Базы данных

07 | Использование табличных выражений

План

- Представления / отображения (Views)
- Временные таблицы
- Табличные переменные
- Табличные функции
- Производные таблицы
- Обобщенные табличные выражения

Что такое отображение (View)?

- Объект БД, можно ссылаться, как на таблицу
- Именованный SELECT-запрос

```
CREATE VIEW HumanResources.EmployeeList
(EmployeeID, FamilyName, GivenName)
AS
SELECT EmployeeID, LastName, FirstName
FROM HumanResources.Employee;
```

Введение в отображения

- Отображение не сохраняет данные (кроме индексированных)
- WITH SCHEMABINDING блокирует изменения в схеме базовых таблиц
- Добавление уникального кластерного индекса делает отображение индексированным
  - Данные сохраняются на диск, увеличивая производительность
- SQL Server Enterprise Edition учитывает индексированные отображения
- Вставка и обновление данных в отображение может происходить только в одной базовой таблице

Запросы к отображениям

- Отображения (представления) – именованные запросы с хранимым в БД определением:
  - Обеспечивают абстракцию, инкапсуляцию и упрощение;
  - С точки зрения администратора, отображения обеспечивают дополнительный уровень безопасности.
- Используются в запросах SELECT как таблицы:

```
CREATE VIEW Sales.vSalesOrders
AS
SELECT  oh.OrderID, oh.Orderdate, oh.CustomerID,
        od.LineItemNo, od.ProductID, od.Quantity
FROM Sales.OrderHeaders AS oh
INNER JOIN Sales.OrderDetails AS od
ON od.OrderID = oh.OrderID;
```

```
SELECT OrderID, CustomerID, ProductID
FROM Sales.vSalesOrder
ORDER BY OrderID;
```

DEMO

Запросы к отображениям

Временные таблицы

```
CREATE TABLE #tmpProducts
(ProductID INTEGER,
 ProductName varchar(50));
GO
…
SELECT * FROM #tmpProducts;
```

- Временные таблицы предназначены для хранения временных результатов в рамках сессии пользователя:
  - Создаются в БД tempdb и удаляются автоматически;
  - Создаются с префиксом “#”;
  - Глобальные временные таблицы создаются с префиксом “##”.

Табличные переменные

```
DECLARE @varProducts table
(ProductID INTEGER,
 ProductName varchar(50));
…
SELECT * FROM @varProducts
```

- Введены из-за частых рекомпиляций временных таблиц;
- Используются как временные таблицы, но ограничены пакетом;
- Не имеют статистики, поэтому изменения не вызывают рекомпиляции (всегда ожидается 1 строка);
- Используйте только на маленьких наборах данных.

DEMO

Временные таблицы и табличные переменные

Табличные функции (ТФ)

- ТФ – именованные объекты с определениями хранимыми в БД;
- ТФ возвращают виртуальную таблицу в точку вызова;
- В отличие от отображений, ТФ поддерживают входные параметры:
  - ТФ можно представлять как параметризованные отображения.

```
CREATE FUNCTION Sales.fn_GetOrderItems (@OrderID AS Integer)
RETURNS TABLE
AS
RETURN
(SELECT ProductID, UnitPrice, Quantity
FROM Sales.OrderDetails
WHERE OrderID = @OrderID);
…
SELECT * FROM Sales.fn_GetOrderItems (1025) AS LineItems;
```

Основы пользовательской табличной функции

- Предложение RETURNS также определяет формат таблицы. Область видимости табличной переменной локализована внутри функции.
- Операторы INSERT Transact-SQL в теле функции формируют ответ, добавляя строки в возвращаемую переменную, определенную в предложении RETURN.
- При выполнении оператора RETURN данные из табличной переменной возвращаются в виде результирующей таблицы. Оператор RETURN не может иметь аргументов.

Пример пользовательской табличной функции

Определяем таблицу:

```
CREATE TYPE [dbo].[IntTableType] AS
TABLE([ID] [int] NULL)
```

```
-- Получаем список входящих счетов, содержащих все леги из списка @Legs
CREATE FUNCTION [dbo].[getInvoices4AllLegs]
(@Legs dbo.IntTableType READONLY)
RETURNS @Invoices TABLE (InvoiceID int, IncrementID int)
AS
/* DECLARE @Legs dbo.IntTableType
INSERT INTO @Legs VALUES(126504); INSERT INTO @Legs VALUES(126830);
SELECT * FROM [dbo].[getInvoices4AllLegs](@Legs) */
BEGIN
DECLARE @cnt int
select @cnt=count(ID) FROM @Legs
INSERT @Invoices(InvoiceID, IncrementID)
SELECT I.InvoiceID, I.IncrementID FROM dbo.Invoices As I
INNER JOIN dbo.Bindings As B ON I.InvoiceID=B.InvoiceID AND I.IncrementID=B.IncrementID AND
B.LegID IN (select ID FROM @Legs)
GROUP BY I.InvoiceID, I.IncrementID
HAVING count(I.InvoiceID)=@cnt
RETURN
END
```

DEMO

Использование табличных функций

Производные таблицы
Введение

```
SELECT orderyear, COUNT(DISTINCT custid) AS cust_count
FROM
        (SELECT YEAR(orderdate) AS orderyear, custid
        FROM Sales.Orders) AS derived_year
GROUP BY orderyear;
```

- Производные таблицы – именованные запросы, созданные в рамках внешнего оператора SELECT
- Не хранятся в БД – представляют виртуальную таблицу
- Область видимости производной таблицы – запрос, в котором она определена

Производные таблицы. Основные правила

- Производные таблицы обязаны:
  - Иметь псевдоним;
  - Иметь уникальные имена для всех столбцов;
  - Не использовать предложение ORDER BY clause (без TOP или OFFSET/FETCH);
  - Избегать множественных ссылок в том же запросе.
- Производные таблицы могут:
  - Использовать внутренние или внешние псевдонимы для столбцов;
  - Ссылаться на параметры и/или переменные;
  - Быть вложенными в другие производные таблицы.

Производные таблицы
Указание псевдонимов столбцов

- Псевдонимы столбцов могут объявляться внутри:

```
SELECT orderyear, COUNT(DISTINCT custid) AS cust_count
FROM (  SELECT YEAR(orderdate) AS orderyear, custid
        FROM Sales.Orders) AS derived_year
GROUP BY orderyear;
```

- Или снаружи:

```
SELECT orderyear, COUNT(DISTINCT custid) AS cust_count
FROM (  SELECT YEAR(orderdate), custid
        FROM Sales.Orders) AS
        derived_year(orderyear, custid)
GROUP BY orderyear;
```

DEMO

Использование производных таблиц

Обобщенные табличные выражения (CTE)

```
WITH CTE_year (OrderYear, CustID)
AS
(
        SELECT YEAR(orderdate), custid
        FROM Sales.Orders
)
SELECT OrderYear, COUNT(DISTINCT CustID) AS Cust_Count
FROM CTE_year
GROUP BY orderyear;
```

- CTE – именованные табличные выражения, определенные в запросе;
- CTE похожи на производные таблицы по области видимости и требованиям к именованию;
- В отличие от производных таблиц, CTE поддерживают множественные ссылки и рекурсию.

Обобщенные табличные выражения (CTE)
Рекурсия

```
WITH OrgReport (ManagerID, EmployeeID, EmployeeName, Level)
AS
(
        SELECT e.ManagerID, e.EmployeeID, EmployeeName, 0
        FROM HR.Employee AS e
        WHERE ManagerID IS NULL
        UNION ALL
        SELECT e.ManagerID, e.EmployeeID, e.EmployeeName, Level + 1
        FROM HR.Employee AS e
        INNER JOIN OrgReport AS o ON e.ManagerID = o.EmployeeID
)
SELECT * FROM OrgReport
OPTION (MAXRECURSION 3);
```

- Укажите корневой запрос;
- Используйте UNION ALL, чтобы добавить рекурсивный запрос;
- Запросы к CTE могут использовать опцию MAXRECURSION.

DEMO

Использование обобщенных табличных выражений

Изучено

- Представления / отображения (Views);
- Временные таблицы;
- Табличные переменные;
- Табличные функции;
- Производные таблицы;
- Обобщенные табличные выражения.