Практическая работа 56
Выборка данных из нескольких таблиц. Создание пользовательских функций

Цель работы: Получить практический опыт выборки данных из нескольких таблиц.

Перечень оборудования и программного обеспечения

Персональный компьютер
Microsoft Office (Word, Visio, Access)
MS SQL Server 2008 R2

Краткие теоретические сведения

С помощью соединения можно получать данные из двух или нескольких таблиц на основе логических связей между ними. Соединения указывают, как Microsoft SQL Server должен использовать данные из одной таблицы для выбора строк из другой таблицы.

Соединение определяет способ связывания двух таблиц в запросе следующим образом:

- для каждой таблицы указываются столбцы, используемые в соединении. В типичном условии соединения указывается внешний ключ из одной таблицы и связанный с ним ключ из другой таблицы;
- указывается логический оператор (например, = или <>,) для сравнения значений столбцов.

Внутреннее соединение можно задавать в предложениях FROM и WHERE.

Внешнее соединение можно указывать только в предложении FROM. Условия соединения сочетаются с условиями поиска WHERE и HAVING для управления строками, выбранными из базовых таблиц, на которые ссылается предложение FROM.

То, что условия соединения задаются в предложении FROM, помогает отделить их от условий поиска, которые могут быть заданы в предложении WHERE. Объединение рекомендуется задавать именно таким способом. Ниже приведен упрощенный синтаксис соединения с использованием предложения FROM стандарта ISO:

FROM first_tablejoin_typesecond_table [ON (join_condition)]

Аргумент join_type задает тип соединения: внутренний, внешний или перекрестный. Аргумент join_condition определяет предикат, который будет

вычисляться для каждой пары соединяемых строк. Ниже приведен пример предложения FROM с заданным соединением:

```
FROM Purchasing.ProductVendor JOIN Purchasing.Vendor
ON (ProductVendor.BusinessEntityID = Vendor.BusinessEntityID)
```

Ниже приведена простая инструкция SELECT, использующая это соединение:

```
SELECT ProductID, Purchasing.Vendor.BusinessEntityID, Name
FROM Purchasing.ProductVendor JOIN Purchasing.Vendor
ON (Purchasing.ProductVendor.BusinessEntityID =
Purchasing.Vendor.BusinessEntityID)
WHERE StandardPrice> $10
AND Name LIKE N'F%'
GO
```

Инструкция возвращает наименование продукта и сведения о поставщике для всех сочетаний запчастей, поставляемых компаниями с названиями на букву F и стоимостью продукта более 10 долларов.

Если один запрос содержит ссылки на несколько таблиц, то все ссылки столбцов должны быть однозначными. В предыдущем примере как таблица ProductVendor, так и таблица Vendor содержат столбец с именем BusinessEntityID. Имена столбцов, совпадающие в двух или более таблицах, на которые ссылается запрос, должны уточняться именем таблицы. Все ссылки на столбец Vendor в этом примере являются уточненными.

Если имя столбца не дублируется в двух или более таблицах, указанных в запросе, то ссылки на него уточнять именем таблицы не обязательно. Это показано в предыдущем примере. Подобную инструкцию SELECT иногда трудно понять, поскольку в ней нет ничего, что указывало бы на таблицы, из которых берутся столбцы. Запрос гораздо легче читать, если все столбцы указаны с именами соответствующих таблиц. Запрос будет читаться еще легче, если используются псевдонимы таблиц, особенно когда имена таблиц сами должны уточняться именами базы данных и владельца. Ниже приведен тот же пример, но чтобы упростить чтение, используются псевдонимы таблиц, уточняющие названия столбцов.

```
SELECT pv.ProductID, v.BusinessEntityID, v.Name
FROM Purchasing.ProductVendor AS pv
JOIN Purchasing.Vendor AS v
ON (pv.BusinessEntityID = v.BusinessEntityID)
WHERE StandardPrice> $10
AND Name LIKE N'F%';
```

В предыдущем примере условие соединения задается в предложении FROM, что является рекомендуемым способом. В следующем запросе это же условие соединения указывается в предложении WHERE:

```
SELECT pv.ProductID, v.BusinessEntityID, v.Name
FROM Purchasing.ProductVendor AS pv, Purchasing.Vendor AS v
WHERE pv.VendorID = v.VendorID
AND StandardPrice> $10
AND Name LIKE N'F%';
```

Список выборки для соединения может ссылаться на все столбцы в соединяемых таблицах или на любое подмножество этих столбцов. Список выборки не обязательно должен содержать столбцы из каждой таблицы в соединении. Например, в соединении из трех таблиц связующим звеном между одной из таблиц и третьей таблицей может быть только одна таблица, при этом список выборки не обязательно должен ссылаться на столбцы средней таблицы.

Хотя обычно в условиях соединения для сравнения используется оператор равенства (=), можно указать другие операторы сравнения или реляционные операторы, равно как другие предикаты.

При обработке соединений в SQL Server механизм запросов выбирает наиболее эффективный метод обработки из нескольких возможных. При физическом выполнении различных соединений можно использовать много разных оптимизаций, поэтому их нельзя надежно прогнозировать.

Столбцы, используемые в условии соединения, не обязательно должны иметь одинаковые имена или одинаковый тип данных. Однако если типы данных не совпадают, то они должны быть совместимыми или допускать в SQL Server неявное преобразование. Если типы данных не допускают неявное преобразование, то условия соединения должны явно преобразовывать эти типы данных с помощью функции CAST. Дополнительные сведения о явных и неявных преобразованиях см. в разделе Преобразование типов данных (компонент DatabaseEngine).

Большинство запросов, использующих соединение, можно переписать с помощью вложенных запросов и наоборот.

Пользовательские функции

Пользовательские функции во многом напоминают хранимые процедуры и представляют упорядоченное множество операторов T-SQL, которые заранее оптимизированы, откомпилированы и могут быть вызваны для выполнения работы в виде единого модуля. Основное различие между пользовательскими функциями и хранимыми процедурами состоит в том, как в них осуществляется возврат полученных результатов.

Разница заключается в том, что возможностей у них меньше (в частности, они должны возвращать только одно значение, например,

скалярное или табличное), но их удобнее использовать с точки зрения синтаксиса.

Синтаксис Functions в SQL Server (Transact-SQL):

```
CREATE FUNCTION [schema_name.]function_name
( [ @parameter [ AS ] [type_schema_name.] datatype
[ = default ] [ READONLY ]
, @parameter [ AS ] [type_schema_name.] datatype
[ = default ] [ READONLY ] ]
)
RETURNS return_datatype
[ WITH { ENCRYPTION
| SCHEMABINDING
| RETURNS NULL ON NULL INPUT
| CALLED ON NULL INPUT
| EXECUTE AS Clause ]
[ AS ]
BEGIN
[declaration_section]
executable_section
RETURN return_value
END;
```

Параметры или аргументы

- schema_name — имя схемы, которой принадлежит эта функция.
- function_name — наименование функции в SQL Server.
- @parameter — один или несколько параметров, которые передаются в функцию.
- type_schema_name — схема, которая владеет типом данных, если это применимо.
- datatype — тип данных для @parameter.
- default — значение по умолчанию для назначения параметру @parameter.
- READONLY — это означает, что @parameter не может быть перезаписана функцией.
- return_datatype — тип данных возвращаемого значения функции.
- ENCRYPTION — это означает, что источник для функции не будет сохранен как обычный текст в системных представлениях SQL Server.
- SCHEMABINDING — это означает, что базовые объекты не могут быть изменены, чтобы влиять на функцию.
- RETURNS NULL ON NULL INPUT — это означает, что функция вернет NULL, если любые параметры имеют значение NULL, без необходимости выполнять функцию.

– CALL ON NULL INPUT — это означает, что функция будет выполняться, даже если любые параметры имеют NULL.
– EXECUTE AS — устанавливает контекст безопасности для выполнения функции.
– return_value — значение, возвращаемое функцией.

Например, создать функцию можно так:

```
CREATE FUNCTION testF(@n1 int, @n2 as int)
RETURNS int
AS
BEGIN
Return (@n1*@n2)
END
```

Или обычным способом средствами Enterprise Manager. Вызвать функцию на выполнение можно так:

```
Select dbo.testF(5, 8)
```

Задания

1 Изучить теоретические сведения.

2 В соответствии с вариантом задания выполнить выборку из нескольких таблиц:

– запрос с использованием логических выражений;
– запрос с агрегатными функциями;
– перекрестный запрос.

3 В соответствии с вариантом задания создать пользовательские функции:

– скалярную;
– возвращающую таблицу.

Порядок выполнения работы

Для примера создадим скалярную функцию, выполняющую расчет среднего значения нагрузки преподавателей по данной специальности по таблице subjects. Но для вычисления среднего значения необходимо иметь данные о количестве предметов, преподаваемом каждым педагогом. Создадим такую таблицу с помощью функции.

а) Создадим функцию nagr () без параметров, результатом работы которой будет таблица, состоящая из двух полей. Это код преподавателя (id_teach) и количество преподаваемых дисциплин.

```
CREATE FUNCTION nagr ()
RETURNS TABLE
AS RETURN
  SELECT subjects.id_teach, COUNT(subjects.id_subject) AS kol
  FROM subjects
  GROUP BY subjects.id_teach
```

Просмотрим, как работает функция:

```
SELECT * FROM dbo.nagr()
```

	id_teach	kol
1	1	2
2	2	5
3	3	2
4	4	1
5	5	2
6	8	3

б) Создадим скалярную функцию sr_n () без параметров, вычисляющую среднее значение в таблице, полученной с помощью функции nagr().

```
CREATE FUNCTION sr_n()
RETURNS int
AS
  BEGIN
    DECLARE @ret int;
    SELECT @ret = AVG(kol)
    FROM nagr()
    RETURN @ret;
END;
```

Проверим результат работы функции:

```
SELECT dbo.sr_n() AS sred
```

	sred
1	2

Запишем запрос на выборку преподавателей, нагрузка которых не превышает среднее значение.

```
SELECT subjects.id_teach, surname_teach, name_teach, patronymic_teach, COUNT(subjects.id_subject) AS kol
  FROM subjects
  JOIN teachers
  ON subjects.id_teach=teachers.id_teach
  GROUP BY subjects.id_teach, surname_teach, name_teach, patronymic_teach
  HAVING COUNT(subjects.id_subject)<=dbo.sr_n()
```

	id_teach	surname_teach	name_teach	patronymic_teach	kol
1	1	Хлестаков	Иван	Александрович	2
2	3	Сквозник-Дмухановски	Антон	Антонович	2
3	4	Пяпкин-Тяпкин	Аммос	Федорович	1
4	5	Добчинский	Петр	Иванович	2

Содержание отчета

1 Название работы
2 Цель работы
3 Перечень технических средств обучения
4 Порядок выполнения работы
5 Ответы на контрольные вопросы
6 Вывод

Варианты заданий

Варианты заданий представлены в практической работе № 39

Контрольные вопросы:

1. Что такое пользовательские функции MS SQL Server?
2. Чем отличаются от хранимых процедур?
3. На какие типы делятся пользовательские функции?
4. Какой инструкцией создаются пользовательские функции?
5. Как осуществляется вызов пользовательской функции?
6. Что такое аргумент пользовательской функции?
7. В чем основное отличие скалярной функции от функции, возвращающей таблицу?

Используемая литература

– Г.Н.Федорова Основы проектирования баз данных. М.: Академия, 2020
– Г.Н.Федорова Разработка, администрирование и защита баз данных. М.: Академия, 2018
– http://www.ikasteko.ru/page/filtracija-dannyh-sql-server
– https://www.bestprog.net/ru/2018/01/02/an-example-of-creating-a-query-in-the-ms-sql-server-database-the-database-is-located-in-the-local-mdf-file_ru/