Семинар 14 – Программирование на Transact-SQL

Обзор

На этом семинаре Вы будете использовать определяемые пользователем функции, написанные на языке Transact-SQL для работы с данными в базе данных **AdventureWorksLT**.

Что необходимо для выполнения

- Доступ к облачному сервису Microsoft Azure SQL Database с БД **AdventureWorksLT**.

или

- Установленный Microsoft SQL Server с SQL Server Management Studio и БД **AdventureWorksLT**.

Задача 1: Определение новых объектов в базе данных

Отдел разработки компании обратил внимание, что при написании различных запросов к базе данных, разработчики часто пользуются общими повторяющимися фрагментами кода. Поэтому было принято решение добавить в БД ряд объектов – функций, представлений и хранимых процедур – облегчающих написание запросов к БД за счет предоставления нового слоя абстракции.

1. Получение итоговой суммы продаж по клиенту

Отдел продаж планирует ввести для клиентов ряд новых акций, которые зависят от потраченной клиентом суммы. Поэтому Вас просят написать функцию с именем **fn_GetOrdersTotalDueForCustomer**, принимающую один входной параметр **@CustomerID** (идентификатор клиента) и возвращающую общую сумму, потраченную клиентом на оплату всех заказов (т.е. сумму **TotalDue**). Приведите примеры использования написанной функции для клиентов с идентификаторами 1 и 30113.

Подсказка: посмотрите документацию по созданию функций и оператору CREATE FUNCTION в справочнике по Transact-SQL.

2. Получение всех типов адресов клиентов

Отдел продаж проводит ревизию адресов клиентов и Вам поручили создать представление **vAllAddresses** для отображения всех типов адресов всех клиентов. Представление должно содержать следующие атрибуты: CustomerID, AddressType, AddressID, AddressLine1, AddressLine2, City, StateProvince, CountryRegion, PostalCode. Протестируйте созданное представление.

3. Получение всех адресов клиента

Вам необходимо для оформления карточки клиента реализовать функцию **fn_GetAddressesForCustomer**, возвращающую все адреса для конкретного клиента (входной параметр **@CustomerID** – идентификатор клиента) из ранее созданного представления **vAllAddresses**. Возвращаемый набор данных должен содержать все доступные атрибуты из представления. Протестируйте созданную функцию – напишите запрос, возвращающий адреса клиентов в виде одного набора данных для клиентов с идентификаторами 0, 29502 и 29503.

4. Определение максимальной и минимальной суммы продажи товара

Для получения статистики по продажам товаров компании Вас просят создать функцию **fn_GetMinMaxOrderPricesForProduct**, принимающую на вход идентификатор товара **@ProductID** и возвращающую строку с двумя столбцами: **MinUnitPrice** и **MaxUnitPrice**, содержащий соответственно минимальную и максимальную цену (из столбца **UnitPrice**) за которую был продан данный товар. Провести тестирование функции для товаров с идентификаторами 0 и 711.

5. Получение всех описаний товара

Отдел маркетинга хочет удостовериться, что все описания продаваемых товаров четко описывают информацию. Вас просят написать функцию **fn_GetAllDescriptionsForProduct**, которая возвращает все описания для товара. Функция принимает на вход идентификатор товара **@ProductID** и возвращает все найденные для данного товара описания в виде кортежей со следующими атрибутами: ProductID, Name, MinUnitPrice, MaxUnitPrice, ListPrice, ProductModel, Culture, Description. Здесь Name – наименование товара, MinUnitPrice, MaxUnitPrice – результат для товара из функции **fn_GetMinMaxOrderPricesForProduct**, ListPrice – розничная цена, ProductModel – поле Name из таблицы **ProductModel**, Culture – язык описания из таблицы **ProductModelProductDescription**, Description – описание из таблицы **ProductDescription**. Подсказка: используйте представление **vProductAndDescription** для сокращения количества соединений в запросе и улучшении его читаемости. Провести тестирование функции для товаров с идентификаторами 0 и 711.

Задача 2: Материализация представлений

Вы ожидаете бурный рост числа клиентов, а значит и их адресов в базу данных. Вместе с этим приходит понимание, что некоторые часто используемые представления могут быть оптимизированы для улучшения скорости доступа к данным. Вы решаете создать уникальный кластерный индекс на представлении **vAllAddresses** из задания о получении всех типов адресов клиентов.

1. Материализация представления vAllAddresses

Включите параметры “Include Actual Execution Plan” и “Include Live Query Statistics” и выполните запрос

```
SELECT * FROM [dbo].[vAllAddresses]
```

Проанализируйте план выполнения запроса и убедитесь, что добавление индекса в представление должно улучшить план. Убедившись, что представление создано с опцией **WITH SCHEMABINDING**, добавьте уникальный кластерный индекс **UIX_vAllAddresses** на представление **vAllAddresses**, включив в него все атрибуты. После создания индекса еще раз выполните запрос (возможно, с опцией “WITH (NOEXPAND)”, т.е. `SELECT * FROM [dbo].[vAllAddresses] WITH (NOEXPAND)`) и убедитесь, что план выполнения использует созданный Вами индекс.

Подсказка: посмотрите документацию по созданию индексированных представлений и оператору CREATE INDEX в справочнике по Transact-SQL.

P.S.

Не все представления можно сделать материализованными. Составьте два списка представлений из БД **AdventureWorksLT**: которые *можно* и которые *нельзя материализовать*. Подумайте над тем, как можно было бы изменить некоторые представления (или базовые таблицы), чтобы получить возможность покрыть представление индексом.