otdela /*меняем только для записи, у которой значение «номер_отдела» равно искомому*/
```

Задание

В базе данных, созданной в лабораторной работе № 2, необходимо реализовать 15 хранимых процедур.