# Ćwiczenia

*Wyzwalacze.*

# Ćwiczenia należy wykonać w bazie danych Northwind (lub w jej kopii).

# Zadanie 1

Napisz wyzwalacz uniemożliwiający dopisanie rekordu do tabeli Categories (użyj RAISERROR i ROLLBACK TRANSACTION)

1. Wersja AFTER INSERT

create trigger z1a

on Categories

after insert

as

rollback;

throw 51000, 'Error', 1;

1. Wersja INSTEAD OF INSERT

create trigger z1b

on Categories

instead of insert

as

throw 51000, 'Error', 1;

# Zadanie 2

Utwórz w bazie danych Northwind tabelę Products\_log (zawierającą kolumny Nr INT IDENTITY(1,1), Data DATETIME oraz Uwagi NVARCHAR(255)) i napisz wyzwalacz, który będzie do niej dodawał jeden rekord przy każdej aktualizacji tabeli products.

CREATE TABLE Products\_log

(

Nr INT IDENTITY(1,1) PRIMARY KEY,

Data DATETIME,

uwagi NVARCHAR(255)

)

GO

create trigger z2t

on Products

after insert, update

as

begin

insert into ProductsLog (Data, Uwagi)

values (getdate(), suser\_sname());

end;

# Zadanie 3

Zmodyfikuj wyzwalacz z zadania 2 tak, aby reagował tylko w przypadku zmiany ceny.

create trigger z3t

on Products

after update

as

begin

if (select UnitPrice from inserted) != (select UnitPrice from deleted)

begin

insert into ProductsLog (Data, Uwagi)

values (getdate(), suser\_sname());

end;

end;

# Zadanie 4

Zmodyfikuj wyzwalacz oraz tabelę Products\_log tak, aby zapamiętywana była informacja o starej cenie (tylko o starej cenie, nowej nie).

alter table ProductsLog

add PoprzedniaCena money;

go

create trigger z4t

on Products

after update

as

begin

declare @OldPrice money;

select @OldPrice = UnitPrice from deleted;

if (select UnitPrice from inserted) != @OldPrice

begin

insert into ProductsLog (Data, Uwagi, PoprzedniaCena)

values (getdate(), suser\_sname(), @OldPrice);

end;

end;

# Zadanie 5

Zmodyfikuj wyzwalacz oraz tabelę Products\_log tak, aby zapamiętywana była informacja o starej cenie i nowej cenie.

alter table ProductsLog

add NowaCena money;

go

create trigger z5t

on Products

after update

as

begin

declare @OldPrice money, @NewPrice money;

select @OldPrice = UnitPrice from deleted;

select @NewPrice = UnitPrice from inserted;

if @NewPrice != @OldPrice

begin

insert into ProductsLog (Data, Uwagi, PoprzedniaCena, NowaCena)

values (getdate(), suser\_sname(), @OldPrice, @NewPrice);

end;

# end;

# Zadanie 6

Napisz wyzwalacz, który po dopisaniu wierszy do tabeli [Order Details] wstawi poprawną cenę do tych wierszy.

Wskazówka: Należy napisać wyzwalacz typu INSTEAD OF, który zamiast dopisania wierszy do [Order Details] dopisze dane z wykorzystaniem złączenia tabel INSERTED i Products. Przy testach można dopisywać nowe produkty np. do zamówienia nr 10248.

create trigger z6t

on [Order Details]

instead of insert

as

begin

declare @orderID int;

select @orderID = OrderID from inserted;

if exists (select 1 from Orders where OrderID = @orderID)

begin

insert into [Order Details]

select i.OrderID, i.ProductID, p.UnitPrice, i.Quantity, i.Discount

from inserted as i

inner join Products as p

on i.ProductID = p.ProductID

end;

# end;

# Zadanie 7

Utwórz kopię tabeli Products, nazwij ją Product2 i dodaj do niej kolumny Login typu NVARCHAR(128) oraz LastModified typu DATETIME.

SELECT \* INTO Products2 FROM Products

ALTER TABLE Products2 ADD PRIMARY KEY (ProductId)

ALTER TABLE Products

ADD Login NVARCHAR(128), LastModified DATETIME

Proszę napisać wyzwalacz, który automatycznie wpisze odpowiednie dane do kolumn Login i LastModified w tabeli. Wyzwalacz ma działać po aktualizacji (rekordów) w tabeli Products2 i ma do kolumny Login wpisać login osoby, która przeprowadziła aktualizację rekordów w tabeli (można to uzyskać wykorzystując funkcję SUSER\_SNAME()) a do kolumny LastModified wpisać datę i godzinę, kiedy modyfikacja została wykonana (można wykorzystać funkcję GETDATE()).

create trigger z7t

on Products2

after update

as

begin

update Products2

set Login = suser\_sname(),

Modified = getdate()

where ProductID = (select ProductID from inserted);

end;

# Zadanie 8

Proszę napisać wyzwalacz, który zmodyfikuje pole UnitsInStock w tabeli Products po dopisaniu wiersza (wierszy) do tabeli [Order Details]. Mamy zatem modyfikować ilość towaru w magazynie, na podstawie danych z zamówienia.

Jednocześnie wyzwalacz ma wycofać transakcję, gdyby wśród nowo dopisanych zdaniem INSERT wierszy w [Order Details] w kolumnie Quantity była wartość większa niż liczba sztuk w magazynie (czyli UnitsInStock).

create trigger z8t

on [Order Details]

after insert

as

begin

declare @soldQty smallint, @productID int, @availableQty smallint;

select @soldQty = Quantity from inserted;

select @productID = ProductID from inserted;

select @availableQty = UnitsInStock from Products where ProductID = @productID;

if @soldQty > @availableQty

begin

rollback;

throw 51000, 'Error!!!', 1;

end;

else

begin

update Products

set UnitsInStock = @availableQty - @soldQty

where ProductID = @productID;

end;

end;