Blaski i cienie wyzwalaczy w relacyjnych bazach danych.

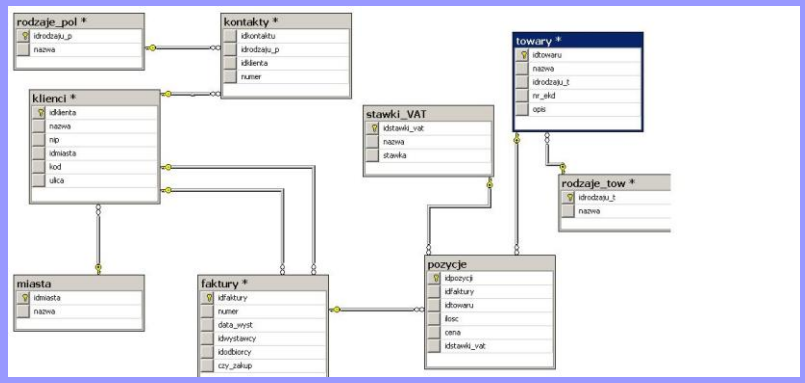
Mgr inż. Andrzej Ptasznik

Technologia

Przykłady praktycznych zastosowań wyzwalaczy będą omawiane na bazie systemu MS SQL Server 2005

Wprowadzenie

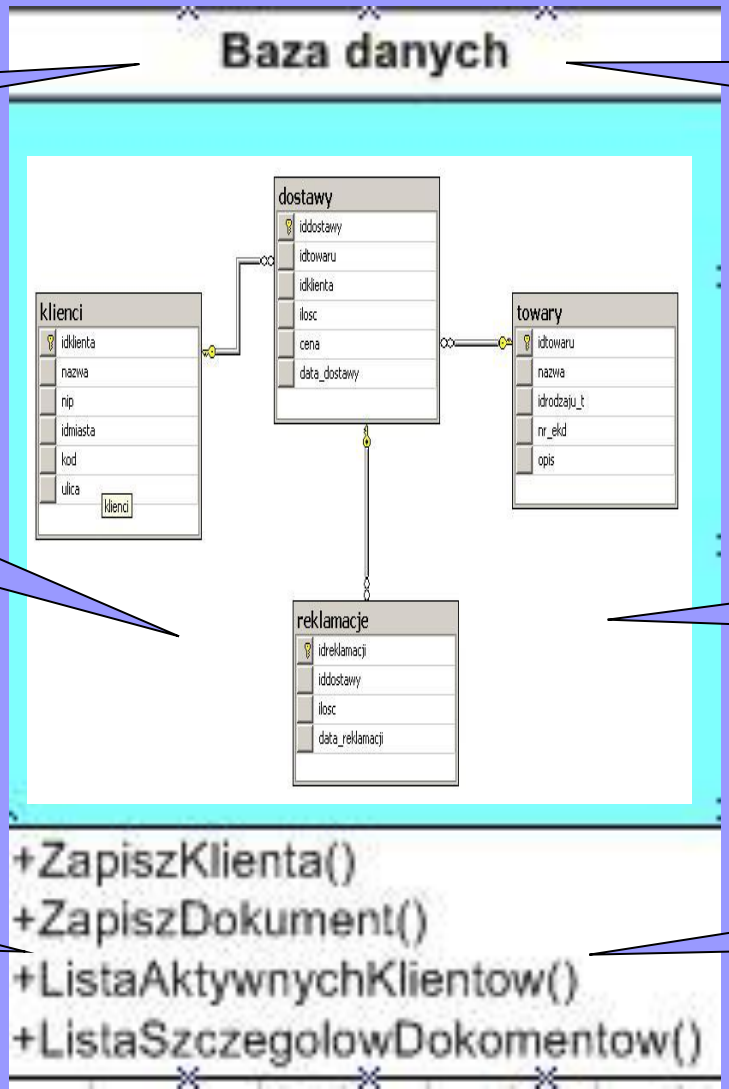
- Obiekty bazy danych
  - Tabele
  - Widoki
  - Reguły i ograniczenia
    - Typ danych
    - Ograniczenie CHECK
    - Ograniczenie PRIMARY KEY
    - Ograniczenie FOREGIN KEY
  - Procedury składowane
  - Funkcja składowane
  - Wyzwalacze



Obiektowe spojrzenie na bazę danych

Baza danych

Obiekt „BazaDanych”



Model logiczny bazy danych

Atrybuty obiektu

Procedury składowane

Metody obiektu

Wyzwalacze

Wyzwalacz, to specjalny typ procedury składowanej, która jest wykonywana automatycznie, jako reakcja na zdarzenie.

```
ALTER TRIGGER [TR_zmiany_regon]
ON [dbo].[numery_identyfikacyjne]
AFTER update,insert,delete
AS

declare @n_nip varchar(50),
        @s_nip varchar(50),
        @idelementu_ident int ,
        @idelementu_nip int,
        @idelementu_regon int,
        @idkontrahenta int,
        @idpodmiotu int,
        @numer_new varchar(50),
        @numer_old varchar(50),
        @ii int,
        @id int,
        @operacja char(1)

select @ii=count(*) from inserted
select @id=count(*) from deleted

if @ii=0 and @id>0
  set @operacja='D'
else
    if @id=0
       set @operacja='I'
    else
       set @operacja='U'
```

Wyzwalacze

Wyzwalacze można wykorzystać między innymi do :

- złożonych warunków poprawności dla kolumny lub wiersza
- implementacji złożonych zachowań dla integralności referencyjnej
- Obliczania wartości kolumn
- Inspekcji działania użytkowników

Wyzwalacze

Wyzwalacze mogą wykonywać następujące czynności:

- porównywać wersje danych z „przed” i „po ” modyfikacji
- anulować nieprawidłowe modyfikacje
- odczytywać dane z innych tabel
- modyfikować inne tabele
- wykonywać procedury składowane

Rodzaje wyzwalaczy

- Wyzwalacze DML (**Data Manipulation Language**)

  Reagują na „zdarzenia” Insert, Update, Delete

- Wyzwalacze DDL (Data Definition Language)

  Reagują na „zdarzenia” CREATE,ALTER,DROP

Wyzwalacze DML

- Rodzaje wyzwalaczy DML
  - Typ AFTER (po operacji DML)
  - Typ BEFORE (przed operacją DML)
  - Typ INSTEAD OF (zamiast operacji DML)

Wyzwalacz jest tworzony dla tabeli i jest związany z jedną (bądź więcej) instrukcji modyfikujących dane(INSERT, UPDATE lub DELETE)

Cechy wyzwalaczy

- Wyzwalacz jest wykonywany tylko raz dla danego polecenia niezależnie od tego ile wierszy modyfikuje te polecenie.
- Wyzwalacz jest integralna częścią transakcji związanej z realizacją polecenia
- Jawne polecenie ROLLBACK usuwa skutki działania wyzwalacza oraz polecenia które go wywołało.

Schemat działania wyzwalaczy „AFTER”

Rozpoczęcie transakcji niejawnej (BEGIN TRANSACTION)

Polecenia DML (Insert |Update | Delete)

Realizacja polecenia DML (sprawdzanie reguł integralności)

```
ALTER TRIGGER [TR_zmiany_regon]
ON [dbo].[numery_identyfikacyjne]
AFTER update,insert,delete
AS

declare @n_nip varchar(50),
        @s_nip varchar(50),
        @idelementu_ident int ,
        @idelementu_nip int,
        @idelementu_regon int,
        @idkontrahenta int,
        @idpodmiotu int,
        @numer_new varchar(50),
        @numer_old varchar(50),
        @ii int,
        @id int,
        @operacja char(1)

select @ii=count(*) from inserted
select @id=count(*) from deleted

if @ii=0 and @id>0
  set @operacja='D'
else
     if @id=0
        set @operacja='I'
     else
        set @operacja='U'
```

Kod wyzwalacza

Zatwierdzenie transakcji niejawnej (COMMIT TRANSACTION)

Schemat działania wyzwalaczy „BEFORE”

Rozpoczęcie transakcji niejawnej (BEGIN TRANSACTION)

Polecenia DML (Insert |Update | Delete)

```
ALTER TRIGGER [TR_zmiany_regon]
ON [dbo].[numery_identyfikacyjne]
AFTER update,insert,delete
AS

declare @n_nip varchar(50),
        @s_nip varchar(50),
        @idelementu_ident int ,
        @idelementu_nip int,
        @idelementu_regon int,
        @idkontrahenta int,
        @idpodmiotu int,
        @numer_new varchar(50),
        @numer_old varchar(50),
        @ii int,
        @id int,
        @operacja char(1)

select @ii=count(*) from inserted
select @id=count(*) from deleted

if @ii=0 and @id>0
  set @operacja='D'
else
     if @id=0
        set @operacja='I'
     else
        set @operacja='U'
```

Kod wyzwalacza

Realizacja polecenia DML (sprawdzanie reguł integralności)

Zatwierdzenie transakcji niejawnej (COMMIT TRANSACTION)

Schemat działania wyzwalaczy „INSTEAD OF”

Rozpoczęcie transakcji niejawnej (BEGIN TRANSACTION)

Polecenia DML (Insert |Update | Delete)

```
ALTER TRIGGER [TR_zmiany_regon]
ON [dbo].[numery_identyfikacyjne]
AFTER update,insert,delete
AS

declare @n_nip varchar(50),
        @s_nip varchar(50),
        @idelementu_ident int ,
        @idelementu_nip int,
        @idelementu_regon int,
        @idkontrahenta int,
        @idpodmiotu int,
        @numer_new varchar(50),
        @numer_old varchar(50),
        @ii int,
        @id int,
        @operacja char(1)

select @ii=count(*) from inserted
select @id=count(*) from deleted

if @ii=0 and @id>0
  set @operacja='D'
else
     if @id=0
        set @operacja='I'
     else
        set @operacja='U'
```

Kod wyzwalacza

Zatwierdzenie transakcji niejawnej (COMMIT TRANSACTION)

Wyzwalacze DML

W trakcie działania wyzwalacza jest dostęp do dwóch pseudotabel o nazwach Inserted i Deleted, które udostępniają te wiersze które zostały zmodyfikowane przez operacje, która jest związana z uruchomionym wyzwalaczem.

Pseudo tabele Inserted i Deleted

Wyrażenie	Zawartość tabeli Inserted	Zawartość tabeli Deleted
INSERT	Dodane wiersze	Pusta
UPDATE	Wiersze po modyfikacji	Wiersze przed modyfikacją
DELETE	Pusta	Usunięte wiersze

Wyzwalacze DDL

Wyzwalacze działające w kontekście bazy danych

Wyzwalacze działające w kontekście serwera

Wyzwalacze typu DDL

- Nowy typ wyzwalaczy dla języka definiowania danych
- Duże możliwości przy zabezpieczaniu bazy danych przed przypadkowymi lub niedopuszczalnymi modyfikacjami

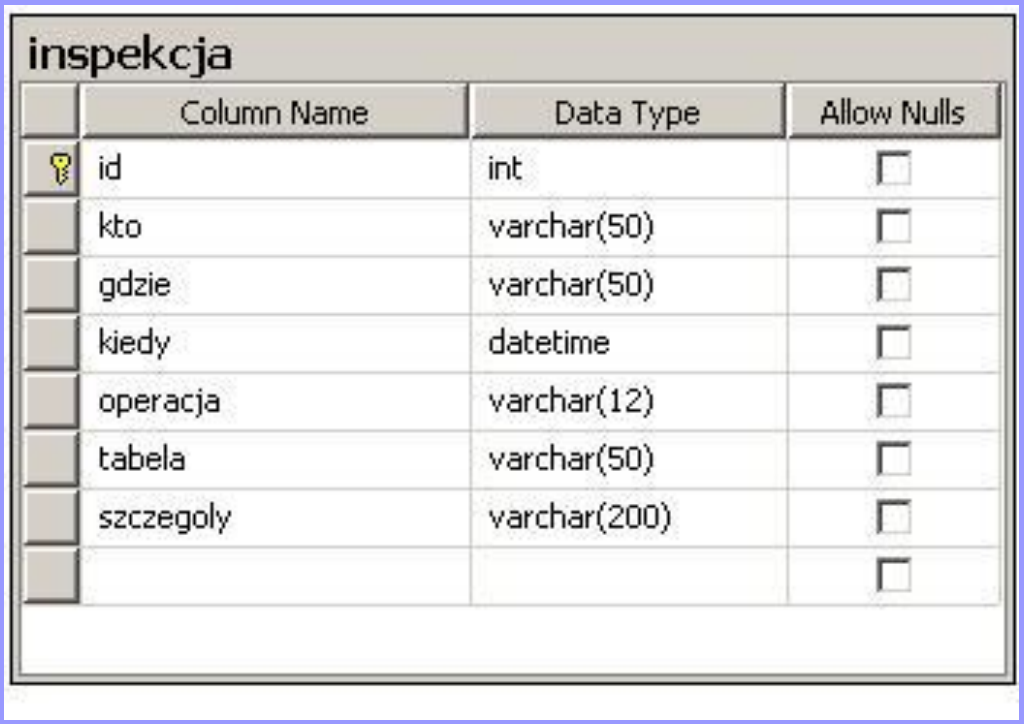
Dane dla wyzwalaczy DDL

```
<EVENT_INSTANCE>
  <EventType>CREATE_TABLE</EventType>
  <PostTime>2007-04-23</PostTime>
<SPID>34</SPID>
<ServerName>nt-14</ServerName>
<LoginName>aptasznik</LoginName>
<UserName>dbo</UserName>
<DatabaseName>testowa</DatabaseName>
<SchemaName>dbo</SchemaName>
<ObjectName>Klienci</ObjectName>
<ObjectType>ala</ObjectType>
<TSQLCommand>
<SetOptions ANSI_NULLS="ON"  ENCRYPTED="FALSE" />
<CommandText>create table ala(id INT NOT NULL PRIMARY
    KEY)</CommandText>
</TSQLCommand>
</EVENT_INSTANCE>
```

Przykłady realizacji poszczególnych typów wyzwalaczy

- Wyzwalacz DML (AFTER) realizujący inspekcję działań użytkowników.

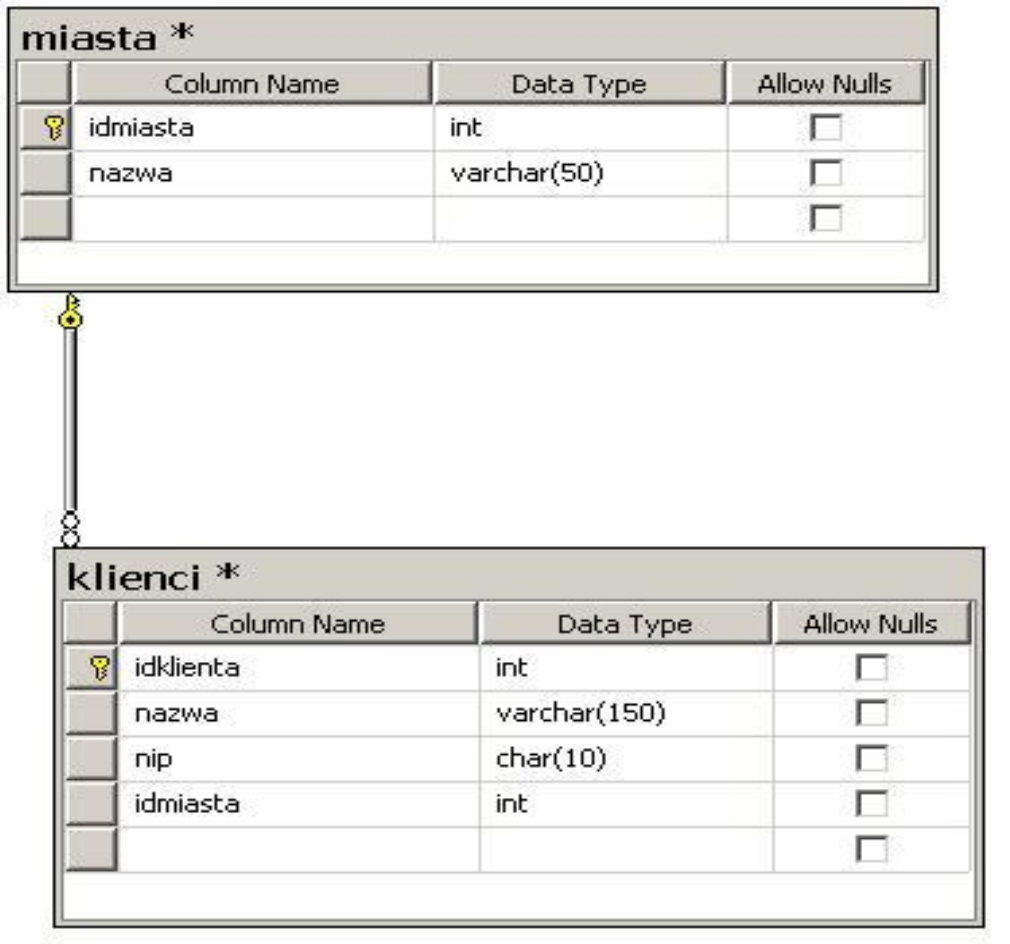
Tabela INSPEKCJA



inspekcja

Column Name	Data Type	Allow Nulls
id	int	☐
kto	varchar(50)	☐
gdzie	varchar(50)	☐
kiedy	datetime	☐
operacja	varchar(12)	☐
tabela	varchar(50)	☐
szczegoly	varchar(200)	☐
		☐

PRZYKŁAD 1 –Schemat bazy



Definicje wyzwalaczy

```
CREATE TRIGGER [TR_miasta_insert]
   ON  [dbo].[miasta] AFTER  INSERT,DELETE,UPDATE
AS
BEGIN
          declare @operacja varchar(12),@szczegoly varchar(200)
     if not exists (select * from inserted)
        set @operacja='DELETE'
     else
        if not exists (select * from deleted)
          set @operacja='INSERT'
        else
          set @operacja='UPDATE'
    if @operacja='INSERT'
      set @szczegoly='Wstawiono wiersz  o idmiasta='+
        cast((select idmiasta from inserted) as varchar(12))
   if @operacja='DELETE'
      set @szczegoly='Usuniêto wiersze o idmiasta='+
        cast((select idmiasta from deleted) as varchar(12))
   if @operacja ='UPDATE'
      begin
        if UPDATE(nazwa)
          set @szczegoly='IDMIASTA='+
              cast((select idmiasta from inserted) as varchar(12))+
             'Zmieniono nazwê z '+
             (select nazwa from deleted)  +' na ' +
             (select nazwa from inserted)
      end
insert into inspekcja(kto,gdzie,kiedy,operacja,tabela,szczegoly)
  values (system_user , host_name(), getdate(),
         @operacja, 'Miasta', @szczegoly)
END
```

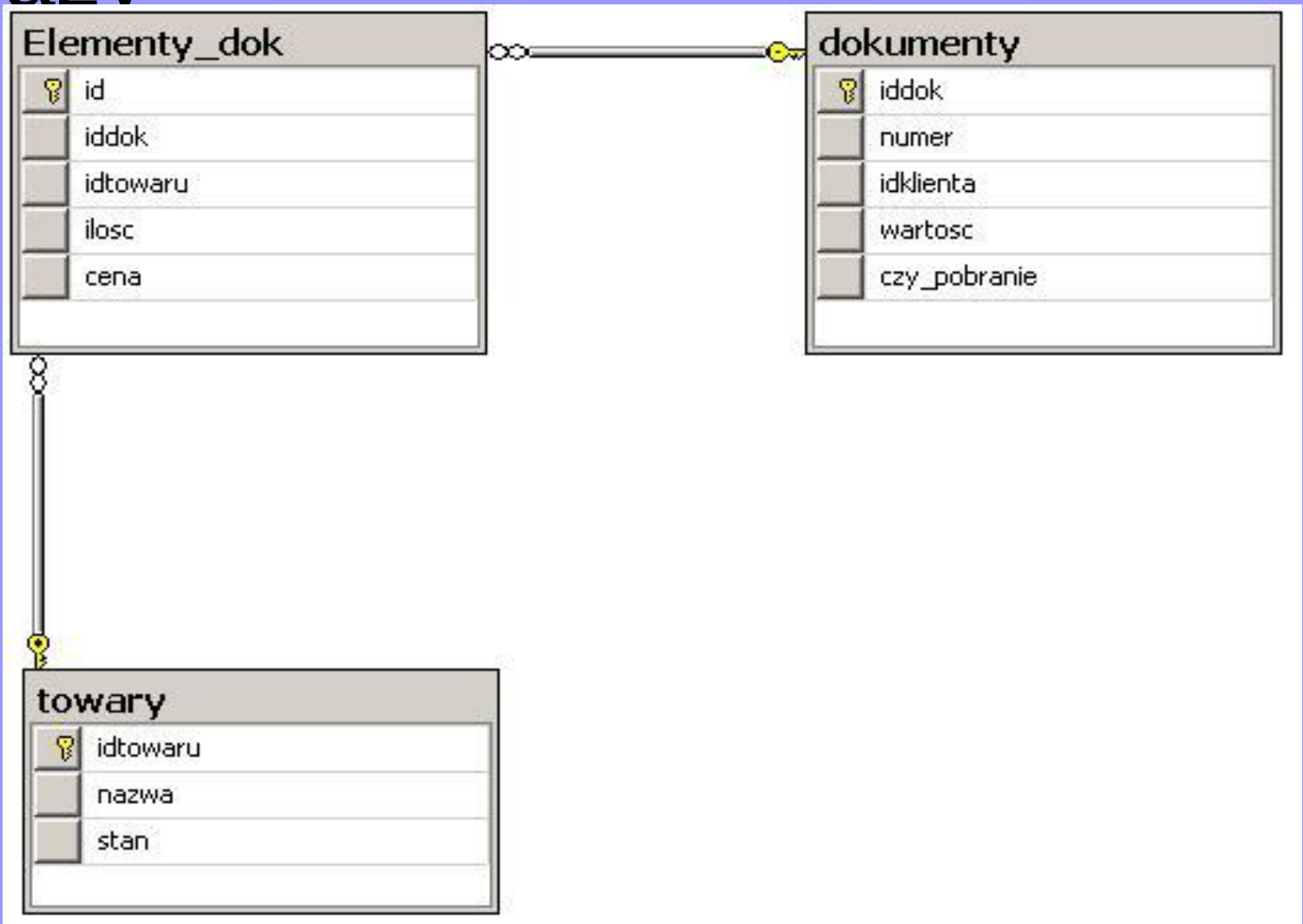
Skrypt modyfikujący

```
insert into miasta (nazwa) values('Warszawa')
insert into miasta (nazwa) values('Opole')
insert into miasta (nazwa) values('Sopot')
insert into klienci(nazwa,nip,idmiasta) values('Pierwszy','1111111111',1)
insert into klienci(nazwa,nip,idmiasta) values('Drugi','2222222222',2)
insert into klienci(nazwa,nip,idmiasta) values('Trzeci','3333333333',3)
update miasta set
nazwa='Gdañsk'
where nazwa='Sopot'

update klienci set
nazwa='DWA'
where nazwa='Drugi'

update klienci set
nazwa='JEDEN',
nip='1010101010'
where nazwa='Pierwszy'

delete from klienci where idklienta=3
```

Zawartość tabeli ***inspekcja***

id	kto	gdzie	kiedy	operacja	tabela	szczegoly
1	ANDRZEJ\Administrator	ANDRZEJ	2007-05-11 2...	INSERT	Miasta	Wstawiono wiersz o idmiasta=1
2	ANDRZEJ\Administrator	ANDRZEJ	2007-05-11 2...	INSERT	Miasta	Wstawiono wiersz o idmiasta=2
3	ANDRZEJ\Administrator	ANDRZEJ	2007-05-11 2...	INSERT	Miasta	Wstawiono wiersz o idmiasta=3
4	ANDRZEJ\Administrator	ANDRZEJ	2007-05-11 2...	INSERT	Klienci	Wstawiono wiersz o idklienta=1
5	ANDRZEJ\Administrator	ANDRZEJ	2007-05-11 2...	INSERT	Klienci	Wstawiono wiersz o idklienta=2
6	ANDRZEJ\Administrator	ANDRZEJ	2007-05-11 2...	INSERT	Klienci	Wstawiono wiersz o idklienta=3
7	ANDRZEJ\Administrator	ANDRZEJ	2007-05-11 2...	UPDATE	Miasta	IDMIASTA=3Zmieniono nazwę z Sopot na Gdańsk
8	ANDRZEJ\Administrator	ANDRZEJ	2007-05-11 2...	UPDATE	Klienci	IDKLIENTA=2Zmieniono nazwę z Drugi na DWA
9	ANDRZEJ\Administrator	ANDRZEJ	2007-05-11 2...	UPDATE	Klienci	IDKLIENTA=1Zmieniono nazwę z Pierwszy na JEDEN
10	ANDRZEJ\Administrator	ANDRZEJ	2007-05-11 2...	DELETE	Klienci	Usunięto wiersze o idklienta=3
NULL	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*

Kolumny wyliczalne –schemat bazy



Problemy

Zawartość kolumny STAN w tabeli ***towary*** oraz kolumny WARTOSC w tabeli ***dokumenty*** może być obliczana na podstawie tabeli ***elementy_dok***

- Problem spójności danych
- Problem wydajności

Rozwiązanie problemu

- Automatyzacja aktualizacji
  - Zapewnić wartość początkową kolumn (powinno być zero)
  - Zapewnić automatyczną modyfikację kolumny STAN i WARTOSC
  - Zapewnić, że modyfikację zawartości w.w kolumn mogą realizować tylko wybrane wyzwalacze

Rozwiązanie problemu

Wyzwalacze w tabelach TOWARY i DOKUMENTY dla operacji INSERT

```
CREATE TRIGGER [tr_towary_insert]
  on [dbo].[towary]
  for insert
as
BEGIN

update towary set
stan=0
where idtowaru in (select idtowaru from inserted)
END
```

Wyzwalacze modyfikujące kolumny STAN i WARTOSC

```
create TRIGGER  [tr_elementy_insert]
   ON  [dbo].[Elementy_dok]
   For  INSERT ,update
AS
BEGIN

 declare @iddok int,
        @idtowaru int,
        @ilosc int,
        @cena money,
        @czy_pobranie bit

select @iddok=iddok,@idtowaru=idtowaru,
      @ilosc=ilosc, @cena =cena
from inserted

select @czy_pobranie=czy_pobranie
from dokumenty
where iddok=@iddok

update dokumenty set
wartosc=wartosc+@ilosc*@cena
where iddok=@iddok

if @czy_pobranie=1
  set @ilosc=-1*@ilosc

update towary  set
stan=stan+@ilosc
where idtowaru=@idtowaru
END
```

Zabezpieczenie modyfikacji kolumn STAN i WARTOSC

```
 CREATE TRIGGER [tr_towary_update]
  on [dbo].[towary]
  for update
as
BEGIN
declare @st1 int, @st2 int, @st3 int
set @st1=trigger_nestlevel(object_id('tr_elementy_insert'))
set @st2=trigger_nestlevel(object_id('tr_elementy_delete'))
set @st3=trigger_nestlevel(object_id('tr_towary_insert'))
if @st1=0 and @st2=0 and @st3=0
      begin
      IF update(stan)
        rollback transaction
      end
END
```

Testowanie rozwiazania

Po wykonaniu skryptu :

```
insert into towary(nazwa,stan) values('Proszek',12)
insert into towary(nazwa,stan) values('Zmywak',20)
insert into towary(nazwa,stan) values('Pasta BHP',4)
```

Zawartość tabeli TOWARY

	idtowaru	nazwa	stan
▶	2	Proszek	0
	3	Zmywak	0
	4	Pasta BHP	0
*	NULL	NULL	NULL

Testowanie rozwiązania

Po wykonaniu skryptu

```
insert into dokumenty (numer,wartosc,czy_pobranie) values('12/07',0,false)

insert into elementy_dok(iddok,idtowaru,ilosc,cena) values(1,4,100,3)
insert into elementy_dok(iddok,idtowaru,ilosc,cena) values(1,2,200,4)
insert into elementy_dok(iddok,idtowaru,ilosc,cena) values(1,3,300,5)
```

Zawartość tabeli TOWARY i DOKUMENTY

	iddok	numer	wartosc	czy_pobranie
▶	1	12/07	2600,0000	False

	idtowaru	nazwa	stan
▶	2	Proszek	200
	3	Zmywak	300
	4	Pasta BHP	100

Przykład wyzwalacza INSTEAD OF

Zakładamy istnienie w bazie danych widoku

```
CREATE VIEW [dbo].[v_klienci]
AS
SELECT     dbo.klienci.idklienta, dbo.klienci.nazwa, dbo.klienci.nip,
      dbo.miasta.nazwa AS miasto
FROM         dbo.klienci INNER JOIN
                  dbo.miasta ON dbo.klienci.idmiasta = dbo.miasta.idmiasta
```

	idklienta	nazwa	nip	miasto
▶	1	JEDEN	1010101010	Warszawa
	2	DWA	2222222222	Opole

Procedura składowana

```
CREATE PROCEDURE [dbo].[klienci_insert]
@nazwa varchar(50),
@nip varchar(10),
@miasto varchar(50),


AS
BEGIN
      declare @idmiasta int

if not exists (select * from miasta where nazwa=@miasto)
  insert into miasta(nazwa) values(@miasto)

select @idmiasta=idmiasta from miasta  where nazwa=@miasto

insert into klienci(nazwa,nip,idmiasta)
   values (@nazwa,@nip,@idmiasta)

END
```

Wyzwalacz INSTEAD OF

```
CREATE  TRIGGER TR_v_klienci_insert
   ON  v_klienci
   INSTEAD OF INSERT
AS
BEGIN
     declare @nazwa varchar(150), @nip varchar(10),@miasto
     varchar(50)

    select @nazwa=nazwa,@nip=nip,@miasto=miasto
    from inserted

 exec klienci_insert @nazwa,@nip,@miasto
END
```

Testowanie

Po wykonaniu skryptu

```
insert into v_klienci(nazwa,nip,miasto) values('Czwarty','4444444444','Zabrze')
```

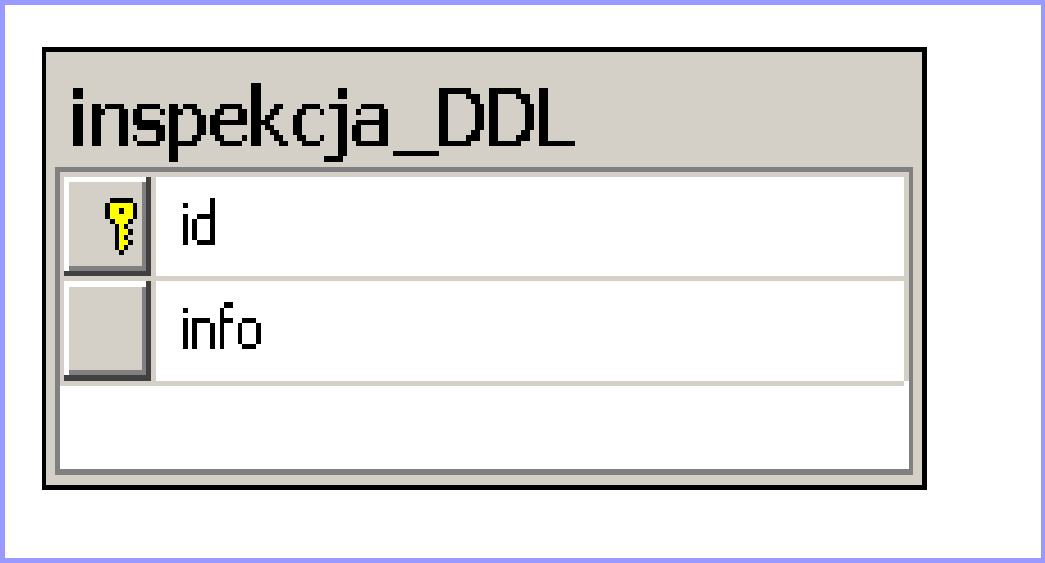
Zawartość tabel KLIENCI I MIASTA

	idklienta	nazwa	nip	idmiasta
1	1	JEDEN	1010101010	1
2	2	DWA	2222222222	2
3	4	Czwarty	4444444444	4

	idmiasta	nazwa
1	1	Warszawa
2	2	Opole
3	3	Gdańsk
4	4	Zabrze

Przykład wyzwalacza DDL

- Tworzymy nową tabelę INSPEKCJA_DDL



Tworzymy wyzwalacz

```
CREATE TRIGGER tr_inspektor
on DATABASE
FOR DDL_DATABASE_LEVEL_EVENTS
as
insert into inspekcja_DDL(info)
   values(eventdata())
```

Testowanie rozwiązania

Po wykonaniu skryptu

```
create view TEST
as
select nazwa,nip from klienci where idmiasta=1

alter table klienci add status bit

drop view TEST
```

Testowanie rozwiązania

Zawartość tabeli INSPEKCJA_DDL

	id	info
▶	2	<EVENT_INSTANCE><EventType>CREATE_VIEW</EventType><PostTime>2007-05-12T09:47:12.597</PostTime><SPID>54</SPID>...
	3	<EVENT_INSTANCE><EventType>ALTER_TABLE</EventType><PostTime>2007-05-12T09:47:35.833</PostTime><SPID>54</SPID>...
	4	<EVENT_INSTANCE><EventType>DROP_VIEW</EventType><PostTime>2007-05-12T09:47:47.460</PostTime><SPID>54</SPID><...
*	NULL	NULL

Wynik

```
<EVENT_INSTANCE>
  <EventType>CREATE_VIEW</EventType>
  <PostTime>2007-05-12T09:47:12.597</PostTime>
  <SPID>54</SPID>
  <ServerName>ANDRZEJ\AP</ServerName>
  <LoginName>ANDRZEJ\Administrator</LoginName>
  <UserName>dbo</UserName>
  <DatabaseName>PWI</DatabaseName>
  <SchemaName>dbo</SchemaName>
  <ObjectName>TEST</ObjectName>
  <ObjectType>VIEW</ObjectType>
  <TSQLCommand>
    <SetOptions ANSI_NULLS="ON" ANSI_NULL_DEFAULT="ON"/>
    <CommandText>create view TEST
as
select nazwa,nip from klienci where idmiasta=1</CommandText>
  </TSQLCommand>
</EVENT_INSTANCE>
```

Wynik

```
<EVENT_INSTANCE>
  <EventType>ALTER_TABLE</EventType>
  <PostTime>2007-05-12T09:47:35.833</PostTime>
  <SPID>54</SPID>
  <ServerName>ANDRZEJ\AP</ServerName>
  <LoginName>ANDRZEJ\Administrator</LoginName>
  <UserName>dbo</UserName>
  <DatabaseName>PWI</DatabaseName>
  <SchemaName>dbo</SchemaName>
  <ObjectName>klienci</ObjectName>
  <ObjectType>TABLE</ObjectType>
  <TSQLCommand>
    <SetOptions ANSI_NULLS="ON" ANSI_NULL_DEFAULT="ON"/>
    <CommandText>alter table klienci add status bit</CommandText>
  </TSQLCommand>
</EVENT_INSTANCE>
```

Wynik

```
<EVENT_INSTANCE>
  <EventType>DROP_VIEW</EventType>
  <PostTime>2007-05-12T09:47:47.460</PostTime>
  <SPID>54</SPID>
  <ServerName>ANDRZEJ\AP</ServerName>
  <LoginName>ANDRZEJ\Administrator</LoginName>
  <UserName>dbo</UserName>
  <DatabaseName>PWI</DatabaseName>
  <SchemaName>dbo</SchemaName>
  <ObjectName>TEST</ObjectName>
  <ObjectType>VIEW</ObjectType>
  <TSQLCommand>
    <SetOptions ANSI_NULLS="ON" ANSI_NULL_DEFAULT="ON"/>
    <CommandText>drop view TEST</CommandText>
  </TSQLCommand>
</EVENT_INSTANCE>
```

Podsumowanie

Przydatność wyzwalaczy w implementacjach baz danych zależy od pomysłu ich wykorzystania.

Podsumowanie - blaski

- Możliwość automatyzacji działań wynikających z logiki projektu lub logiki biznesowej
- Możliwość implementacji bardzo złożonych reguł integralności danych
- Usprawnienie procesów administrowania bazy danych
- Zmniejszanie prawdopodobieństwa utraty spójności danych
- Możliwość poprawy wydajności (kolumny wyliczane)

Podsumowanie - cienie

- Możliwość spadku wydajności związana z wydłużeniem czasu realizacji poleceń DDL lub DML
- Zwiększenie skomplikowania logiki bazy danych
- Możliwość generowania „trudnych” błędów (w sytuacji nieumiejętnego wykorzystania mechanizmu)

Dziękuje za uwagę