Проектирование распределенных запросов

После того как подключение к удаленному источнику данных установлено, SQL Server может обращаться к внешним данным с помощью запросов. В табл. 15.2 представлены четыре основных доступных синтаксических метода, которые отличаются методами обработки и настройки.

|  |  |
| --- | --- |
| Место выполнения запроса | Настройка подключения |
| Локальный экземпляр SQL Server | Четырехкомпонентное имя |
| Внешний источник данных | Четырехкомпонентное имя |
| Связанный сервер | openQuery () |
| Подключение, определенное в запросе | OpenDataSource() OpenRowSet() |

**Распределенные запросы и Management Studio**

Утилита Management Studio не поддерживает графические методы инициализации распределенных запросов. Не существует методов перетаскивания подключенного сервера или удаленной таблицы в конструктор запросов. Однако распределенный запрос можно ввести вручную на панели SQL (рис. 15.5), а затем выполнить его.

**Локальные распределенные запросы**

Хотя термин локальный распределенный запрос звучит странно, все не так уж сложно. Это запрос, который собирает внешние данные в SQL Server, а затем выполняет запрос на локальном сервере. Так как обработка таких запросов выполняется на локальном сервере, в них используется синтаксис Т-SQL, и поэтому их иногда называют локальными запросами T-SQL.

**Использование четырехкомпонентного имени**

Если данные находятся на другом экземпляре SQL Server, то полный синтаксис четырехкомпонентного имени следующий:

Сервер. База\_данных. Схема . Имя\_Объекта

Четырехкомпонентное имя может использоваться в любых запросах извлечения или модификации данных. На моем компьютере существует второй экземпляр SQL Server с именем [XPS\Yukon]. Имя владельца объекта является обязательным, если обращение осуществляется к внешнему серверу.

Следующий запрос извлекает таблицу Person из экземпляра SQL2:

SELECT LastName, FirstName

FROM [XPS\Yukon].Family.dbo.person

Результат запроса следующий:

LastName FirstName

Halloway Kelly Halloway James

При выполнении инструкций INSERT, UPDATE и DELETE в качестве распределенных запросов для имени таблицы можно использовать либо четырехкомпонентную форму, либо функцию распределенного запроса. В качестве примера приведем следующий код, который можно взять из файла CHA2\_Convert. sql и который заполняет учебную базу данных СНА2. В этом примере в качестве источника данных для инструкции INSERT использовано четырехкомпонентное имя таблицы. Этот запрос извлекает названия базовых лагерей из электронной таблицы Excel и вставляет их в SQL Server:

INSERT BaseCamp(Name)

SELECT DISTINCT [Base Camp]

FROM CHAl\_Schedule…[Base\_Camp]

WHERE [Base Camp] IS NOT NULL

Если вы уже выполняли сценарий CHA2\_Convert. sql и заполнили свою копию базы СНА2, еще раз запустите сценарий CHA2\_Create. sql, чтобы начатьработу с пустой базы данных.

А вот еще один пример использования четырехкомпонентного имени для распределенного запроса. В нем обновляется база данных Family, находящаяся на другом экземпляре SQL Server:

UPDATE [Noli\SQL2].Family.dbo.Person

SET LastName = ‘Wilson’

WHERE PersonID = 1

**Использование функции OpenDataSource ()**

Использование функции OpenDataSource () ничем не отличается от использования четырехкомпонентного имени для доступа к связанному серверу, за исключением того, что функция определяет подключение в своем теле, а не использует предопределенный связанный сервер. В то время как определение связи в программе пропускает определение самого сервера, если местонахождение связи изменяется, это коснется всех запросов, использующих функцию OpenDataSource (). К тому же функция OpenDataSource () не принимает переменные в качестве аргументов.

Функция OpenDataSource () заменяет имя сервера в четырехкомпонентном имени и может использоваться в любой инструкции DML.

Синтаксис функции OpenDataSource () выглядит довольно простым:

OpenDataSource {провайдер, строка\_инициализации)

Однако первое впечатление обманчиво. Строка\_инициализации представляет собой символьную строку с несколькими параметрами, разделенными точками с запятыми (точный список параметров зависит от конкретного источника данных). Потенциально в строке инициализации указывается источник данных, его местоположение, дополнительные параметры, время ожидания подключения, идентификатор пользователя, его пароль и каталог. В строке инициализации должны быть указаны все необходимые параметры подключения к источнику данных, в том числе контекст безопасности. Отдельные параметры в строке инициализации не нужно заключать в кавычки. Самой распространенной ошибкой, замеченной при реализации функции OpenDataSource (), является путаница между запятыми и точками с запятой.

Если функция OpenDataSource () подключается к другому серверу с помощью Windows, то необходима аутентификация с поддержкой Kerberos.

Вот относительно простой пример использования функции OpenDataSource () в качестве механизма доступа к таблице в другом экземпляре SQL Server:

SELECT FirstName, Gender

FROM OPENDATASOURCE(

1SQLOLEDB1,

‘Data Source=NOLI\SQL2;User ID=Joe;Password=j’

).Family.dbo.Person;

Результат выполнения запроса:

FirstName      Gender

Adam       M

Alexia      F

В следующем примере распределенного запроса, использующего функцию OpenDataSource (), мы ссылаемся на базу данных Cape Hatteras Adventures. Поскольку файл Access содержит всего одну базу данных и обращение к таблицам не требует указания владельца, эти части в четырехкомпонентном имени можно опустить:

SELECT ContactFirstName, ContactLastName FROM OPENDATASOURCE(

‘Microsoft.Jet.OLEDB.4.0′,

‘Data Source =

С:\SQLServerBible\CHAl\_Customers.mdb’

)…Customers;

Результат выполнения запроса:

ContactFirstName ContactLastName

Neal                 Garrison

Melissa            Anderson

Gary                Quill

Для иллюстрации использования функции OpenUpdateSource () в запросе UPDATE мы обновим все строки в рабочей книге Excel CHAl\_Schedule.xls. Именованный диапазон был определен заранее: Tours ‘ =Sheetl! $Е$5 : $Е$24 1. Теперь он будет использован в запросе SQL в качестве таблицы в источнике данных. Вместо того чтобы обновлять отдельно каждую ячейку рабочего листа, этот запрос выполняет инструкцию UPDATE, затрагивающую все строки, в которых названием тура является Gauley River Rafting, и обновляет столбец Base Camp значением Ashville.

Распределенный запрос SQL Server для доступа к механизму Jet, который открывает рабочий лист Excel, будет использовать поставщика OLE DB. Функции OpenDataSource () мы передаем только имя сервера в четырехкомпонентной форме; при этом, как и в случае с Access, имя базы данных и владельца опускаем:

UPDATE OpenDataSource(

1Microsoft.Jet.OLEDB.4.0′,

1          Data Source=C:\SQLServerBible\CHAl\_Schedule.xls;

User ID=Admin;Password=;Extended properties=Excel 5.0′

)…Tour

SET [Base Camp] = ‘Ashville1

WHERE Tour = ‘Gauley River Rafting';

На рис. 15.6 показан план выполнения распределенного запроса UPDATE. Он начинается справа с блока Remote Scan, который возвращает все 19 строк именованного диапазона Excel. После этого данные обрабатываются в SQL Server. Логическая операция Remote Update сводится к тому, что распределенный запрос обновления на самом деле изменяет значения всего в двух строках.

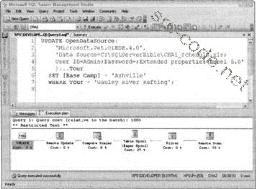
[](http://src-code.net/img/Nilsen_SQLServer2005_image083.jpg)

Рис. 15.6. План выполнения распределенного запроса обновления, использующего функцию OpenDataSource ()

Чтобы завершить пример, следующий запрос считывает тот же рабочий лист Excel и проверяет, действительно ли имело место обновление. И снова функция OpenDataSource (), единственная в распределенном запросе, указывает на внешний сервер:

SELECT \*

FROM OpenDataSource(

‘Microsoft.Jet.OLEDB.4.0′,

‘Data Source=C:\SQLServerBible\CHAl\_Schedule.xls;

User ID=Admin;Password=;Extended properties=Excel 5.0′

)…Tour

WHERE Tour = ‘Gauley River Rafting';

Результат выполнения запроса будет следующим:

Base Camp   Tour

Ashville      Gauley River Rafting

Ashville      Gauley River Rafting

**Сквозные распределенные запросы**

Сквозные запросы выполняются на внешнем источнике данных, после чего результат передается на SQL Server. Основная причина использования таких запросов заключается в сокращении объема данных, передаваемых с сервера (внешнего источника данных) и клиента (SQL Server). Вместо передачи миллиона строк на SQL Server, чтобы использовать только двадцать пять из них, может оказаться выгоднее извлечь только эти строки из внешнего источника данных.

В то же время сквозные запросы должны использовать синтаксис внешнего сервера. Если источником данных является база данных Oracle, то в сквозном запросе должен использоваться язык PL/SQL; если база данных Access — то Access SQL.

В случае сквозного запроса, модифицирующего данные, сам тип удаленных данных определяет, будет ли обновление выполняться локально или удаленно.

?               Если обновляются данные на другом экземпляре SQL Server, то операция будет выполняться именно на нем.

?               Если данные обновляются на сервере, отличном от SQL Server, то поставщик определяет, где именно будут обрабатываться данные. Часто передаваемые запросы просто отбирают корректные строки. Эти строки передаются на SQL Server, обновляются на нем, а затем возвращаются удаленному источнику данных для обновления.

Существуют две формы локальных распределенных запросов: одна для связанных серверов и одна для внешних источников данных, определяемых в запросе. Также существуют две формы сквозных распределенных запросов. В одном случае функция OpenQuery () использует уже подключенный сервер; во втором — функция OpenRowSet () определяет связь непосредственно в запросе.

**Использование четырехкомпонентного имени**

Если распределенный запрос осуществляет доступ к другому экземпляру SQL Server, четырехкомпонентное имя называют гибридным методом распределенных запросов. В зависимости от предложений FROM и WHERE, SQL Server будет пытаться передать как можно большую часть запроса удаленному серверу, чтобы увеличить производительность.

При создании сложных распределенных запросов, использующих четырехкомпонентное имя, довольно сложно предсказать заранее, какая часть запроса будет передана. Мне встречались серверы, которые, в зависимости от предложения WHERE, либо передавали весь запрос, либо каждая таблица передавала свой запрос, либо запрос передавала только одна таблица.

Из четырех представленных методов распределенных запросов лучшими являются использование четырехкомпонентного имени и функции OpenQuery () соответственно. Оба метода используют административную настройку подключения к удаленному серверу, что делает запрос более надежным, если конфигурация сервера изменяется.

Выбор между этими двумя методами зависит от общего объема данных, их извлекаемого количества и производительности сервера. Рекомендую вам протестировать оба метода и сравнить их планы выполнения, после чего выбрать тот, который лучше проявляет себя в вашей ситуации. Если оба метода показывают равную производительность, то используйте метод четырехкомпонентного имени, чтобы сервер имел возможность автоматически оптимизировать распределенный запрос.

**Функция OpenQuery ()**

Среди передаваемых запросов функция OpenQuery () задействует ресурсы уже подключенного сервера, поэтому с ней проще обращаться. Она также воспринимает изменения в конфигурации серверов без изменения кода.

Функция OpenQuery () используется в языке SQL DML в качестве таблицы. Она принимает два аргумента: имя связываемого сервера и сам передаваемый запрос. В следующем примере функция OpenQuery () используется для извлечения данных из рабочей книги Excel CHAl\_Schedule:

SELECT \*

FROM OPENQUERY(CHAl\_Schedule,

‘SELECT \* FROM Tour WHERE Tour = "Gauley River Rafting"’);

Результат выполнения запроса:

Tour                      Base Camp

Gauley River Rafting Ashville Gauley River Rafting Ashville

Как показано на рис. 15.7, передаваемый запрос, использующий функцию OpenQuery (), практически не требует обработки сервером — ему возвращаются ровно две строки. Предложение WHERE обрабатывается механизмом Jet при извлечении данных из рабочего листа Excel.

В следующем примере функция OpenQuery () дает указание механизму Jet, чтобы тот извлек только две строки, требующие обновления. Реальная инструкция UPDATE выполняется на сервере, а результат возвращается внешнему источнику данных. В результате передаваемый запрос выполняет в инструкции UPDATE только часть функции SELECT:

UPDATE OPENQUERY(CHAl\_Schedule,

‘SELECT \* FROM Tour WHERE Tour = "Gauley River Rafting"’)

SET [Base Camp] = ‘Ashville’

WHERE Tour = ‘Gauley River Rafting';

**Функция OpenRowSet ()**

Эта функция является двойником функции OpenDataSet (). Обе требуют, чтобы в распределенном запросе был полностью определен удаленный источник данных. Функция OpenRowSet () имеет дополнительный аргумент, определяющий передаваемый запрос:

SELECT ContactFirstName, ContactLastName

FROM OPENROWSET (‘Microsoft.Jet.OLEDB.4.0′, ‘C:\SQLServerBible\CHAl\_Customers.mdb'; ‘Admin';”,

‘SELECT \* FROM Customers WHERE CustomerlD = 1′);

[](http://src-code.net/img/Nilsen_SQLServer2005_image084.jpg)

Рис. 15.7. Распределенный запрос, использующий функцию OpenQuery (), возвращает только две строки, отобранные предложением WHERE

Результат запроса следующий:

ContactFirstName ContactLastName

Tom                Mercer

Чтобы выполнить обновление с помощью функции OpenRowSet (), вставьте ее на место модифицируемой таблицы. В следующем примере мы изменим фамилию заказчика в базе данных Access. Предложение WHERE инструкции UPDATE обрабатывается передаваемой частою функции OpenRowSet ():

UPDATE OPENROWSET (‘Microsoft.Jet.OLEDB.4.0′,

1С:\SQLServerBible\CHAl\_Customers.mdb1; ‘Admin’

‘SELECT \* FROM Customers WHERE CustomerlD = 1′)

SET ContactLastName = ‘Wilson';

Операции массового заполнения поддерживаются функцией OpenRowSet (),