Учреждение образования “БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ”

Лабораторная работа № 14. Применение триггеров

Выполнил:

студент 2 курса, 7 группы

Бобрович Глеб Сергеевич

Минск 2022ПроТТеоре

|  |  |
| --- | --- |
| **Задание** |  |

|  |  |
| --- | --- |
| 1.С помощью сценария, представленного на рисунке, создать таблицу **TR\_AUDIT**.    Таблица предназначена для добавления в нее строк триггерами.  В столбец **STMT** триггер должен поместить событие, на которое он среагировал, а в столбец **TRNAME −** собственное имя.  Разработать AFTER-триггер с именем **TR\_TEACHER\_INS** для таблицы **TEACHER**, реагирующий на событие **INSERT**. Триггер должен записывать строки вводимых данных в таблицу **TR\_AUDIT**. В столбец **СС** помещаются значения столбцов вводимой строки.  2.Создать AFTER-триггер с именем **TR\_TEACHER\_DEL** для таблицы **TEA-CHER**, реагирующий на событие **DELETE**. Триггер должен записывать строку данных в таблицу **TR\_AUDIT** для каждой удаляемой строки. В столбец **СС** помещаются значения столбца **TEACHER** удаляемой строки.  3.Создать AFTER-триггер с именем **TR\_TEACHER\_UPD** для таблицы **TEA-CHER**, реагирующий на событие **UPDATE**. Триггер должен записывать строку данных в таблицу **TR\_AUDIT** для каждой изменяемой строки. В столбец **СС** помещаются значения столбцов изменяемой строки до и после изменения. | **Задание 1**  USE ЛР14;  go  create trigger TR on AUDITORIUM\_TYPE instead of INSERT  as  begin  DECLARE curs CURSOR local for SELECT AUDITORIUM\_TYPENAME from AUDITORIUM\_TYPE;  declare @numb nvarchar(30), @aud nvarchar(30), @count int, @type nvarchar(5);  set @count = 0;  OPEN curs;  fetch curs into @aud;  while @@fetch\_status = 0  begin  set @numb = (select AUDITORIUM\_TYPENAME from INSERTED);  set @type = (select AUDITORIUM\_TYPE from INSERTED);  if(@numb = @aud)  set @count = @count + 1;  if(@count > 1)  begin  raiserror('Нельзя вставить повторяющееся значение',10,1);  rollback;  --delete from AUDITORIUM\_TYPE where AUDITORIUM\_TYPE=@type;  end;  fetch curs into @aud;  end;  CLOSE curs;  end;  return;  --открыть транакцию,комит и ролбэк  insert into AUDITORIUM\_TYPE(AUDITORIUM\_TYPE,AUDITORIUM\_TYPENAME) values('ЛК\_17','Лекционная2');  --Разработать AFTER-триггер для таблицы TEACHER, реагир на событие INSERT  --запис. строки вводимых данных в таблицу TR\_AUDIT  --В столбец СС помещаются значения столбцов вводимой строки.  drop table TR\_AUDIT  go  create table TR\_AUDIT  (  ID int identity,  STMT varchar(20)  check (STMT in ('INS', 'DEL', 'UPD')),  TRNAME varchar(50),  CC varchar(300)  )  drop trigger TR\_TEACHER\_INS  go  create trigger TR\_TEACHER\_INS  on TEACHER after INSERT  as  declare @a1 char(10), @a2 varchar(100), @a3 char(1), @a4 char(20), @in varchar(300);  print 'Вставка';  set @a1 = (select TEACHER from INSERTED);  set @a2= (select TEACHER\_NAME from INSERTED);  set @a3= (select GENDER from INSERTED);  set @a4 = (select PULPIT from INSERTED);  set @in = @a1+' '+ @a2 +' '+ @a3+ ' ' +@a4;  insert into TR\_AUDIT(STMT, TRNAME, CC)  values('INS', 'TR\_TEACHER\_INS', @in);  return;  go  insert into TEACHER values('ИВНВ', 'Иванов', 'м', 'ИСиТ');  select \* from TR\_AUDIT  --delete from TEACHER where TEACHER='ИВНВ';    **Задание 2**  USE ЛР14;  go  create trigger TR\_TEACHER\_DEL  on TEACHER after DELETE  as  declare @a1 char(10), @a2 varchar(100), @a3 char(1), @a4 char(20), @in varchar(300);  print 'Удаление';  set @a1 = (select TEACHER from DELETED);  set @a2= (select TEACHER\_NAME from DELETED);  set @a3= (select GENDER from DELETED);  set @a4 = (select PULPIT from DELETED);  set @in = @a1+' '+ @a2 +' '+ @a3+ ' ' +@a4;  insert into TR\_AUDIT(STMT, TRNAME, CC)  values('DEL', 'TR\_TEACHER\_DEL', @in);  return;  go  delete TEACHER where TEACHER='ИВНВ'  select \* from TR\_AUDIT    **Задание 3**  USE ЛР14;  go  alter trigger TR\_TEACHER\_DEL  on TEACHER after UPDATE  as  declare @a1 char(10), @a2 varchar(100), @a3 char(1), @a4 char(20), @in varchar(300);  declare @ins int = (select count(\*) from inserted),  @del int = (select count(\*) from deleted);  print 'Обновление';  set @a1 = (select TEACHER from INSERTED);  set @a2= (select TEACHER\_NAME from INSERTED);  set @a3= (select GENDER from INSERTED);  set @a4 = (select PULPIT from INSERTED);  set @in = @a1+' '+ @a2 +' '+ @a3+ ' ' +@a4;  set @a1 = (select TEACHER from deleted);  set @a2= (select TEACHER\_NAME from DELETED);  set @a3= (select GENDER from DELETED);  set @a4 = (select PULPIT from DELETED);  set @in =@in + '' + @a1+' '+ @a2 +' '+ @a3+ ' ' +@a4;  insert into TR\_AUDIT(STMT, TRNAME, CC)  values('UPD', 'TR\_TEACHER\_UPD', @in);  return;  go  update TEACHER set GENDER = 'ж' where TEACHER='Кирд'  select \* from TR\_AUDIT  delete from TR\_AUDIT where STMT = 'UPD' |
| 4.Создать AFTER-триггер с именем **TR\_TEACHER** для таблицы **TEACHER**, реагирующий на события **INSERT**, **DELETE**, **UPDATE**.  Триггер должен записывать строку данных в таблицу **TR\_AUDIT** для каждой изменяемой строки. В коде триггера определить событие, активизировавшее триггер и поместить в столбец **СС** соответствующую событию информацию.  Разработать сценарий, демонстрирующий работоспособность триггера. | **Задание 4**  USE ЛР14;  go  create trigger TR\_TEACHER on TEACHER after INSERT, DELETE, UPDATE  as declare @a1 char(10), @a2 varchar(100), @a3 char(1), @a4 char(20), @in varchar(300);  declare @ins int = (select count(\*) from inserted),  @del int = (select count(\*) from deleted);  if @ins > 0 and @del = 0  begin  print 'Событие: INSERT';  set @a1 = (select TEACHER from INSERTED);  set @a2= (select TEACHER\_NAME from INSERTED);  set @a3= (select GENDER from INSERTED);  set @a4 = (select PULPIT from INSERTED);  set @in = @a1+' '+ @a2 +' '+ @a3+ ' ' +@a4;  insert into TR\_AUDIT(STMT, TRNAME, CC)  values('INS', 'TR\_TEACHER\_INS', @in);  end;  else  if @ins = 0 and @del > 0  begin  print 'Событие: DELETE';  set @a1 = (select TEACHER from DELETED);  set @a2= (select TEACHER\_NAME from DELETED);  set @a3= (select GENDER from DELETED);  set @a4 = (select PULPIT from DELETED);  set @in = @a1+' '+ @a2 +' '+ @a3+ ' ' +@a4;  insert into TR\_AUDIT(STMT, TRNAME, CC)  values('DEL', 'TR\_TEACHER\_DEL', @in);  end;  else  if @ins > 0 and @del > 0  begin  print 'Событие: UPDATE';  set @a1 = (select TEACHER from INSERTED);  set @a2= (select TEACHER\_NAME from INSERTED);  set @a3= (select GENDER from INSERTED);  set @a4 = (select PULPIT from INSERTED);  set @in = @a1+' '+ @a2 +' '+ @a3+ ' ' +@a4;  set @a1 = (select TEACHER from deleted);  set @a2= (select TEACHER\_NAME from DELETED);  set @a3= (select GENDER from DELETED);  set @a4 = (select PULPIT from DELETED);  set @in =@in + '' + @a1+' '+ @a2 +' '+ @a3+ ' ' +@a4;  insert into TR\_AUDIT(STMT, TRNAME, CC)  values('UPD', 'TR\_TEACHER\_UPD', @in);  end;  return;  delete TEACHER where TEACHER='ИВНВ'  insert into TEACHER values('ИВНВ', 'Иванов', 'м', 'ИСиТ');  update TEACHER set GENDER = 'ж' where TEACHER='Кирд'  select \* from TR\_AUDIT  **Задание 5**  USE ЛР14;  update TEACHER set GENDER = 'й' where TEACHER='Кирд'  select \* from TR\_AUDIT |
| 5.Разработать сценарий, который демонстрирует на примере базы данных X\_UNIVER, что проверка ограничения целостности выполняется до срабатывания AFTER-триггера. |  |
| 6.Создать для таблицы **TEACHER** три AFTER-триггера с именами: **TR\_TEACHER\_ DEL1**, **TR\_TEACHER\_DEL2** и **TR\_TEA-CHER\_ DEL3**. Триггеры должны реагировать на событие DELETE и формировать соответствующие строки в таблицу **TR\_AUDIT**. Получить список триггеров таблицы **TEACHER**.  Упорядочить выполнение триггеров для таблицы **TEACHER**, реагирующих на событие **DELETE** следующим образом: первым должен выполняться триггер с именем **TR\_TEA-CHER\_DEL3**, последним – триггер **TR\_TEACHER\_DEL2**.  Примечание: использовать системные представления **SYS.TRIGGERS** и **SYS.TRIG-GERS\_ EVENTS**, а также системную процедуру **SP\_SETTRIGGERORDERS**. | **Задание 6**  USE ЛР14;  go  create trigger AUD\_AFTER\_DEL1 on FACULTY after DELETE  as print 'AUD\_AFTER\_DEL1';  return;  go  create trigger AUD\_AFTER\_DEL2 on FACULTY after DELETE  as print 'AUD\_AFTER\_DEL2';  return;  go  create trigger AUD\_AFTER\_DEL3 on FACULTY after DELETE  as print 'AUD\_AFTER\_DEL3';  return;  go  select t.name, e.type\_desc  from sys.triggers t join sys.trigger\_events e on t.object\_id = e.object\_id  where OBJECT\_NAME(t.parent\_id)='FACULTY' and e.type\_desc = 'DELETE' ;  exec SP\_SETTRIGGERORDER @triggername = 'AUD\_AFTER\_DEL3',  @order='First', @stmttype = 'DELETE';  exec SP\_SETTRIGGERORDER @triggername = 'AUD\_AFTER\_DEL2',  @order='Last', @stmttype = 'DELETE';  select t.name, e.type\_desc  from sys.triggers t join sys.trigger\_events e on t.object\_id = e.object\_id  where OBJECT\_NAME(t.parent\_id)='FACULTY' and e.type\_desc = 'DELETE'; |
| 7.Разработать сценарий, демонстрирующий на примере базы данных X\_UNIVER утверждение: AFTER-триггер является частью транзакции, в рамках которого выполняется оператор, активизировавший триггер. | **Задание 7**  USE ЛР14;  go  create trigger PTran  on PULPIT after INSERT, DELETE, UPDATE  as declare @c int = (select count (\*) from PULPIT);  if (@c >26)  begin  raiserror('Общая количество кафедр не может быть >26', 10, 1);  rollback;  end;  return;  insert into PULPIT(PULPIT) values ('ТТПЛ') |
| 8.Для таблицы **FACULTY** создать **INSTEAD OF**-триггер, запрещающий удаление строк в таблице.  Разработать сценарий, который демонстрирует на примере базы данных X\_UNIVER, что проверка ограничения целостности выполнена, если есть INSTEAD OF-триггер.  С помощью оператора DROP удалить все DML-триггеры, созданные в этой лабораторной работе. | **Задание 8**  USE ЛР14;  go  create trigger F\_INSTEAD\_OF  on FACULTY instead of DELETE  as  raiserror(N'Удаление запрещено', 10, 1);  return;  delete FACULTY where FACULTY = 'ИДиП'  drop trigger F\_INSTEAD\_OF  drop trigger PTran  drop trigger TR\_TEACHER  drop trigger TR\_TEACHER\_DEL  go |