Задание

1. Изучить и законспектировать материал лекции.
2. Фотоотчет присылать на электронную почту

С уважением, Хвастова Светлана Ивановна

!!! Если возникнут вопросы обращаться по телефону 0721389311 (ватсап). Электронная почта: xvsviv@rambler.ru

ХРАНИМЫЕ ПРОЦЕДУРЫ И ТРИГГЕРЫ.

Триггеры представляют собой методы, с помощью которых можно обеспечивать целостность базы данных даже в том случае, если она используется множеством приложений.

Триггер – это специальный тип хранимой процедуры, которая автоматически выполняется при каждой попытке изменить защищаемые ей данные. Триггеры обеспечивают целостность данных, предотвращая их несанкционированное или неправильное изменение.

Допустим, что в базе данных есть таблицы, связанные через поле Surname. Например, это могут быть таблица клиентов предприятия и их заказов. Разумно определить триггер, который при каждой попытке удалить запись клиента проверит наличие у него заказов и позволит удалить эту запись только при их отсутствии. Конечно, подобную задачу можно решить при помощи средств декларативной ссылочной целостности. Однако при помощи триггеров можно создавать значительно более сложные рабочие правила. Можно создать триггер, который при каждом добавлении записи в таблицу заказов анализирует предыдущие заказы этого же клиента и определяет приемлемый срок оплаты этого заказа.

Триггеры не принимают параметров и не возвращают значений. Они выполняются неявно. То есть триггер запускается только при попытке изменения данных.

Триггеры могут иметь до 32 уровней вложенности. Вложенные триггеры работают следующим образом: пусть при создании записи о новом заказе

триггер добавляет информацию в таблицу неоплаченных счетов. При этом выполняется другой триггер, который проверяет, имеет ли клиент просроченные неоплаченные счета и, если они есть, триггер выводит сообщение об этом. В этом примере один триггер обновляет таблицу, вызывая при этом выполнение другого триггера.

По умолчанию все триггеры (INSERT, DELETE, UPDATE) срабатывают сразу после выполнения операции изменения данных. Эти триггеры относятся к типу AFTER (после). Начиная с SQL Server 2000 появилась еще одна группа триггеров – INSTEAD OF (вместо), которые выполняются вместо оператора, предполагающего изменение данных.

С точки зрения быстродействия триггеры не имеют никаких преимуществ. Выполнение триггера связано с постоянными обращениями к различным таблицам. Соответственно его работа быстрее, если используемые таблицы находятся в оперативной памяти, и медленнее – если данные считываются с диска.

Триггер является частью транзакции, следовательно, если триггер терпит неудачу, отменяется вся транзакция. И наоборот, если какая-то часть транзакции не удалась, то и триггер будет отменен.

В своей работе триггеры используют таблицы Inserted и Deleted. Это логические таблицы, они постоянно находятся в оперативной памяти и имеют туже структуру, что и таблица, для которой создан триггер. Каждая добавленная к защищаемой триггером таблице строка добавляется и в таблицу Inserted. Обновление производится почти так же, как и удаление с последующей вставкой. Когда строка обновляется, старая строка записывается в таблицу Deleted, затем обновленная строка записывается в базовую таблицу и в таблицу Inserted.

Создание триггера

Триггер создается оператором CREATE TRIGGER. Рассмотрим его синтаксис:

CREATE TRIGGER [владелец.] имя_триггера

ON [владелец.] имя_таблицы │имя_представления

FOR (AFTER│ INSTEAD OF} (INSERT │UPDATE I DELETE}

[WITH ENCRYPTION]

AS оператор_SQL

Таблица может иметь произвольное количество триггеров любых типов (INSERT, UPDATE или DELETE). ПО умолчанию триггер выполняется после изменения данных, однако, если указать параметр INSEAD OF, то создается триггер, выполняющийся вместо изменения данных.

Каждая операция (INSERT, UPDATE и DELETE) может вызывать выполнение произвольного количества триггеров. С единственным ограничением имена триггеров, вызываемых одной операцией, должны быть уникальными. Изменить триггер можно, удалив его и создав заново в другом виде или при помощи оператора ALTER TRIGGER. При удалении таблицы, имеющей триггеры, все они также удаляются.

При создании триггеров необходимо придерживаться следующих правил:

- Триггеры создаются для поддержания целостности данных, ссылочной целостности и рабочих правил.
- Нельзя создавать триггеры для временных таблиц. Однако триггеры могут к таким таблицам обращаться так же, как и к представлениям.
- Триггер не может возвращать результирующих наборов данных. Это значит, что к использованию оператора SELECT при его создании нужно подходить крайне осторожно. Обычно в этих случаях используется оператор SELECT с директивой IF EXISTS.
- С помощью опции WITH ENCRYPTION исходный код триггера, хранящийся в таблице syscomments, можно зашифровать.
- Операторы WRITETEXT не инициализируют триггеры. Они используются для изменения данных типа Text или Image, а эти изменения не заносятся в журнал транзакций.

- В триггерах нельзя использовать следующие операторы: все операторы CREATE, все операторы DROP, ALTER TABLE, ALTER DATABASE, TRUNCATE TABLE, GRANT и REVOKE, RECONFIGURE, LOAD DATABASE или TRANSACTION, UPDATE STATISTICS, SELECT INTO и все операторы DISK.
- Операторы отмены транзакций, входящие в состав триггера, могут стать причиной непредсказуемого поведения операторов вызывающей программы.

Триггеры также применяются для поддержки правил. Конечно, для этого часто достаточно использования ограничений ANSI, значений по умолчанию или пользовательских типов данных, однако при необходимости обращения к другим таблицам триггеры оказываются совершенно незаменимыми. Кроме того, при помощи триггеров реализуется механизм ссылочной целостности. Нужно отметить, что триггеры изначально были разработаны именно для этой цели. Они особенно эффективны при каскадном удалении и обновлении данных. При изменении данных условия триггеров проверяются в последнюю очередь, а в начале проверяются ограничения. То есть если условие ограничения нарушено, то операция отменяется, и триггер не срабатывает.

Отдельное внимание стоит уделить триггерам типа INSTEAD OF. Если триггер создается с этой опцией, то код триггера выполняется не после заданной пользователем (или удаленной программой) команды, а вместо нее. Например, разумно использовать триггеры INSTEAD OF для сообщений о невозможности удаления какого-либо объекта. Разумеется, эту функцию можно реализовать и с помощью триггера AFTER, однако в этом случае придется отменять уже проделанную операцию, что неприемлемо для высоко загруженных систем, где ведется строгий контроль производительности.

Хранимые процедуры. Назначение

Хранимая процедура – это последовательность компилированных операторов Transact – SQL, хранящихся в системной базе данных SQL Server.

Хранимые процедуры предварительно откомпилированы, поэтому эффективность их выполнения выше, чем у обычных запросов. Хранимые процедуры работают непосредственно на сервере и хорошо укладываются в модель клиент- сервер.

Существует два вида хранимых процедур: системные и пользовательские.

Системные хранимые процедуры предназначены для получения информации из системных таблиц и выполнения различных служебных операций и особенно полезны при администрировании базы данных. Их имена начинаются с sp_ (stored procedure).

Пользовательские хранимые процедуры создаются непосредственно разработчиками или администраторами базы данных.

Полезность хранимых процедур определяется в первую очередь высокой скоростью их выполнения. Кроме того, они являются средством систематизации часто выполняемых операций. При выполнении в первый раз хранимой процедуры можно выделит ряд этапов.

- Процедура разбивается на отдельные компоненты лексическим анализатором выражений.
- Компоненты, ссылающиеся на объекты базы данных (таблицы, индексы представления и т. п.), сопоставляются с этими объектами с предвари тельной проверкой их существования. Этот процесс носит название *разрешение ссылок*.
- В системной таблице syscomments сохраняется исходный текст процедуры, а в таблице sysobjects – ее название.
- Создается предварительный план выполнения запроса. Этот предварительный план называется нормализованным планом или деревом запроса и хранится в системной таблице sysprocedures.
- При первом выполнении хранимой процедуры дерево запроса считывается и окончательно оптимизируется. Выполняется ранее созданный План процедуры.

Такая схема дает возможность при повторных вызовах не тратить время на синтаксический анализ, разрешение ссылок и компиляцию дерева запросов.

А при последующих вызовах выполняется только пятый шаг. Причем план хранимой процедуры после первого выполнения содержится в быстродействующем процедурном кэше. Это значит, что во время вызова процедуры скорость его считывания будет очень высока.

Использование хранимых процедур имеют еще ряд дополнительных Преимуществ.

- Хранимые процедуры позволяют выделять правила в отдельную структуру. В дальнейшем эти правила используются многими приложениями, образуя устойчивый к ошибкам интерфейс данных. Выгода такого подхода состоит в том, что можно осуществлять изменение правил только для отдельной части объектов базы данных, а не для всех ее приложений.
- Использование хранимых процедур значительно повышает производительность запросов, однако наибольшей ее прирост достигается при выполнении многократно повторяющихся операций, когда план запроса постоянно хранится в системном кэше.
- Хранимые процедуры могут принимать аргументы при запуске и возвращать значения (в виде результирующих наборов данных).
- Хранимые процедуры могут запускаться по расписанию (в режиме автоматического выполнения), задаваемому при запуске SQL Server.
- Хранимые процедуры используются для извлечения или изменения данных в любое время.
- Хранимые процедуры, в отличие от триггеров, вызываются явно. То есть при непосредственном обращении к процедуре из приложения, сценария, пакета или задачи.

Хранимые процедуры – мощное средство обработки данных. Системные хранимые процедуры играют очень важную роль в администрировании и поддержке базы данных. Пользовательские хранимые процедуры при меняются при решении практически любых задач. Кроме того, пользователь может

получить право выполнения хранимой процедуры, даже если он не имеет права доступа к объектам, к которым обращается процедура.