УВАЖАЕМЫЕ СТУДЕНТЫ! Изучите теоретический материал к практической работе, ответьте письменно на контрольные вопросы, выполните задание практической работы, подготовьте отчет, сопровождая этапы выполнения скриншотами.

Результаты работы, фотоотчет, предоставить преподавателю на e-mail: xvsviv@rambler.ru в трехдневный срок с момента получения задания.

При возникновении вопросов по приведенному материалу обращаться по следующим номерам телефонов:072-138-93-11.

ВНИМАНИЕ!!! При отправке работы, не забывайте указывать ФИО студента, наименование дисциплины, дата проведения занятия (по расписанию).

Лабораторная работа

Создание триггеров в базах данных (различных типов).

Цель работы:

Изучить способы создания триггеров на языке Transact-SQL, приобрести практические навыки разработки и использования триггеров в приложениях баз данных.

Задания

1. Изучите приведенный в лабораторной работе теоретический материал.
2. Изучите приведенные в лабораторной работе примеры создания триггеров.
3. Выполните индивидуальное задание (см. после теоретических сведений):
   - создание DML-триггеров;
   - создание DDL-триггера;
   - получить метаинформацию по создаваемым триггерам.
4. Покажите работу преподавателю.
5. Ответьте на контрольные вопросы.

Краткие теоретические сведения

Триггер – это специальный тип хранимых процедур, запускаемых сервером автоматически в ответ на изменение таблиц или представлений.

Каждый триггер привязывается к конкретной таблице или представлению.

Триггеры применяются для обеспечения целостности данных и реализации сложной бизнес-логики, а также создания записи аудита о модификации данных

Все производимые триггером модификации данных рассматриваются как одна транзакция. В случае обнаружения ошибки или нарушении целостности данных происходит откат этой транзакции. Тем самым внесение изменений будет запрещено. Будут также отменены все изменения, уже сделанные триггером.

Типы триггеров

Триггеры подразделяются на триггеры языка определения данных (DataDefinitionLanguage – DDL), на триггеры языка манипулирования данными (DataManipulationLanguage – DML) и триггеры входа. Последний тип триггера запускается при регистрации в экземпляре SQL Server.

Триггеры DML используют логические (концептуальные) таблицы DELETED и INSETED. По своей структуре они подобны таблице, на которой определен триггер, то есть таблице, к которой применяется действие пользователя. В таблицах DELETED и INSETED содержатся старые или новые значения строк, которые могут быть изменены действиями пользователя.

По типу изменения данных в таблице существует четыре типа триггеров:

– INSERT TRIGGER – триггеры запускаются при попытке вставки данных с помощью команды INSERT;

– UPDATE TRIGGER – триггеры запускаются при попытке изменения данных с помощью команды UPDATE;

– DELETE TRIGGER – триггеры этого типа запускаются при попытке удаления данных с помощью команды DELETE;

– Триггеры, создаваемые с учетом одновременного возникновения и совпадения событий.

По типу поведения триггеры классифицируются на триггеры FOR (AFTER) и INSTEAD OF.

Триггер FOR (AFTER) выполняется после выполнения команды, изменяющей данные в таблице. Если же команда по каким-либо причинам не может быть успешно завершена, то триггер также не выполняется. Изменения данных в результате выполнения запроса пользователя и выполнение триггера осуществляется в теле одной транзакции. Если триггер не выполняется, то также будет выполнен откат и пользовательских изменений.

По умолчанию в SQL Server 2008 R2 все триггеры являются триггерами типа FOR (AFTER).

Триггеры типа FOR (AFTER) невозможно определить для представлений. Для каждой таблицы можно определить более одного триггера AFTER для каждой операции (INSERT, UPDATE, DELETE).

Триггер INSTEAD OF выполняется в обход действий, вызывавших их срабатывание, заменяя эти действия. Например, обновление таблицы, в которой есть триггер INSTEAD OF, вызовет срабатывание этого триггера. В результате вместо оператора обновления выполняется код триггера. Это позволяет размешать в триггере сложные операторы обработки, которые дополняют действия оператора, модифицирующего таблицу.

Триггеры INSTEAD OF могут быть определены для таблиц и представлений. Можно определить только один триггер INSTEAD OF для каждой операции (INSERT, UPDATE, DELETE).

Триггеры INSTEAD OF не разрешены для обновляемых представлений, использующих параметр WITH CHECK OPTION. SQL Server вызывает ошибку, если триггер INSTEAD OF добавляется к обновляемому представлению с параметром WITH CHECK OPTION. Пользователь должен удалить этот параметр при помощи инструкции ALTER VIEW перед определением триггера INSTEAD OF.

Нельзя устанавливать триггеры на временные или системные таблицы, хотя на такие таблицы разрешено ссылаться в операторах T-SQL самого триггера.

Нельзя триггеры INSTEAD OF DELETE и INSTEAD OF UPDATE определять в таблицах, где заданы ограничения каскадной ссылочной целостности ON DELETE или ON UPDATE соответственно.

К таблице разрешено привязывать триггеры обоих классов: INSTEAD OF и FOR (AFTER). Если в таблице определены ограничения и триггеры обоих классов, то первым из них срабатывает триггер INSTEAD OF, затем обрабатываются ограничения и последним срабатывает FOR (AFTER)-триггер. При нарушении ограничения выполняется откат действий INSTEAD OF-триггера. Если нарушаются ограничения или происходят какие-либо другие события, не позволяющие модифицировать таблицу, FOR (AFTER)-триггер не исполняется.

Как и в случае хранимых процедур, глубина вложенности триггеров достигает 32 уровней, также возможно рекурсивное срабатывание триггеров

С помощью системной хранимой процедуры sp_settriggerorder возможно указать, какой триггер будет выполняться первым, а какой последним. Однако

назначить триггер, который будет выполняться вторым, третьим и т. д. (но не последним) нельзя.

Синтаксис оператора создания триггера

```
CREATETRIGGER<имя триггера>
ON[<имясхемы.>]{ <имя таблицы | <имя представления> }
[ WITH ENCRYPTION | EXECUTE AS {<CALLER | SELF | <USER>>} ]
{ FOR | AFTER | INSTEAD OF }
{ [DELETE] [,] [INSERT] [,] [UPDATE] }
[ WITH APPEND ]
[ NOT FOR REPLICATION ]
AS
{sql_statement| EXTERNAL NAME <method specifier> }
```

Имя схемы – имя схемы, которой принадлежит триггер DML. Триггеры DML ограничены областью схемы таблицы или представления, для которых они созданы. Аргумент <имя схемы> не может быть указан для триггеров DDL или входа.

Имя триггера идентификатор триггера должен соответствовать правилам для идентификаторов, за исключением того, что trigger_name не может начинаться с символов # или ##.

Талица|Представление – таблица или представление, в которых выполняется триггер DML.

WITH ENCRYPTION – шифрует и делает недоступным (включая администратора) текст инструкции CREATETRIGGER. Использование аргумента WITH ENCRYPTION не позволяет публиковать триггер как часть репликации SQL Server. Параметр WITH ENCRYPTION не может быть указан для триггеров CLR.

EXECUTE AS – указывает контекст безопасности, в котором выполняется триггер. Позволяет управлять учетной записью пользователя, используемой экземпляром SQL Server для проверки разрешений на любые объекты базы данных, ссылаемые триггером.

FOR | AFTER – тип AFTER указывает, что триггер DML срабатывает только после успешного выполнения всех операций в инструкции SQL, запускаемой триггером. Все каскадные действия и проверки ограничений, на которые имеется ссылка, должны быть успешно завершены, прежде чем триггер сработает.

INSTEAD OF – указывает, что триггер DML срабатывает вместо инструкции SQL, используемой триггером, переопределяя таким образом

действия выполняемой инструкции триггера. Аргумент INSTEAD OF не может быть указан для триггеров DDL или триггеров входа.

{ [DELETE] [,] [INSERT] [,] [UPDATE] } – определяет инструкции изменения данных, по которым срабатывает триггер DML, если он применяется к таблице или представлению. Необходимо указать как минимум одну инструкцию. В определении триггера разрешены любые их сочетания в любом порядке.

Для триггеров INSTEAD OF параметр DELETE не разрешен в таблицах, имеющих ссылочную связь с указанием каскадного действия ON DELETE. Точно так же параметр UPDATE не разрешен в таблицах, имеющих ссылочную связь с указанием каскадного действия ON UPDATE.

WITH APPEND – указывает, что требуется добавить триггер существующего типа. Аргумент WITH APPEND не может быть использован для триггеров INSTEAD OF или при явном указании триггера AFTER. Аргумент WITH APPEND может использоваться только при указании параметра FOR без INSTEAD OF или AFTER из соображений поддержки обратной совместимости. Аргумент WITH APPEND не может быть указан, если указан параметр EXTERNAL NAME (в случае триггера CLR).

Предложение будет удалено в следующей версии Microsoft SQL. Не следует использовать его при создании новых приложений

NOT FOR REPLICATION– при создании триггера с этим параметром запрещается его запуск при модификации таблиц механизмами репликации. Этот параметр часто используется при создании системных триггеров поддержки подсистемы репликации.

sql_statement –набор команд, которые будут выполнены при запуске триггера (тело триггера).

Примеры триггеров

Примеры триггеров приведены для базы данных, диаграмма которой представлена на рисунке 1.

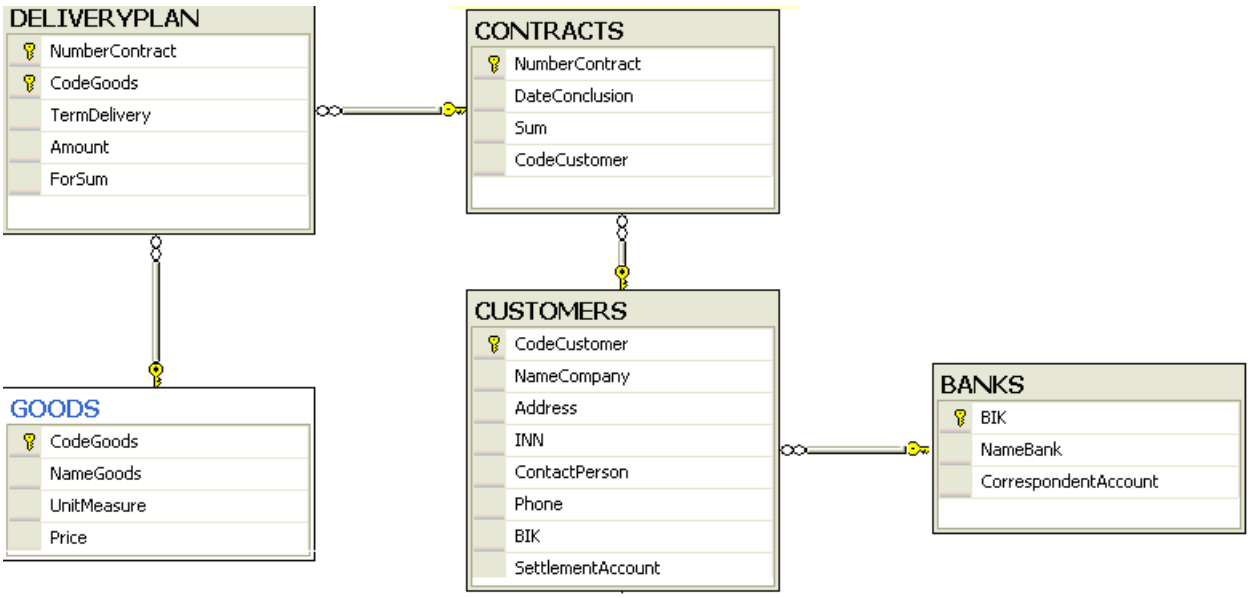

Рисунок 1 – Схема учебной базы данных

1. Запретить вставлять новые строки в таблицу BANKS, выводя при этом сообщение «Вставка строк запрещена»:

```
CREATE TRIGGER BANKS_zapINS
ON dbo.BANKS
FOR INSERT AS
PRINT 'Вставка строк запрещена'
ROLLBACK TRAN
```

2. Не заключать ДОГОВОРА с ПОКУПАТЕЛЯМИ, если на фирме прекращена поставка товаров на склад.

```
CREATE TRIGGER noDeliveryPlan
ON dbo.DeliveryPlan
FOR INSERT, UPDATE
AS
IF EXISTS
(SELECT *
FROM Inserted I
JOIN dbo.GOODS G ON I.CodeGoods = G.CodeGoods
WHERE G.DiscontinuedDate IS NOT NULL)
BEGIN
RAISERROR ('ПРЕКРАШЕНА ПОСТАВКА ТОВАРА',16,1)
ROLLBACKTRAN
END
```

3. Запретить поставки товары ПОКУПАТЕЛЮ, если потребуется отпустить более половины запаса товара.

```
CREATE TRIGGER controlAmount
ON dbo.GOODS
FOR UPDATE
AS
IF EXISTS
(SELECT *
FROM INSERTED I JOIN DELETED DON I.CodeGoods = D.CodeGoods
WHERE (D.Amount-I.Amount)>D.Amount/2 AND
(D.Amount- I.Amount)>0)
BEGIN
RAISERROR (Запрещено отпускать товар в количестве, превышающим 50%% товарного запаса',16,1)
ROLLBACKTRAN
END
```

4. Создать для таблицы DELIVERYPLAN триггер типа DELETE, который будет выводить информацию о попытках удаления и количестве удаляемых строк:

```
CREATE TRIGGER del_DeliveryPlan
ON DeliveryPlan
FOR DELETE
AS
PRINT 'Попыткаудаления '+STR(@@ROWCOUNT)+' строкизтаблицыDeliveryPlan'
PRINT 'Пользователь '+ CURRENT_USER
IF CURRENT_USER!= 'dbo'
BEGIN
PRINT 'Удаление запрщено'
ROLLBACK TRANSACTION
END
ELSEPRINT 'Удаление разрешено'
```

Созданный триггер будет выводить информацию о количестве строк, которое пытается удалить пользователь, и имя пользователя, выполнившего команду DELETE. Если пользователь не 'dbo', то удаление запрещается и выдается соответствующее предупреждение.

5. Изменение (модификация) триггера.

Изменить триггер BANKS_zapINS так, чтобы было разрешена вставка строки в таблицу BANKS_zapINS, но при этом дополнительно записывал в специальную таблицу все данные вставляемой строки с указанием имени пользователя, пытавшегося вставить строку, а также времени вставки данных.

Для этого нам необходимо создать в базе данных дополнительную таблицу, в которой будет храниться список вставляемых строк. Выполним это с помощью следующей команды:

```
SELECT TOP 0 *, suser = SUSER_SID(), _date = GETDATE()
INTO BANKS_buffer FROM BANKS
```

Теперь изменим триггер:

```
ALTER TRIGGER BANKS_zapINS ON BANKS FOR INSERT AS
INSERT INTO BANKS_buffer SELECT *, SUSER_SID(), GETDATE()
FROMinsertedPRINT 'Операция вставки строки зафиксирована'
```

6. Удалить триггер

```
DROP TRIGGER BANKS_zapINS
```

Получение информации о триггере

1. Получить код Transact-SQL, выполняемого при вызове триггера:

```
sp_helptext [@objname =] 'name'
```

name – имя триггера, о котором необходимо получить информацию.

2. Получить список триггеров, определенных для конкретной таблицы базы данных, используется следующая хранимая процедура:

```
sp_helptrigger [@tabname =] 'table' [,[@triggertype =] 'type']
```

table – имя таблицы, для которой нужно получить список созданных триггеров.

type – определяет тип триггеров, о которых будет выведена информация. Если этот аргумент опущен, то будет возвращен список всех триггеров.

Список столбцов возвращаемого результата и их назначение:

TRIGGER_NAME (sysname) – имя триггера, присвоенное ему при создании или после переименования;

TRIGGER_OWNER (sysname) – имя владельца триггера;

ISUPDATE (int) – значение 1 означает, что триггер будет вызываться при выполнении команды UPDATE;

ISDELETE (int) – значение 1 означает, что триггер будет вызываться при выполнении команды DELETE;

ISINSERT (int) – значение 1 означает, что триггер будет вызываться при выполнении команды INSERT.

3. Просмотр списка объектов, от которых зависит триггер:

sp_depends [@objname] 'object'

object – содержит имя триггера, о котором необходимо получить информацию.

Возвращаемый результат разделен на две таблицы: первая для объектов, от которых зависит триггер, вторая – для объектов, зависящих от триггера.

Список столбцов первой таблицы следующий:

NAME (nvarchar(40)) – имя объекта, от которого зависит триггер;

TYPE (nvarchar(16)) – тип объекта, от которого зависит триггер;

UPDATED (nvarchar(9)) – определяет, является ли объект изменяемым;

SELECTED (nvarchar(8)) – определяет, включается ли объект в результат выборки SELECT;

COLUMN (sysname) – имя столбца или другого параметра, от которого конкретно зависит триггер.

Список столбцов второй таблицы:

NAME (nvarchar(40)) — имя объекта, который зависит от триггера;

TYPE (nvarchar(16)) — тип объекта, который зависит от триггера.

Создание триггера в среде MS SQL Server Management Studio

Для создания триггера необходимо вызвать соответствующий пункт контекстного меню объекта **Триггеры базы данных** в папке **Программирование**. В правой части окна среды появиться шаблон триггера.

Индивидуальная работа

1. Создайте триггеры для таблиц проектируемой БД данных, используя окно редактора запросов, и проверьте их работу.

a. триггер на добавление записи в одну из таблиц БД с выводом сообщения об этом событии;

b. триггер, запускаемый при занесении новой строки в одну из таблиц БД. Триггер должен увеличивать счетчик числа добавленных строк;

c. триггер, запускаемый при удалении записи в родительской таблице и запрещающий ее удаление, если есть связанные с ней записи в дочерней таблице;

d. триггер, запрещающий ввод записи в дочернюю таблицу, если значение поля внешнего ключа не совпадает ни с одним значением первичного ключа родительской таблицы. Обеспечить вывод сообщения об этом событии;

e. создайте триггер, который при вводе записи в таблицу, имеющую вычисляемое поле, вычисляет это поле (если такой таблицы нет, то согласуйте задание с преподавателем).

2. Получить список триггеров, определенных для конкретной (заданной) таблицы БД. Дать комментарии по возвращаемому результату.

3. Создание триггера DDL и его тестирование.

a. Создайте в проектируемой БД таблицу Test, содержащую один столбец с именем ID. Тип данных столбца – целые числа; неопределенные значения в столбце не допустимы.

b. Введите в таблицу 2-3 записи, используя оператор INSERT.

c. Создайте триггер DDL:

```
CREATE TRIGGER ddl_drop_table_test
ON DATABASE
FOR DROP_TABLE
AS
PRINT 'Вы пытаетесь удалить таблицу в базе данных.'
PRINT 'Если Вы действительно хотите удалить таблицу, то отключите триггер DDL.'
PRINT 'После того, как таблица будет удалена, необходимо вновь включить триггер'
ROLLBACK TRANSACTION
```

d. Создайте и запустите на исполнение запрос на удаление таблицы Test.

e. Самостоятельно изучите синтаксис операторов отключения и активирования триггера.

f. Выполните последовательно команды:

- отключить триггер ddl_drop_table_test;
- удалить таблицу Test;
- проверить, что удаление таблицы завершилось успешно, создав запрос на выборку из таблицы Test;
- активировать триггер ddl_drop_table_test;
- проверить работу триггера при попытке удаления какой-либо таблицы проектируемой Вами базы данных.

Контрольные вопросы

1. Что такое ограничения целостности?
2. Перечислите типы ограничений целостности.

3. Какие ограничения целостности можно поддержать с помощью триггеров?

4. При каких изменениях в базе данных активизируются триггеры DDL?

5. Можно ли действия, выполняемые триггером, закодировать в хранимой процедуре?

6. В чем заключаются отличия триггеров и хранимых процедур?

7. Дайте комментарии по синтаксису оператора CREATE TRIGGER.

8. Каково назначение таблиц INSERTED и DELETED?

9. Какая системная процедура позволяют получить код триггера?

10. Как узнать от каких объектов базы данных зависит триггер?

11. Как получить список триггеров конкретной таблицы?