**ЛАБОРАТОРНАЯ РАБОТА № 10.**

*«Триггеры»*

**Цель**: получить практические навыки создания и использования триггеров

**Теоретическая часть**

Триггер — это программный код, который запускается на выполнение, когда в связанном с ним объекте базы данных таблице происходит определенное действие. Каждый триггер имеет следующие основные составляющие:

* имя;
* действие;
* исполнение.

Имя триггера может содержать максимум 128 символов. Действием триггера может быть или инструкция DML (INSERT, UPDATE или DELETE), или инструкция DDL. Таким образом, существует два типа триггеров: триггеры DML и триггеры DDL. Исполнительная составляющая триггера обычно состоит из хранимой процедуры или пакета операторов SQL.

**1. Создание триггера DML**

Триггер создается с помощью инструкции CREATE TRIGGER, которая имеет следующий синтаксис:

**CREATE TRIGGER Имя\_триггера**

**ON {Имя\_таблицы | Имя\_представления}**

**[WITH опции [,...]]**

**{FOR | AFTER | INSTEAD OF} {[INSERT] [,] [UPDATE] [,] [DELETE]}**

**[WITH APPEND]**

**{ AS sql\_операторы | EXTERNAL NAME имя\_метода}**

Фигурные скобки {} означают выбор из нескольких возможных элементов.

Тип триггера определяется с помощью параметров AFTER и INSTEAD OF. (Параметр FOR является синонимом параметра AFTER.) Триггеры типа AFTER вызываются после выполнения действия, запускающего триггер, а триггеры типа INSTEAD OF выполняются вместо действия, запускающего триггер. Триггеры AFTER можно создавать только для таблиц, а триггеры INSTEAD — как для таблиц, так и для представлений.

Параметры INSERT, UPDATE и DELETE задают действие триггера. Под действием триггера имеется в виду инструкция Transact-SQL. Допускается любая комбинация этих трех инструкций.

Исполнение триггера указывается в спецификации AS sql\_операторы.

**Изменение структуры триггера**

Язык Transact-SQL также поддерживает инструкцию ALTER TRIGGER , которая модифицирует структуру триггера. Эта инструкция обычно применяется для изменения тела триггера. Все предложения и параметры инструкции ALTER TRIGGER имеют такое же значение, как и одноименные предложения и параметры инструкции CREATE TRIGGER.

Для удаления триггеров в текущей базе данных применяется инструкция DROP TRIGGER.

**Использование виртуальных таблиц deleted и inserted**

При создании действия триггера обычно требуется указать, ссылается ли он на значение столбца до или после его изменения действием, запускающим триггер. По этой причине, для тестирования следствия инструкции, запускающей триггер, используются две специально именованные виртуальные таблицы:

**deleted** — содержит копии строк, удаленных из таблицы;

**inserted** — содержит копии строк, вставленных в таблицу.

Таблицы deleted и inserted являются специальными представлениями журнала транзакций, доступными лишь во время срабатывания триг­гера и в пределах триггера.

Структура этих таблиц эквивалентна структуре таблицы, для которой определен триггер.

Таблица deleted используется в том случае, если в инструкции CREATE TRIGGER указывается предложение DELETE или UPDATE, а если в этой инструкции указывается предложение INSERT или UPDATE, то используется таблица inserted. Это означает, что для каждой инструкции DELETE, выполненной в действии триггера, создается таблица deleted. Подобным образом для каждой инструкции INSERT, выполненной в действии триггера, создается таблица inserted.

Инструкция UPDATE рассматривается, как инструкция DELETE, за которой следует инструкция INSERT. Поэтому для каждой инструкции UPDATE, выполненной в действии триггера, создается как таблица deleted, так и таблица inserted (в указанной последовательности).

Когда для таблицы с соответствующими триггерами выполняется инструкция DML INSERT, UPDATE или DELETE, для всех изменений в этой таблице всегда создаются версии строк. Когда триггеру требуется информация из таблицы deleted, он обращается к данным в хранилище версий строк. В случае таблицы inserted, триггер обращается к самым последним версиям строк.

**2. Области применения DML-триггеров**

**Триггеры AFTER**

Триггеры AFTER вызываются после того, как выполняется действие, запускающее триггер. Триггер AFTER задается с помощью ключевого слова AFTER или FOR. Триггеры AFTER можно создавать только для базовых таблиц.

Триггеры этого типа можно использовать для выполнения, среди прочих, следующих операций:

* создание журнала аудита действий в таблицах базы данных;
* реализация бизнес-правил;
* принудительное обеспечение ссылочной целостности.

**Создание журнала аудита**

В примере 10.1 показывается, как с помощью триггеров можно создать журнал аудита действий в таблицах базы данных.

Пример 10.1. Создание журнала аудита действий в таблицах базы данных

/\* Таблица audit\_sales используется в качестве журнала аудита действий в таблице sales/

**USE Shop;**

**GO**

**CREATE TABLE audit\_sales (**

**recId INT PRIMARY KEY,**

**recdate DATETIME,**

**goodId\_old INT,**

**goodId\_new INT NULL,**

**quant\_old NUMERIC(5,2),**

**quant\_new NUMERIC(5,2) NULL,**

**price\_old NUMERIC(5,2),**

**price\_new NUMERIC(5,2) NULL,**

**saleId\_old INT,**

**saleId\_new INT NULL**

**);**

**GO**

**CREATE TRIGGER modify\_sales ON Sales AFTER UPDATE, DELETE**

**AS**

**BEGIN**

**DECLARE @goodId\_old INT, @goodId\_new INT**

**DECLARE @quant\_old NUMERIC(5,2), @quant\_new NUMERIC(5,2)**

**DECLARE @price\_old NUMERIC(5,2), @price\_new NUMERIC(5,2)**

**DECLARE @saleId\_old INT, @saleId\_new INT**

**SET @goodId\_old = (SELECT goodId FROM deleted)**

**SET @goodId\_new = (SELECT goodId FROM inserted)**

**SET @quant\_old = (SELECT quantity FROM deleted)**

**SET @quant\_new = (SELECT quantity FROM inserted)**

**SET @price\_old = (SELECT price FROM deleted)**

**SET @price\_new = (SELECT price FROM inserted)**

**SET @saleId\_old = (SELECT saleId FROM deleted)**

**SET @saleId\_new = (SELECT saleId FROM inserted)**

**INSERT INTO audit\_sales VALUES**

**(GETDATE(), @goodId\_old, @goodId\_new, @quant\_old, @quant\_new, @price\_old, @price\_new, @saleId\_old, @saleId\_new)**

**END**

В примере 10.1 создается таблица audit\_sales, в которой сохраняются все изменения в таблице Sales, произведенные с помощью операторов UPDATE или DELETE. При этом записываются как старые значения полей, так и измененные. Запись в таблицу audit\_sales производится с помощью триггера modify\_sales. Этот триггер активируется при каждого изменения или удалении записи в таблице Sales.

Если из таблицы Sales случайно или умышленно была удалена запись командой

**DELETE FROM Sales**

**WHERE quantity=2.0 and price=14.50**

то в таблице audit\_sales появится следующая запись

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **recdate** | **goodId\_**  **old** | **goodId\_**  **new** | **quant\_**  **old** | **quant\_**  **new** | **price\_**  **old** | **price\_**  **new** | **saleId\_**  **old** | **saleId\_**  **new** |
| 2018-11-21 14:00:05 | 4 | NULL | 2.00 | NULL | 14.50 | NULL | 3 | NULL |

Если в таблице sales была изменена запись командой

**UPDATE Sales**

**SET price=4,50,**

**SET quantity=1**

**WHERE quantity=2.0 and price=14.50**

То в таблице audit\_sales появится следующая запись

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **recdate** | **goodId\_**  **old** | **goodId\_**  **new** | **quant\_**  **old** | **quant\_**  **new** | **price\_**  **old** | **price\_**  **new** | **saleId\_**  **old** | **saleId\_**  **new** |
| 2018-11-28 17:00:54.347 | 3 | 3 | 2,00 | 1,00 | 14,50 | 4,50 | 3 | 3 |

Таким образом, по таблице audit\_sales можно отследить все изменения, сделанные в таблице Sales и при необходимости отменить их.

**Реализация бизнес-правил**

С помощью триггеров можно создавать бизнес-правила для приложений. Например, если в магазине прибыль составляет 15% от выручки и 20% прибыли идет в фонд развития бизнеса, то в таблице sales можно предусмотреть два дополнительных столбца – прибыль (profit) и фонд развития бизнеса (fund). При каждом изменении количества (quantity) проданного товара триггер может вычислять прибыль и отчисления в фонд развития.

Создание такого триггера показано в примере 10.2.

Пример 10.2. Создание триггера для бизнес-правила.

Триггер check\_price является примером использования триггера для реализации бизнес-правила

**USE Shop;**

**GO**

**CREATE TRIGGER profit ON sales AFTER INSERT**

**AS**

**BEGIN**

**DECLARE @profit DECIMAL(5,2)**

**DECLARE @quant DECIMAL(5,2)**

**DECLARE @price DECIMAL(5,2)**

**DECLARE @fund Decimal(5,2)**

**DECLARE @recId INT**

**SET @quant = (SELECT quantity FROM inserted)**

**SET @price=(SELECT price FROM inserted)**

**SET @profit=@quant\*@price\*0.15**

**SET @recId=(SELECT recordId FROM inserted)**

**SET @fund=@profit\*0.2**

**UPDATE sales**

**SET profit=@profit,**

**fund=@fund**

**WHERE recordId=@recId**

**END**

**Принудительное обеспечение ограничений целостности**

В связанных таблицах значения внешнего ключа должно выбираться только из множества значений первичного ключа. Если внешнему ключу не соответствует ни один из первичных ключей, то возникает нарушение целостности данных. Для обеспечения целостности данных используются ограничения целостности данных, обычно устанавливаемые при создании таблиц.

Если при создании таблиц такие ограничения по каким-либо причинам не были установлены, то обеспечение целостности данных может быть возложено на соответствующие триггеры. В этом случае задачей триггера является проверка вставляемой в таблицу строки на наличие внешнего ключа, значение которого равно значению одного из первичных ключей связанной таблицы. Если значение внешнего ключа во вставляемой строке не равно ни одному из значений первичного ключа связанной таблицы, то триггер не должен позволить вставку такой строки.

В примере 10.3 показано принудительное обеспечение ссылочной целостности посредством триггеров для таблиц Employees и Depts.

Пример 10.3. Обеспечение ссылочной целостности посредством триггера

**USE Shop;**

**GO**

**CREATE TRIGGER empl\_integrity ON Employees AFTER INSERT, UPDATE**

**AS**

**BEGIN**

**IF (SELECT Depts.deptId FROM depts, inserted**

**WHERE depts.deptId = inserted.deptId) IS NULL**

**BEGIN**

**ROLLBACK TRANSACTION**

**PRINT 'Строка не вставлена/не изменена'**

**END**

**ELSE PRINT 'Строка вставлена/изменена'**

**END**

Триггер empl\_integrity в примере 8.3 проверяет ссылочную целостность для таблиц Employees и Depts. При вставке или изменении строки в таблице Employees триггер проверяет значение поля deptId. Если такого значения нет в таблице depts, то для того чтобы не произошло нарушение ссылочной целостности операция вставки или модификации отменяется с помощью инструкции

ROLLBACK TRANSACTION.

**Триггеры INSTEAD OF**

Триггер с предложением INSTEAD OF заменяет соответствующее действие, которое запустило его. Этот триггер выполняется после создания соответствующих таблиц inserted и deleted, но перед выполнением проверки ограничений целостности или каких-либо других действий.

Триггеры INSTEAD OF можно создавать как для таблиц, так и для представлений, он определяется также, как триггер AFTER, за тем исключением, что он может определяться только для одной операции - INSERT, DELETE или UPDATE применяется только для таблиц).

Например, добавим в таблицу “Журнал учета продаж» (Ledger) столбец IsDeleted:

**USE Shop;**

**ALTER TABLE Ledger**

**ADD IsDeleted BIT NULL;**

Столбец IsDeleted указывает, удалена ли запись. То есть вместо удаления записи из таблицы будет выполняться заполнение поля IsDeleted, а сама запись останется в таблице.

Пример 10.4. Триггер Instead Of

**USE Shop**

**GO**

**CREATE TRIGGER sale\_delete**

**ON Ledger**

**INSTEAD OF DELETE**

**AS**

**UPDATE Ledger**

**SET IsDeleted = 1**

**WHERE saleId =(SELECT saleId FROM deleted)**

Таким образом, удаляемые записи на самом деле не будут удаляться, просто у них будет устанавливаться значение для столбца IsDeleted:

**Триггеры first и last**

Компонент Database Engine позволяет создавать несколько триггеров для каждой таблицы или представления и для каждой операции (INSERT, UPDATE и DELETE) с ними. Кроме этого, можно указать порядок выполнения для нескольких триггеров, определенных для конкретной операции. С помощью системной процедуры sp\_settriggerorder можно указать, что один из определенных для таблицы триггеров AFTER будет выполняться первым или последним для каждого обрабатываемого действия. Эта системная процедура имеет параметр @order, которому можно присвоить одно из трех значений:

**first** — указывает, что триггер является первым триггером AFTER, выполняющимся для модифицирования действия;

**last** — указывает, что данный триггер является последним триггером AFTER, выполняющимся для инициирования действия;

**none** — указывает, что для триггера отсутствует какой-либо определенный порядок выполнения. (Это значение обычно используется для того, чтобы выполнить сброс ранее установленного порядка выполнения триггера как первого или последнего.)

Применение системной процедуры sp\_settriggerorder показано в примере 10.5.

Пример 10.5. Применение системной процедуры sp\_settriggerorder

**EXEC sp\_settriggerorder @triggername = 'modify\_sales',**

**@order = 'first', @stmttype='update'**

Узнать порядок выполнения триггера можно с помощью следующих средств:

* системной процедуры sp\_helptrigger;
* функции OBJECTPROPERTY.

Возвращаемый системной процедурой sp\_helptrigger результирующий набор содержит столбец order, в котором указывается порядок выполнения указанного триггера. При вызове функции OBJECTPROPERTY в ее втором параметре указывается значение ExecIsFirstTrigger или ExecIsLastTrigger, а в первом параметре всегда указывается идентификационный номер объекта базы данных. Если указанное во втором параметре свойство имеет значение TRUE, функция возвращает значение 1.

**3. Триггеры DDL и области их применения**

Триггеры для инструкций DDL имеют следующий синтаксис:

**CREATE TRIGGER [schema\_name.]trigger\_name**

**ON {ALL SERVER | DATABASE}**

**[WITH {ENCRYPTION | EXECUTE AS clause\_name]**

**{FOR | AFTER} {event\_group | event\_type | LOGON} AS {batch | EXTERNAL NAME method\_name}**

Как можно видеть по их синтаксису, триггеры DDL создаются таким же способом, как и триггеры DML. Для изменения и удаления этих триггеров используются те же инструкции ALTER TRIGGER и DROP TRIGGER, что и для триггеров DML.

Первым делом при определении триггера DDL нужно указать его область действия. Предложение DATABASE указывает в качестве области действия триггера DDL текущую базу данных, а предложение ALL SERVER — текущий сервер.

После указания области действия триггера DDL нужно в ответ на выполнение одной или нескольких инструкций DDL указать способ запуска триггера. В параметре event\_type указывается инструкция DDL, выполнение которой запускает триггер, а в альтернативном параметре event\_group указывается группа событий языка Transact-SQL. Триггер DDL запускается после выполнения любого события языка Transact-SQL, указанного в параметре event\_group. Ключевое слово LOGON указывает триггер входа.

Кроме сходства триггеров DML и DDL, между ними также есть несколько различий. Основным различием между этими двумя видами триггеров является то, что для триггера DDL можно задать в качестве его области действия всю базу данных или даже весь сервер, а не всего лишь отдельный объект. Кроме этого, триггеры DDL не поддерживают триггеров INSTEAD OF. Для триггеров DDL не требуются таблицы inserted и deleted, поскольку эти триггеры не изменяют содержимого таблиц.

**Триггеры DDL уровня базы данных**

В примере 10.5 показано, как можно реализовать триггер DDL, чья область действия распространяется на текущую базу данных.

Пример 8.5. Создание триггера DDL уровня базы данных

**USE Shop;**

**GO**

**CREATE TRIGGER prevent\_drop\_triggers**

**ON DATABASE FOR DROP\_TRIGGER**

**AS PRINT 'Для удаления триггера нужно отключить "prevent\_drop\_triggers" '**

**ROLLBACK**

Триггер в примере 10.5 предотвращает удаление любого триггера для базы данных Shop любым пользователем. Предложение DATABASE указывает, что триггер prevent\_drop\_trigger является триггером уровня базы данных. Ключевое слово DROP\_TRIGGER указывает предопределенный тип события, запрещающий удаление любого триггера.

**Триггеры DDL уровня сервера**

Триггеры уровня сервера реагируют на серверные события. Триггер уровня сервера создается посредством использования предложения ALL SERVER в инструкции CREATE TRIGGER. В зависимости от выполняемого триггером действия, существует два разных типа триггеров уровня сервера: обычные триггеры DDL и триггеры входа. Запуск обычных триггеров DDL основан на событиях инструкций DDL, а запуск триггеров входа — на событиях входа.

В примере 10.6 демонстрируется создание триггера уровня сервера, который является триггером входа.

Пример 10.6. Создание триггера уровня сервера, являющегося триггером входа

**USE master;**

**GO**

**CREATE LOGIN login\_test WITH PASSWORD = 'login\_test§$!',**

**CHECK\_EXPIRATION = ON;**

**GO**

**GRANT VIEW SERVER STATE TO login\_test;**

**GO**

**CREATE TRIGGER connection\_limit\_trigger**

**ON ALL SERVER WITH EXECUTE AS 'login\_test'**

**FOR LOGON AS**

**BEGIN IF ORIGINAL\_LOGIN()= 'login\_test' AND**

**(SELECT COUNT(\*) FROM sys.dm\_exec\_sessions**

**WHERE is\_user\_process = 1 AND original\_login\_name = 'login\_test') > 1**

**ROLLBACK;**

**END;**

В примере 10.6 сначала создается имя входа SQL Server login\_test, которое потом используется в триггере уровня сервера. По этой причине, для этого имени входа требуется разрешение VIEW SERVER STATE, которое и предоставляется ему посредством инструкции GRANT. После этого создается триггер connection\_limit\_trigger. Этот триггер является триггером входа, что указывается ключевым словом LOGON.

С помощью представления sys.dm\_exec\_sessions выполняется проверка, был ли уже установлен сеанс с использованием имени входа login\_test. Если сеанс уже был установлен, выполняется инструкция ROLLBACK. Таким образом имя входа login\_test может одновременно установить только один сеанс.

**Задание на лабораторную работу.**

Для одной из таблиц своей базы данных создать

1. Триггер журнала аудита
2. Триггер для реализации бизнес-правил
3. Триггер для обеспечения целостности
4. Триггер для запрещения удаления триггеров

**Контрольные вопросы**

1. Что такое триггер?
2. Какие триггеры используются в SQL Server?
3. Для каких операций используются DML триггеры?
4. Для каких операция используются DDL триггеры?
5. Как можно установить порядок применения триггеров?