**Лабораторная работа № 6**

**Создание хранимых процедур**

**Задание**

1.В соответствие с техническим заданием создать необходимые хранимые процедуры.

2.Предусмотреть для каждой группы пользователей не менее 5 хранимых процедуры. Некоторые процедуры могут содержать запросы из лабораторной работы №4.

3.Подготовить и сдать преподавателю отчёт по лабораторной работе в печатном виде.

4.Отчёт должен содержать:

* титульный лист;
* описание разработанных процедур;
* для каждой хранимых процедуры привести:
  + назначение процедуры,
  + группы пользователей, которые будут её выполнять в процессе работы,
  + скрипт создания процедуры на языке SQL,
  + скриншот результата выполнения процедуры.

6.Продемонстрировать преподавателю базу данных с работающими процедурами.

**Учебный пример разработки базы данных**

**Создание хранимых процедур**

Хранимые процедуры аналогичны процедурам в языках программирования:

-они обрабатывают входные параметры и возвращают вызывающему коду значения в виде выходных параметров;

-они содержат программные инструкции, которые выполняют операции в базе данных, в том числе вызывающие другие процедуры;

-они возвращают значение состояния вызывающей процедуре или пакету, таким образом передавая сведения об успешном или неуспешном завершении (и причины последнего).

Хранимая процедура состоит из заголовка и тела. В заголовке указывается имя процедуры и объявляются её входные и выходные параметры (если они есть). Имена параметров начинаются со знака @. Параметры являются локальными в пределах процедуры. Все типы данных Transact-SQL можно использовать в качестве параметров.

Для параметров можно использовать следующие свойства:

OUT | OUTPUT

-показывает, что параметр процедуры является выходным. Используйте выходные параметры для возврата значений коду, вызвавшему процедуру.

READONLY

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

Тело процедуры состоит из одной или несколько инструкций Transact-SQL.

Во время создания хранимой процедуры выполняется только проверка синтаксиса. Процедура не компилируется до первого выполнения.

При выполнении процедуры в первый раз она компилируется, при этом определяется оптимальный план получения данных. При последующих вызовах процедуры можно снова использовать уже созданный план, если он еще находится в кэше планов компонента Database Engine.

Ключевое слово RECOMPILE показывает, что компонент Database Engine не кэширует план запроса для этой процедуры, что вызывает ее компиляцию при каждом выполнении.

Рассмотрим пример.

Для нашей учебной базы данных «Кафе» мы создавали запрос «Меню нашего кафе» и представление «Меню» на его основе, которые выдавали список предлагаемых блюд.

На языке SQL запрос выглядел так:

-- Меню нашего кафе

SELECT Вид, Блюдо AS [Название блюда], Цена

FROM Блюда;

Создадим на основе этого запроса хранимую процедуру без параметров с именем «Меню2».

CREATE PROCEDURE Меню2

AS

SELECT Вид, Блюдо AS [Название блюда], Цена

FROM Блюда

ORDER BY Вид, Цена DESC;

Теперь из любого скрипта или из другой процедуры мы можем выполнить инструкцию, запускающую эту процедуру на выполнение:

EXECUTE Меню2;

или коротко:

EXEC Меню2;

И получим тот же результат, что и при выполнении запроса или представления. Конечно везде где возможно надо использовать представления, но есть много вещей, которые нельзя выполнить с помощью представления. Например, нам надо выполнить сразу несколько инструкций. Выбрать помимо списка блюд ещё и список продуктов. Здесь нужна хранимая процедура:

CREATE PROCEDURE БлюдаПродукты

AS

BEGIN

SELECT Вид, Блюдо AS [Название блюда], Цена

FROM Блюда

ORDER BY Вид, Цена DESC;

SELECT Продукт AS [Название продукта], Калорийность

FROM Продукты

ORDER BY Продукт;

END;

GO;

EXEC БлюдаПродукты;

Также хранимые процедуры используются, например, если надо вставить, обновить или удалить данные в таблицах.

CREATE PROCEDURE ДобавлениеПродукта

@Продукт varchar(50),

@Калорийность int

AS

INSERT INTO Продукты ( Продукт, Калорийность)

VALUES ( @Продукт, @Калорийность );

Данная процедура получает в качестве входных параметров название продукта и его калорийность и вставляет одну строку с данными о новом продукте в таблицу Продукты.

Запустим её:

EXEC ДобавлениеПродукта @Продукт=‘картофель’, @Калорийность=80 ;

или сокращенно:

EXEC ДобавлениеПродукта ‘картофель’, 80 ;

В среде MS SQL Server Management Studio имеющиеся в базе данных хранимые процедуры отображаются в Обозревателе объектов в узле «Программирование».

Попробуйте самостоятельно разработать наиболее удобный для работы каждой группы пользователей набор процедур.

**Процедура 1.**

CREATE PROCEDURE [Удаление старых поставок]

@OldDate date

as

DELETE FROM Поставки

WHERE [Дата поставки] <= @OldDate;

**Процедура 2.**

**Назначение:** Запрос на показ цен от разных поставщиков для определенной детали

**Группы пользователей:** Кладовщик.

CREATE PROCEDURE [Поставщики детали]

@Name nvarchar(100)

as

SELECT Z.[Название запчасти],Z.Стоимость,P2.Название as Поставщик,P2.[Откуда поставляет] as Доставлена

FROM Запчасти as Z

JOIN Поставки as P1 ON P1.[Номер поставки]=Z.[Номер поставки]

JOIN Поставщики as P2 ON P1.ID\_поставщика=P2.ID\_поставщика

where Z.[Название запчасти]=@Name

ORDER BY Стоимость

**Процедура 3.**

**Назначение:**

**Группы пользователей:** Администратор,директор.

CREATE PROCEDURE [Модели компании]

@Model nvarchar(50)

as

SELECT [Серийный номер],Модель,[Страна производства],Состояние

FROM Техника

WHERE Компания = (SELECT Компания

FROM Техника

GROUP BY Компания,Модель

HAVING Модель=@Model)

**Процедура 4.**

Назначение: вывод информации о всех должностях и зарплатах.

Запрос может выполнять директор.

CREATE PROCEDURE Информация\_про\_должности

AS

SELECT Сотрудники.Должность, Сотрудники.ЗП

FROM Сотрудники

GROUP BY Сотрудники.Должность, Сотрудники.ЗП

ORDER BY Сотрудники.Должность

**Процедура 5.**

Назначение: вывод информации о всех сотрудниках с зарплатой более указанно суммы.

Запрос может выполнять директор и бухгалтер.

CREATE PROC [Сотрудники с зарплатой более указанной суммы]

@sum int

AS

SELECT Сотрудники.Фамилия + ' ' + Сотрудники.Имя + ' ' + Сотрудники.Отчество AS ФИО,

Сотрудники.Должность, Сотрудники.ЗП

FROM Сотрудники

WHERE Сотрудники.ЗП > @sum

ORDER BY ЗП, ФИО

**Процедура 6.**

**Назначение: “**Поставки за второе полугодия 2021 года. Нужно для создание отчетности

**Группы пользователей:** Кладовщик.

CREATE PROCEDURE [Отчет за поставку после]

@date date

as

SELECT P2.Название,P2.Реквизиты,P2.Телефон,P1.[Стоимость поставки],P1.[Дата поставки],S1.Фамилия as Принял,S1.Телефон as [Телефон сотрудника]

FROM Поставки as P1

JOIN Поставщики as P2 ON P2.ID\_поставщика=P1.[ID\_Поставщика]

JOIN Сотрудники as S1 ON S1.ID\_сотрудника=P1.ID\_сотрудника

WHERE P1.[Дата поставки]>@date

**Процедура 7.**

Назначение: расчёт подоходного налога для сотрудника.

Запрос может выполнять директор.

CREATE PROC [НДФЛ сотрудника]

@id\_сотрудника int,

@ФИО nvarchar(100) OUTPUT,

@подоходныйНалог money OUTPUT

AS

WITH consts AS (SELECT 0.13 as NDFL)

SELECT @ФИО = Сотрудники.Фамилия + ' ' + Сотрудники.Имя + ' ' + Сотрудники.Отчество,

@подоходныйНалог = Сотрудники.ЗП \* consts.NDFL

FROM Сотрудники,consts

WHERE id\_сотрудника = @id\_сотрудника;

**Процедура 8.**

**Назначение: добавление нового сотрудника.**

**Запрос может выполнять директор.**

CREATE PROC Добавление\_сотрудника

@Фамилия nvarchar(100),

@Имя nvarchar(100),

@Отчество nvarchar(100),

@Телефон nvarchar(18),

@Должность nvarchar(100),

@Спец\_навыки nvarchar(7),

@Дата\_рождения date,

@Уровень\_образования nvarchar(10),

@ЗП money,

@ФИО NVARCHAR(300) OUTPUT,

@ДОЛЖНОСТЬ\_ВЫВОД NVARCHAR(200) OUTPUT,

@ЗАРПЛАТА MONEY OUTPUT

AS

INSERT INTO Сотрудники

VALUES (@Фамилия, @Имя, @Отчество,@Телефон,@Должность,@Спец\_навыки, @Дата\_рождения,@Уровень\_образования,@ЗП )

SET @ФИО = @Фамилия + ' ' + @Имя + ' ' + @Отчество

SET @ДОЛЖНОСТЬ\_ВЫВОД = @Должность

SET @ЗАРПЛАТА = @ЗП

**Процедура 9.**

Назначение: увольнение сотрудника.

Запрос может выполнять директор.

CREATE PROCEDURE [Увольнение сотрудника]

@ID\_сотрудника int

AS

DELETE Сотрудники

WHERE Сотрудники.ID\_сотрудника = @ID\_сотрудника

**Процедура 10.**

CREATE PROC Расходы\_на\_скидки

@startDate date,

@endDate date,

@колвоКлиентов int OUTPUT,

@общСумма int OUTPUT

AS

SELECT @колвоКлиентов = COUNT(\*), @общСумма = SUM(Стоимость\*Скидка/100)

FROM Обращения

WHERE (Обращения.[Дата принятия] BETWEEN @startDate AND @endDate)

**Процедура 11.**

CREATE PROC Определенное\_производство\_Компании

@company nvarchar(100),

@country nvarchar(100)

AS

SELECT [Серийный номер],Модель,[Страна производства],Состояние

FROM Техника

WHERE [Страна производства] =@country and Компания=@company

**Процедура 12.**

**Назначение: “**Общая сумма поставок от определенной компании” может использоваться для подготовки к оплате

**Группы пользователей:** Кладовщик.

CREATE PROCEDURE [Кол-во поставок в период]

@price int,

@startDate date,

@endDate date

as

SELECT Поставщики.Название, SUM(Поставки.[Стоимость поставки])

FROM Поставки

JOIN Поставщики ON Поставщики.ID\_Поставщика=Поставки.ID\_поставщика

WHERE (Поставки.[Дата поставки] BETWEEN @startDate AND @endDate)

GROUP BY Поставщики.Название

HAVING SUM(Поставки.[Стоимость поставки])>@price

**Процедура 13.**

CREATE PROCEDURE Добавление\_Поставки

@idPos int,

@sum int,

@countProd NVARCHAR(20),

@idEpl INT,

@date date,

@id INT OUTPUT

AS

INSERT INTO Поставки(ID\_поставщика,[Стоимость поставки],[Ед.товара],ID\_сотрудника,[Дата поставки])

VALUES(@idPos, @sum, @countProd,@idEpl,@date)

SET @id = @@IDENTITY

**Процедура 14.**

CREATE PROCEDURE gg

@поставщик int

as

SELECT ID\_поставщика,Название,[Откуда поставляет],Телефон

FROM Поставщики

WHERE @поставщик IN(SELECT ID\_поставщика FROM Поставки

where Поставки.ID\_поставщика=Поставщики.ID\_поставщика)

ORDER BY Название

**Процедура 15.**

SELECT Фамилия,Имя,ЗП,Должность

FROM Сотрудники as S2

WHERE ЗП > (SELECT AVG(ЗП)

FROM Сотрудники

WHERE Сотрудники.Должность=S2.Должность)

ORDER BY Фамилия,имя