**Лабораторная работа № 14. Применение триггеров**

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.

CREATE TABLE TR\_AUDIT

(

ID int identity,

STMT varchar(20) check (STMT in ('INS','DEL','UPD')),

TRNAME varchar(50),

CC varchar(300)

);

GO

CREATE TRIGGER TR\_TEACHER\_INS on TEACHER after INSERT AS

insert into TR\_AUDIT values (

'INS',

'TR\_TEACHER\_INS',

'Преподаватель:' + rtrim((SELECT TEACHER from INSERTED)) +

'. Имя:' + rtrim((SELECT TEACHER\_NAME from INSERTED)) +

'. Пол:' + rtrim((SELECT GENDER from INSERTED)) +

'. Кафедра: ' + rtrim((SELECT PULPIT from INSERTED))

);

INSERT into TEACHER(TEACHER,TEACHER\_NAME,GENDER,PULPIT)

VALUES ('ДБРН','Добриян Александр Витальевич','м','ИСиТ');

SELECT \* from TR\_AUDIT;



1. 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.

GO

CREATE TRIGGER TR\_TEACHER\_DEL on TEACHER after DELETE AS

insert into TR\_AUDIT values (

'DEL',

'TR\_TEACHER\_DEL',

'Преподаватель:' + rtrim((SELECT TEACHER from DELETED)) +

'. Имя:' + rtrim((SELECT TEACHER\_NAME from DELETED)) +

'. Пол:' + rtrim((SELECT GENDER from DELETED)) +

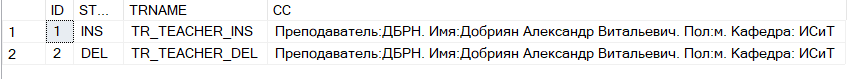
'. Кафедра: ' + rtrim((SELECT PULPIT from DELETED))

);

GO

DELETE FROM TEACHER where TEACHER = 'ДБРН';

SELECT \* from TR\_AUDIT;



1. 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.

GO

CREATE TRIGGER TR\_TEACHER\_UPD on TEACHER after UPDATE AS

insert into TR\_AUDIT values (

'UPD',

'TR\_TEACHER\_UPD',

'Преподаватель:' + rtrim((SELECT TEACHER from INSERTED)) +

'. Имя:' + rtrim((SELECT TEACHER\_NAME from INSERTED)) +

'. Пол:' + rtrim((SELECT GENDER from INSERTED)) +

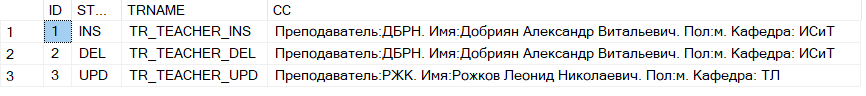
'. Кафедра: ' + rtrim((SELECT PULPIT from INSERTED))

);

GO

UPDATE TEACHER SET PULPIT = 'ТЛ' where TEACHER = 'РЖК';

SELECT \* from TR\_AUDIT;



1. 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.

GO

CREATE TRIGGER TR\_TEACHER on TEACHER after INSERT, DELETE, UPDATE AS

DECLARE @upd int = (select count(\*) from inserted),

@del int = (select count(\*) from deleted);

if @upd > 0 and @del = 0

BEGIN

insert into TR\_AUDIT values (

'INS',

'TR\_TEACHER',

'Преподаватель:' + rtrim((SELECT TEACHER from INSERTED)) +

'. Имя:' + rtrim((SELECT TEACHER\_NAME from INSERTED)) +

'. Пол:' + rtrim((SELECT GENDER from INSERTED)) +

'. Кафедра: ' + rtrim((SELECT PULPIT from INSERTED))

);

END

else if @upd = 0 and @del > 0

BEGIN

insert into TR\_AUDIT values (

'DEL',

'TR\_TEACHER',

'Преподаватель:' + rtrim((SELECT TEACHER from DELETED)) +

'. Имя:' + rtrim((SELECT TEACHER\_NAME from DELETED)) +

'. Пол:' + rtrim((SELECT GENDER from DELETED)) +

'. Кафедра: ' + rtrim((SELECT PULPIT from DELETED))

);

END

else if @upd > 0 and @del > 0

BEGIN

insert into TR\_AUDIT values (

'UPD',

'TR\_TEACHER',

'Преподаватель:' + rtrim((SELECT TEACHER from INSERTED)) +

'. Имя:' + rtrim((SELECT TEACHER\_NAME from INSERTED)) +

'. Пол:' + rtrim((SELECT GENDER from INSERTED)) +

'. Кафедра: ' + rtrim((SELECT PULPIT from INSERTED))

);

END

return;

GO

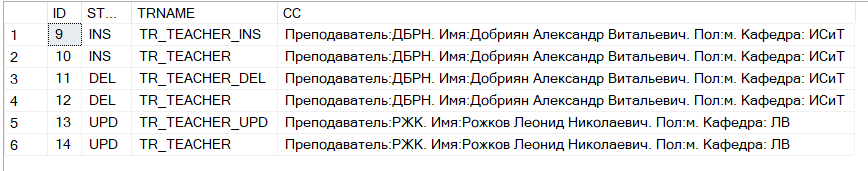
INSERT into TEACHER(TEACHER,TEACHER\_NAME,GENDER,PULPIT) VALUES

('ДБРН','Добриян Александр Витальевич','м','ИСиТ');

DELETE FROM TEACHER where TEACHER = 'ДБРН';

UPDATE TEACHER SET PULPIT = 'ЛВ' where TEACHER = 'РЖК';

SELECT \* from TR\_AUDIT;



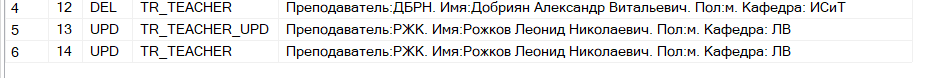
1. Разработать сценарий, который демонстрирует на примере базы данных X\_UNIVER, что проверка ограничения целостности выполняется до срабатывания AFTER-триггера.

alter table TEACHER add constraint PULPIT\_LEN check(len(TEACHER) <= 7)

UPDATE TEACHER SET TEACHER = '123456789' where TEACHER = 'РЖК';

SELECT \* from TR\_AUDIT;

alter table TEACHER drop constraint PULPIT\_LEN;



1. 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**.

GO

CREATE TRIGGER TR\_TEACHER\_DEL1 on TEACHER after DELETE AS

insert into TR\_AUDIT values (

'DEL',

'TR\_TEACHER\_DEL1',

'Преподаватель:' + rtrim((SELECT TEACHER from DELETED)) +

'. Имя:' + rtrim((SELECT TEACHER\_NAME from DELETED)) +

'. Пол:' + rtrim((SELECT GENDER from DELETED)) +

'. Кафедра: ' + rtrim((SELECT PULPIT from DELETED))

);

GO

GO

CREATE TRIGGER TR\_TEACHER\_DEL2 on TEACHER after DELETE AS

insert into TR\_AUDIT values (

'DEL',

'TR\_TEACHER\_DEL2',

'Преподаватель:' + rtrim((SELECT TEACHER from DELETED)) +

'. Имя:' + rtrim((SELECT TEACHER\_NAME from DELETED)) +

'. Пол:' + rtrim((SELECT GENDER from DELETED)) +

'. Кафедра: ' + rtrim((SELECT PULPIT from DELETED))

);

GO

GO

CREATE TRIGGER TR\_TEACHER\_DEL3 on TEACHER after DELETE AS

insert into TR\_AUDIT values (

'DEL',

'TR\_TEACHER\_DEL3',

'Преподаватель:' + rtrim((SELECT TEACHER from DELETED)) +

'. Имя:' + rtrim((SELECT TEACHER\_NAME from DELETED)) +

'. Пол:' + rtrim((SELECT GENDER from DELETED)) +

'. Кафедра: ' + rtrim((SELECT PULPIT from DELETED))

);

GO

exec SP\_SETTRIGGERORDER @triggername = 'TEACHER.TR\_TEACHER\_DEL3',

@order = 'First', @stmttype = 'DELETE';

exec SP\_SETTRIGGERORDER @triggername = 'TEACHER.TR\_TEACHER\_DEL2',

@order = 'Last', @stmttype = 'DELETE';

select t.name, e.type\_desc

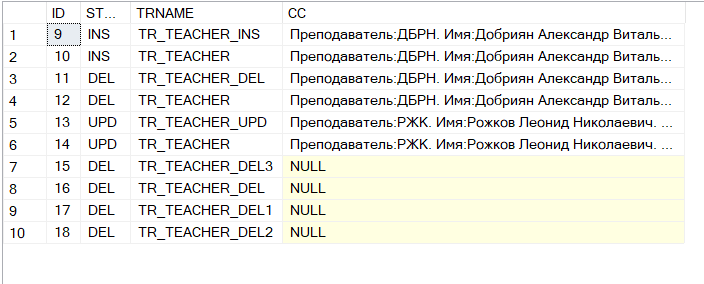
from sys.triggers t

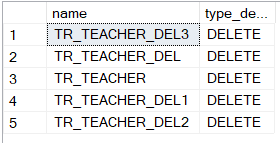
join sys.trigger\_events e on t.object\_id = e.object\_id

where OBJECT\_NAME(t.parent\_id) = 'TEACHER' and e.type\_desc = 'DELETE';

DELETE FROM TEACHER where TEACHER = 'ДБРН';

SELECT \* from TR\_AUDIT;





1. Разработать сценарий, демонстрирующий на примере базы данных X\_UNIVER утверждение: AFTER-триггер является частью транзакции, в рамках которого выполняется оператор, активизировавший триггер.

use UNIVER;

go

create trigger TEACHER\_TRAN on TEACHER after INSERT, DELETE, UPDATE AS

declare @count int = (select count(\*) from TEACHER);

begin try

if (@count >= 9)

begin

raiserror('Количество преподавателей не больше 9!', 11, 1);

rollback;

end;

end try

begin catch

print 'Error number: ' + cast(ERROR\_NUMBER() as varchar(6));

print 'Error message: ' + ERROR\_MESSAGE();

print 'Error line: ' + cast(ERROR\_LINE()as varchar(8));

print 'Error secerity: ' + cast(ERROR\_SEVERITY()as varchar(6));

print 'Error state: ' + cast(ERROR\_STATE()as varchar(8));

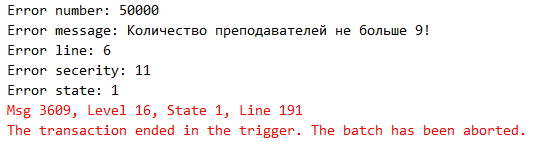
rollback;

end catch

return;

INSERT into TEACHER(TEACHER,TEACHER\_NAME,GENDER,PULPIT)

VALUES ('ДБРН','Добриян Александр Витальевич','м','ИСиТ');



1. 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.

use UNIVER;

set nocount on;

GO

create trigger FACULTY\_INSTEAD\_OF on FACULTY instead of DELETE AS

PRINT 'Instead of delete';

return;

delete from FACULTY where FACULTY ='ИТ';



1. Создать DDL-триггер, реагирующий на все DDL-события в БД **UNIVER**. Триггер должен запрещать создавать новые таблицы и удалять существующие. Свое выполнение триггер должен сопровождать сообщением, которое содержит: тип события, имя и тип объекта, а также пояснительный текст, в случае запрещения выполнения оператора. Разработать сценарий, демонстрирующий работу триггера.

GO

create trigger DDL\_UNIVER on database for DDL\_DATABASE\_LEVEL\_EVENTS AS

declare @t varchar(50) = EVENTDATA().value('(/EVENT\_INS-TANCE/EventType)[1]', 'varchar(50)');

declare @t1 varchar(50) = EVENTDATA().value('(/EVENT\_INS-TANCE/ObjectName)[1]', 'varchar(50)');

declare @t2 varchar(50) = EVENTDATA().value('(/EVENT\_INS-TANCE/ObjectType)[1]', 'varchar(50)');

print 'Тип события: '+@t;

print 'Имя объекта: '+@t1;

print 'Тип объекта: '+@t2;

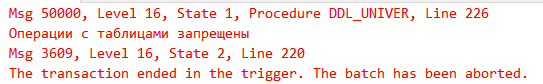
raiserror( N'Операции с таблицами запрещены ', 16, 1);

rollback;

CREATE TABLE BUBUB (

ID int identity(1,1)

)



10. Разработать различные виды триггеров для базы данных **X\_MyBASE** и продемонстрировать их работу.

-- #1

use Dobriyan\_MyBase;

CREATE TABLE TR\_SPEC

(

ID int identity,

STMT varchar(20) check (STMT in ('INS','DEL','UPD')),

TRNAME varchar(50),

CC varchar(300)

);

GO

CREATE TRIGGER TR\_SPEC\_INS on Specialities after INSERT AS

insert into TR\_SPEC values (

'INS',

'TR\_SPEC\_INS',

'ID:' + rtrim((SELECT id from INSERTED)) +

'. Специальность:' + rtrim((SELECT Speciality from INSERTED))

);

INSERT into Specialities VALUES (15, 'test1');

SELECT \* from TR\_SPEC;

DROP TRIGGER TR\_SPEC\_INS;

-- #2

use Dobriyan\_MyBase;

GO

CREATE TRIGGER TR\_SPEC\_DEL on Specialities after DELETE AS

insert into TR\_SPEC values (

'DEL',

'TR\_SPEC\_DEL',

'ID:' + rtrim((SELECT id from deleted)) +

'. Специальность:' + rtrim((SELECT Speciality from deleted))

);

DELETE Specialities where Speciality = 'test1';

SELECT \* from TR\_SPEC;

DROP TRIGGER TR\_SPEC\_DEL;

-- #3

use Dobriyan\_MyBase;

GO

CREATE TRIGGER TR\_SPEC\_UPD on Specialities after UPDATE AS

insert into TR\_SPEC values (

'UPD',

'TR\_SPEC\_UPD',

'ID:' + rtrim((SELECT id from inserted)) +

'. Специальность:' + rtrim((SELECT Speciality from inserted))

);

UPDATE Specialities set id = id + 1 where Speciality = 'test1';

SELECT \* from TR\_SPEC;

DROP TRIGGER TR\_SPEC\_UPD;

-- #4

use Dobriyan\_MyBase;

GO

CREATE TRIGGER TRG\_SPEC on Specialities after INSERT, DELETE, UPDATE AS

DECLARE @upd int = (select count(\*) from inserted),

@del int = (select count(\*) from deleted);

if @upd > 0 and @del = 0

BEGIN

insert into TR\_SPEC values (

'INS',

'TRG\_SPEC',

'ID:' + rtrim((SELECT id from INSERTED)) +

'. Специальность:' + rtrim((SELECT Speciality from INSERTED))

);

END

else if @upd = 0 and @del > 0

BEGIN

insert into TR\_SPEC values (

'DEL',

'TRG\_SPEC',

'ID:' + rtrim((SELECT id from deleted)) +

'. Специальность:' + rtrim((SELECT Speciality from deleted))

);

END

else if @upd > 0 and @del > 0

BEGIN

insert into TR\_SPEC values (

'UPD',

'TRG\_SPEC',

'ID:' + rtrim((SELECT id from inserted)) +

'. Специальность:' + rtrim((SELECT Speciality from inserted))

);

END

return;

GO

DELETE Specialities where Speciality = 'test1';

INSERT into Specialities VALUES (17, 'test1');

UPDATE Specialities set id = id + 1 where Speciality = 'test1';

SELECT \* from TR\_SPEC;

DROP TRIGGER TRG\_SPEC;