Bazy danych

Wykład 6

Agenda wykładu

- Procedury i funkcje
- Wyzwalacze
- Wyjątki

Deklaracja zmiennych

- DECLARE
  - @nazwaZmiennej AS TypDanych

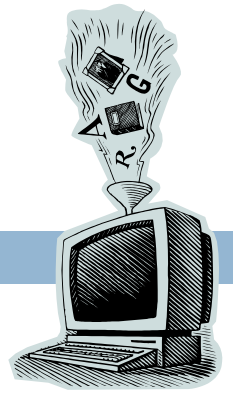
- Zmienne globalne:
  - @@SERVERNAME – nazwa instancji serwera bazodanowego
  - @@VERSION – wersja MS SQL Server
  - @@ERROR – kod błędu

Deklaracja zmiennych – typy danych

- DECLARE @NazwaZmiennej AS TypDanych

- Przykłady typów danych:
  - NVARCHAR(Liczba znaków) – dane tekstowe
  - np. NVARCHAR(64), NVARCHAR(1024)
  - SYSNAME – nazwy tabel, kolumn i innych obiektów bazodanowych
  - INT – liczba całkowita

Przykłady deklaracji zmiennych



```
1. DECLARE      @Schemat  AS SYSNAME,
2.              @NazwaTabeli   AS SYSNAME
3. DECLARE      @Kolumna AS SYSNAME,
4.              @Pozycja int,
5.              @Nazwa as NVARCHAR(64)
6. DECLARE  @InstrukcjaSQL  AS NVARCHAR(1024)
7. DECLARE  @InstrukcjaSQL2  AS NVARCHAR(1024)
```

Przypisywanie wartości zmiennym

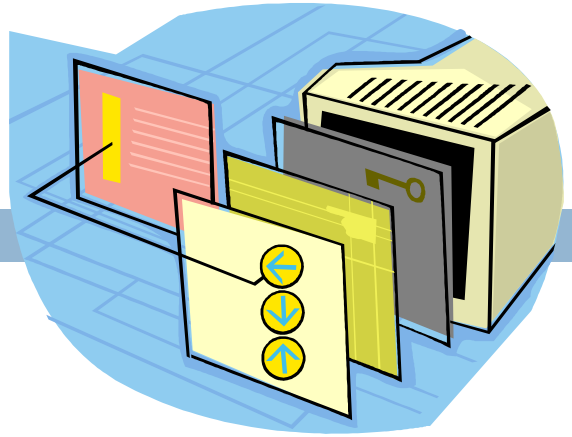
- SET @Zmienna= ‘Wartość‘
- SELECT @Zmienna = Wartość

- Przykłady:
  - SET @Schemat = 'dbo'
  - SET @NazwaTabeli = 'tab_2013_1'
  - SELECT @Pozycja = 41
  - SELECT @NumerStypendium = 0

Deklaracja bloku kodu

- Rozpoczęcie bloku kodu np. dla instrukcji IF
  - BEGIN
- Zakończenie bloku kod np. dla instrukcji IF
  - END

Instrukcja IF

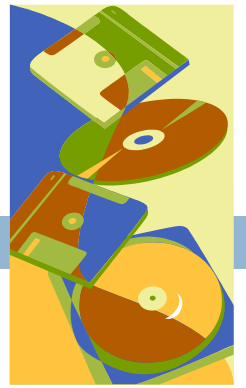


- IF (Warunek)
  - BEGIN
    - Instrukcje
  - END

☐ Przykład:

- IF (@Pozycja<=30)
  - BEGIN
    - SELECT @Pozycja = @Pozycja+2
  - END

Instrukcja WHILE



- WHILE Warunek
  - BEGIN
    - Instrukcje
  - END

- Przykład:
  - WHILE @Pozycja < 170
    - BEGIN
      - SELECT @Pozycja=@Pozycja+1
    - END

Słowa kluczowe BREAK i CONTINUE

- BREAK
  - przerywa iterację i opuszcza pętlę
- CONTINUE
  - przerywa bieżącą iterację pętli i rozpoczyna kolejną

Instrukcja WAITFOR DELAY

- Instrukcja WAITFOR DELAY umożliwia przerwania wykonywania kodu na określony czas.

- Przykład:
  - WAITFOR DELAY '02:00';
  - przerywa wykonywanie kodu na 2 godziny

Słowo kluczowe RETURN

- Słowo kluczowe RETURN przerywa wykonywanie funkcji lub procedury opuszczając ją bezwarunkowo.
- Słowo kluczowe RETURN może być wykorzystane do zwracania wartości np. liczby lub tekstu.

Instrukcja PRINT

- Instrukcja ta wykorzystywana jest do wyświetlania wartości na ekranie monitora.

- Przykład:
  - PRINT 'Student';

Zapytania złożone

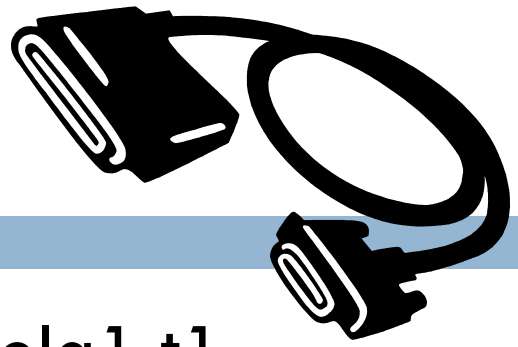
- WITH nazwa_zapytania AS (
  - ***podzapytanie (wewnętrzne)***
-)
  - ***zapytanie zewnętrzne***

- Wszystkie kolumny oraz aliasy kolumn dostępne w podzapytaniu będą do wykorzystania w zapytaniu zewnętrznym.

Przykład zapytania wewnętrznego i zewnętrznego

```
WITH Zapytanie AS (
SELECT '2012' as 'Rok',  woj as 'Wojewodztwo‘,
CASE
    WHEN symbol<500 THEN 1
    WHEN symbol<=800 THEN 2
    else 1
END as 'Forma',
'3' as 'Rodzaj‘, liczba as 'Wartosc’ FROM Tabela
)
SELECT rok, wojewodztwo, forma, rodzaj, sum(wartosc) as Wartosc
FROM Zapytanie
GROUP BY rok, wojewodztwo, forma, rodzaj
```

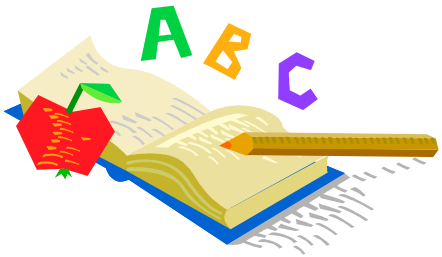
Łączenie wielu tabel



- SELECT t1.kolumna, t2.kolumna FROM tabela1 t1, tabela2 t2 WHERE t1.id=t2.id

- Przykład:
  - SELECT a.wojewodztwo, a.nazwa, b.wartosc FROM opis a, dane b where a.id_firmy=b.id_firmy

Sortowanie wyników (ORDER BY)



- SELECT a.wojewodztwo, a.nazwa, b.wartosc FROM opis a, dane b where a.id_firmy=b.id_firmy **ORDER BY wojewodztwo, nazwa**

- SELECT a.wojewodztwo, a.nazwa, b.wartosc FROM opis a, dane b where a.id_firmy=b.id_firmy **ORDER BY 1, 2, 3**

Porządek sortowania

- SELECT a.wojewodztwo, a.nazwa, b.wartosc FROM opis a, dane b where a.id_firmy=b.id_firmy
  ORDER BY wojewodztwo DESC,
  nazwa ASC

- Przełączniki zapytania:
  - DESC – malejąco
  - ASC – rosnąco

Procedury i funkcje T-SQL

Różnice pomiędzy procedurami a funkcjami

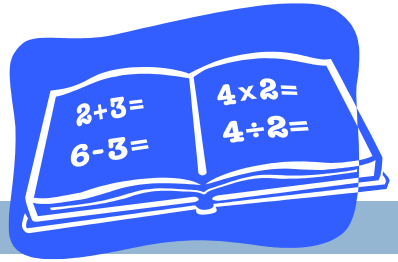
PROCEDURA

- nie zwraca wartości
- zwykle wykorzystywana do instrukcji INSERT
- może być wykorzystywana do zapytań SELECT

FUNKCJA

- zwraca wartość
- jej konstrukcja jest zbliżona do konstrukcji metody
- koniecznie musi zwracać wartość, inaczej nie zostanie utworzona

Cel stosowania procedur i funkcji

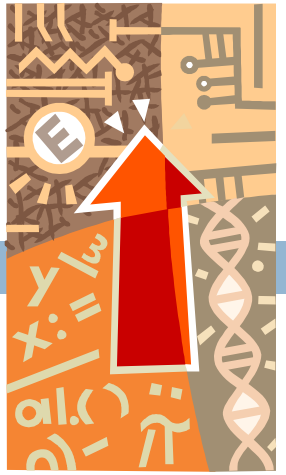


Wywoływanie zapytań tylko odpowiednio przygotowanych.

Są wywoływane najczęściej z kodu napisanego w innym języku (np. ASP .NET, PHP).

Pozwalają zapobiegać różnym atakom, w tym również SQL Injection.

Składnia funkcji

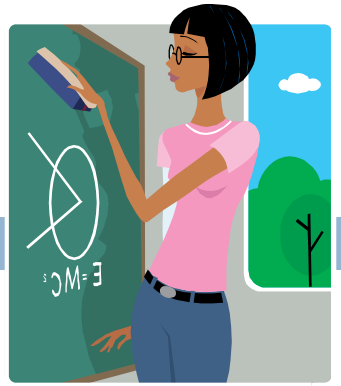


1. CREATE FUNCTION **NAZWA_FUNKCJI(ARGUMENTY)**
2. RETURNS **TYP_ZWRACANY**
3. WITH EXECUTE AS CALLER
4. AS
5. BEGIN
6. **-- LINIE KODU**
7. END;

Przykład funkcji


```
1.  CREATE FUNCTION dbo.ilewierszy (@Liczba INT)
2.  RETURNS Varchar(200)
3.  WITH EXECUTE AS CALLER
4.  AS
5.  BEGIN
6.      DECLARE
7.      @Wiersze INT,
8.      @Wynik VARCHAR(20)
9.      SELECT @Wiersze=200;
10.     IF(@Liczba<@Wiersze)
11.     BEGIN
12.       SET @Wynik='Mniej niż ' + Convert(Varchar,@Wiersze);
13.     END
14.     ELSE SET @Wynik='Równo lub więcej niż ' + Convert(Varchar,@Wiersze);
15.     RETURN @Wynik;
16. END;
```

Usuwanie funkcji



1. DROP FUNCTION ilewierszy;
2. GO

- Instrukcja GO jest niezbędna, jeżeli kolejna linia ma tworzyć funkcję.

Zmiana bazy danych

1. USE baza_operacyjna;
2. GO

- Przed utworzeniem funkcji należy się upewnić, że wybrana została odpowiednia baza danych do jej przechowywania.

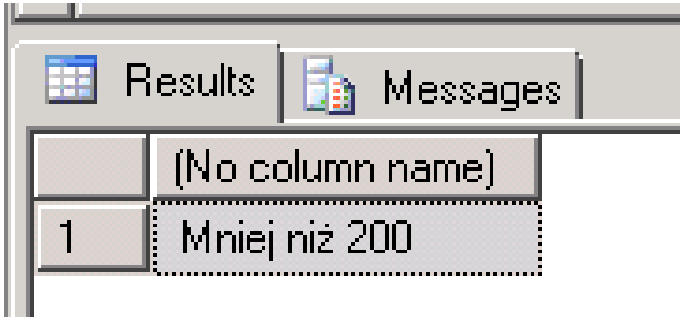
Pełny skrypt tworzący funkcję

```
1.  DROP FUNCTION ilewierszy;
2.  GO
3.  USE baza_operacyjna;
4.  GO
5.  CREATE FUNCTION dbo.ilewierszy (@Liczba INT)
6.  RETURNS Varchar(200)
7.  WITH EXECUTE AS CALLER
8.  AS
9.  BEGIN
10.     DECLARE
11.     @Wiersze INT,
12.     @Wynik VARCHAR(20)
13.     SELECT @Wiersze=200;
14.     IF(@Liczba<@Wiersze)
15.     BEGIN
16.         SET @Wynik='Mniej niż ' + Convert(Varchar,@Wiersze);
17.     END
18.     ELSE SET @Wynik='Równo lub więcej niż ' + Convert(Varchar,@Wiersze);
19.     RETURN @Wynik;
20. END;
```

Wywołanie funkcji zwracającej pojedynczą wartość

- select **nazwa_funkcji(argumenty);**

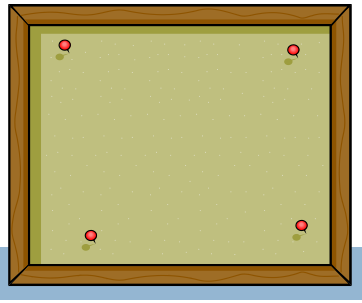
- Przykład:
  - select ilewierszy(40);



Przykład funkcji wyświetlającej dane z tabeli

```
1. CREATE FUNCTION dbo.funkcja_zawartosctabeli()
2. RETURNS TABLE
3. AS
4. RETURN
5. (
6.     SELECT DISTINCT NAZWA_WOJEWODZTWA
7.     FROM M30_REGION
8. );
```

Wywołanie funkcji zwracającej tabelę



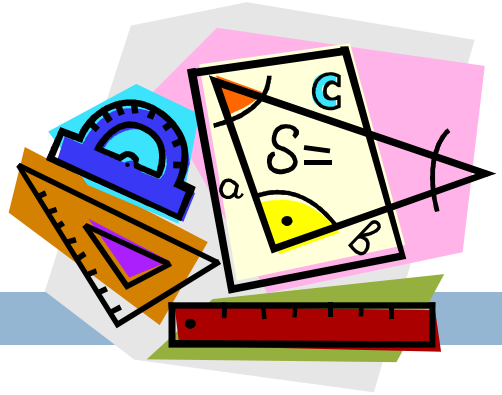
- select * from **nazwa_funkcji(argumenty);**

- Przykład:
  - select * from dbo.funkcja_zawartosctabeli();

Results | Messages

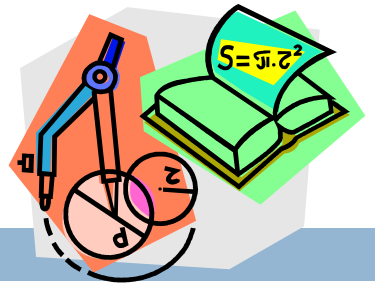
	NAZWA_WOJEWODZTWA
1	dolnośląskie
2	małopolskie
3	pomorskie
4	zachodniopomorskie

Składnia procedury



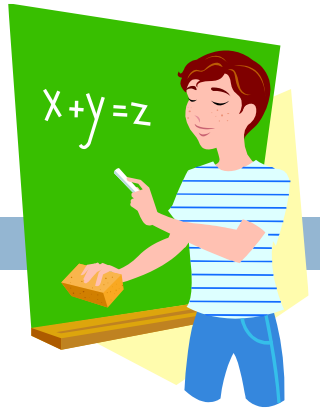
1. CREATE PROCEDURE **NAZWA_PROCEDURY**
2. **-- argumenty jako zmienne deklarowane**
3. AS
4. (
5. **-- blok kodu procedury**
6.);

Przykład procedury wyświetlającej dane z tabeli



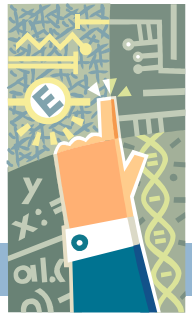

```
1. CREATE PROCEDURE dbo.zawartosctabeli
2. AS
3. (
4.     SELECT DISTINCT NAZWA_WOJEWODZTWA
5.     FROM M30_REGION
6. );
```

Przykład procedury z argumentem




```
1. DROP PROCEDURE sprawdz;
2. GO
3. CREATE PROCEDURE dbo.sprawdz
4. @Liczba INT
5. AS
6. (
7.     SELECT TOP 3 * FROM M30_REGION
8.     WHERE ID_REGION_PK>@Liczba
9. );
```

Wykonanie procedury

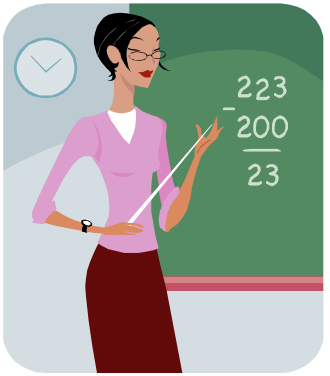


- Dwa alternatywne sposoby wykonujące procedurę.

1. EXEC **nazwa_procedury;**
2. EXECUTE **nazwa_procedury;**

3. EXEC **nazwa_procedury @argument=wartość;**
4. EXECUTE **nazwa_procedury @argument=wartość;**

Przykład wykonania procedury



- EXEC sprawdz @Liczba=12;

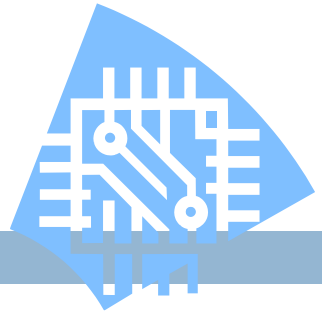
Results | Messages

	id_region_pk	woj	nazwa_wojewodztwa	region	nazwa_regionu
1	22	22	pomorskie	6	północny
2	32	32	zachodniopomorskie	6	północny

Inne przykłady procedur (1/3)

1. CREATE PROCEDURE procedura_insert (@nazwa varchar(14), @wartosc tinyint) AS
2. BEGIN
3. INSERT INTO TABELA VALUES (@nazwa, @wartosc)
4. END

Inne przykłady procedur (2/3)



1. CREATE PROCEDURE czy_istnieje_ identyfikator
2. @ identyfikator varchar(14)
3. AS
4. SELECT count(identyfikator) from tabela
5. where identyfikator like @identyfikator

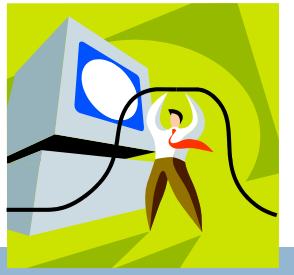
Inne przykłady procedur (3/3)

```
1.  CREATE PROCEDURE aktualizuj_dane
2.  (@identyfikator varchar(14),
3.  @p1 tinyint,
4.  @p2 tinyint,
5.  @p3 tinyint) AS
6.  BEGIN
7.  UPDATE DANE SET
8.  p1=@p1,
9.  p2=@p2,
10. p3=@p3
11. where identyfikator=@identyfikator
12. END
```

Wyświetlanie wszystkich funkcji utworzonych przez użytkownika

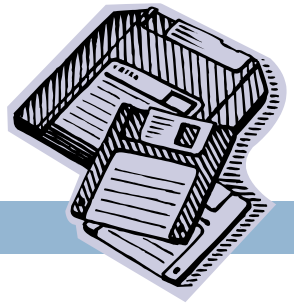
```
1. SELECT definition, type
2. FROM sys.sql_modules AS m
3. JOIN sys.objects AS o ON m.object_id =
   o.object_id
4.     AND type IN ('FN', 'IF', 'TF');
5. GO
```

Wyświetlanie wszystkich obiektów utworzonych przez użytkownika



1. SELECT definition, type
2. FROM sys.sql_modules AS m
3. JOIN sys.objects AS o ON m.object_id = o.object_id
4. GO

Utworzone funkcje i procedury

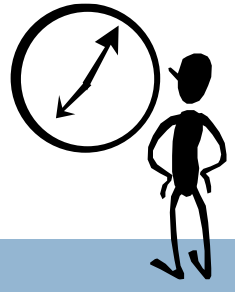


Results | Messages

	definition	type
1	CREATE FUNCTION dbo.ilewierszy (@Liczba INT) RETURNS Varchar(200) WITH EXECUTE AS CALLER AS BE...	FN
2	CREATE PROCEDURE dbo.zawartosctabeli AS (SELECT DISTINCT NAZWA_WOJEWODZTWA FROM M...	P
3	CREATE FUNCTION dbo.funkcja_zawartosctabeli() RETURNS TABLE AS RETURN (SELECT DISTINCT N...	IF
4	CREATE PROCEDURE dbo.sprawdz @Liczba INT AS (SELECT TOP 3 * FROM M30_REGION WHERE ID_R...	P

Wyzwalacze

Wyzwalacze



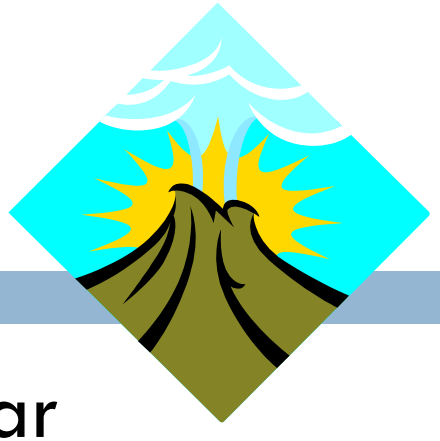
Małe programy zbliżone konstrukcją do procedur i funkcji, najczęściej wykonujące się w momencie wystąpienia określonego zdarzenia, np. dodania kolejnego wiersza do tabeli

Wyzwalacze są kompilowane i umieszczane w bazie danych

Wyzwalacze

- **CREATE TRIGGER** nazwa
- **ON** nazwa_tabeli
- **FOR [INSERT], [UPDATE], [DELETE]**
- **AS**
  - polecenia_triggera

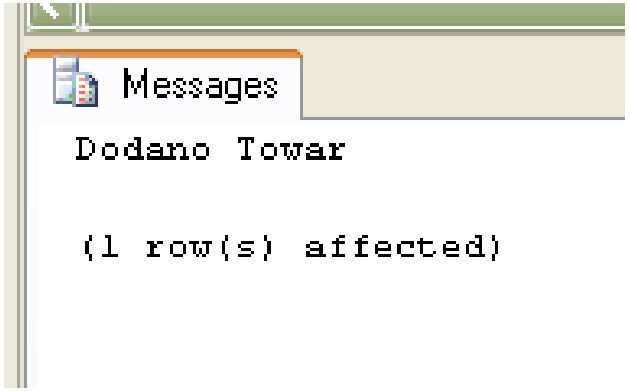
Przykład nr 1



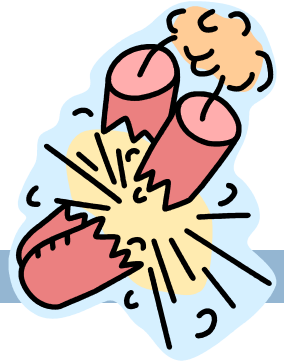
1. CREATE TRIGGER dodaj_towar ON towar
2. FOR INSERT
3. AS
4. BEGIN
5. PRINT 'Dodano Towar'
6. END

Działanie wyzwalacza dla przykładu nr 1

- insert into Towar (nazwa_towaru, cena, data_dodania) values (‘Ołówek',2.32,GETDATE());



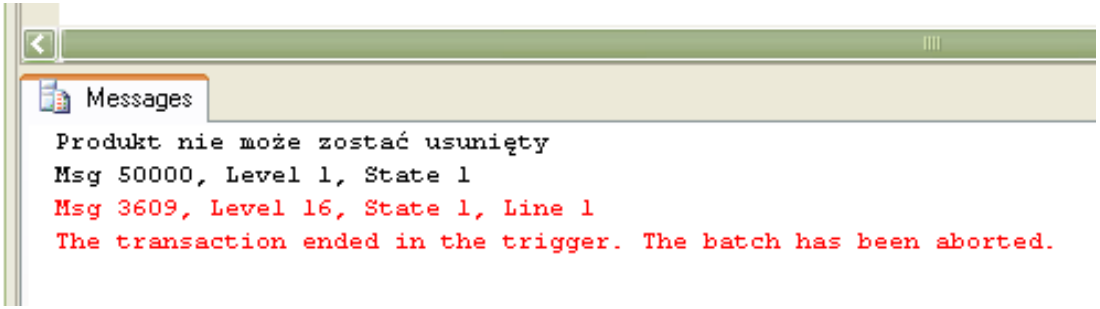
Przykład nr 2



```
1. CREATE TRIGGER usun_towar ON towar
2. FOR DELETE
3. AS
4. BEGIN
5. ROLLBACK TRANSACTION
6. RAISERROR('Produkt nie może zostać usunięty',1,1)
7. END
```

Działanie wyzwalacza dla przykładu nr 2

- delete from Towar where id=2;

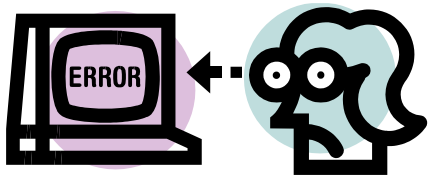


Instrukcje ROLLBACK TRANSACTION i RAISERROR

- **ROLLBACK TRANSACTION** – anuluje wszystkie zmiany wprowadzone przez wykonywaną transakcję

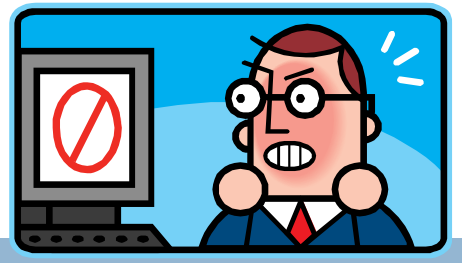
- **RAISERROR** – przerywa wykonywanie transakcji i wyświetla komunikat o błędzie

Instrukcja RAISERROR



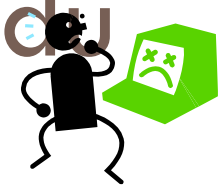
- **RAISERROR('Produkt nie może zostać usunięty',1,1)**
  - **'Produkt nie może zostać usunięty'** – skutkuje kodem błędu 50000
  - 1 – surowość błędu (ang. severity), przyjmuje wartość 1-18 oraz 19-25 (tylko SYSADMIN)
  - 1 – zdefiniowany przez użytkownika numer z zakresu od 0 do 255, pozwala na znalezienie fragmentu kodu odpowiedzialnego za błąd

Przykład nr 3

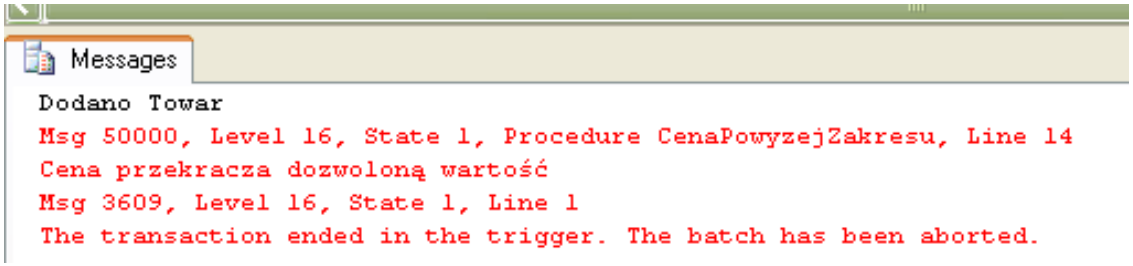


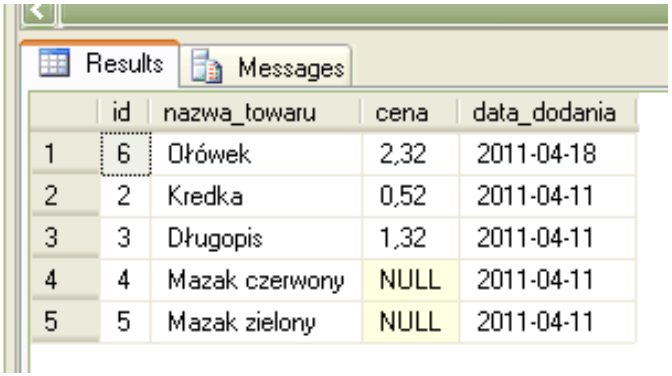

```
1.  CREATE TRIGGER CenaPowyzejZakresu ON Towar
2.  AFTER INSERT, UPDATE
3.  AS
4.  DECLARE
5.  @cena NUMERIC(5,2)
6.  BEGIN
7.  -- wyłączenie opcji wypisywania dodatkowego
8.  -- zestawu danych po instrukcji select
9.  SET NOCOUNT ON;
10. SELECT @cena = cena FROM towar
11. IF @cena > 200
12. BEGIN
13. ROLLBACK TRANSACTION
14. RAISERROR('Cena przekracza dozwoloną wartość',16,1)
15. END
16. END
```

Działanie wyzwalacza dla przykładu nr 3



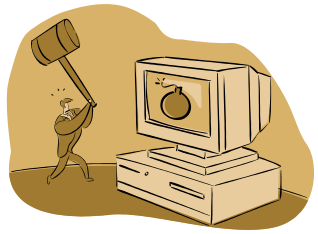
- insert into Towar (nazwa_towaru, cena, data_dodania) values ('Temperówka',299.31,GETDATE());



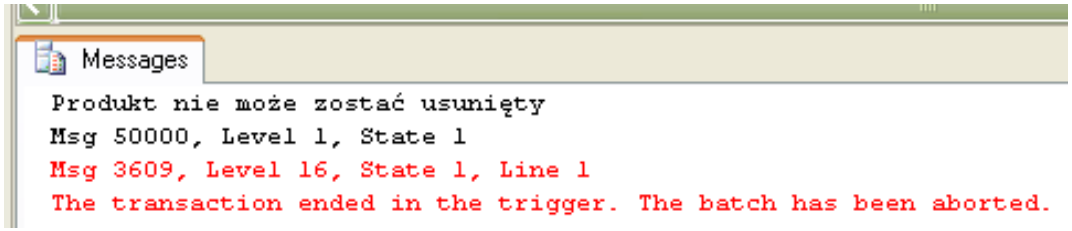


	id	nazwa_towaru	cena	data_dodania
1	6	Ołówek	2,32	2011-04-18
2	2	Kredka	0,52	2011-04-11
3	3	Długopis	1,32	2011-04-11
4	4	Mazak czerwony	NULL	2011-04-11
5	5	Mazak zielony	NULL	2011-04-11

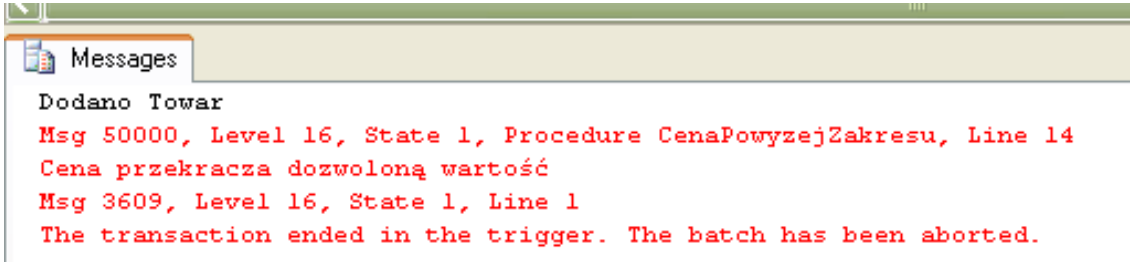
Porównanie surowości błędów



1. **ROLLBACK TRANSACTION**
2. **RAISERROR('Produkt nie może zostać usunięty',1,1)**



3. **ROLLBACK TRANSACTION**
4. **RAISERROR('Cena przekracza dozwoloną wartość',16,1)**

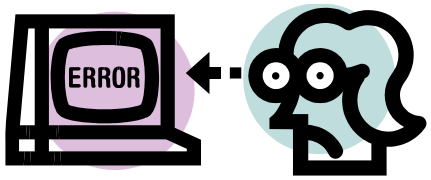


Wyjątki

Wyjątki

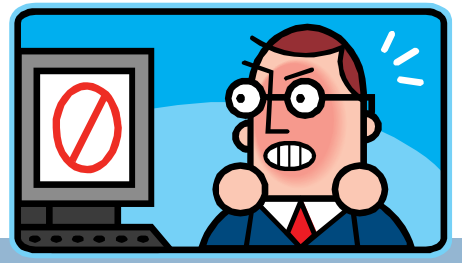
- Służą do przechwytywania oczekiwanych błędów
- Przykład wyjątków:
  - liczba wierszy przekracza spodziewaną
  - zapytanie nie zwróciło żadnego wiersza, a występuje instrukcja przypisania wiersza
  - dostęp do tabeli, która nie istnieje
  - brak uprawnień do tabeli

Ogólna składnia wyjątków



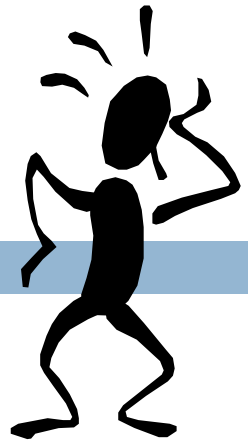
- BEGIN TRY
- { składnia SQL | blok składni }
- END TRY
- BEGIN CATCH
- { składnia SQL| blok składni }
- END CATCH
- [;]
  - gdzie:
    - składnia SQL – pojedyncza instrukcja
    - blok składni – zestaw instrukcji SQL umieszczony w bloku BEGIN – END

Przykład wyjątku



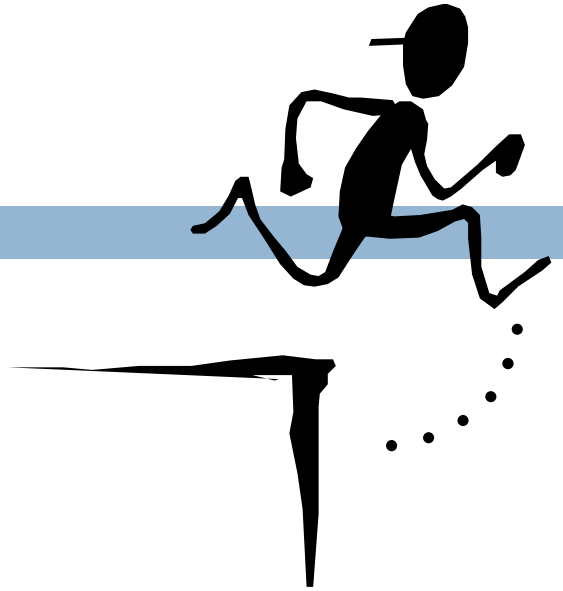

```
1.  BEGIN TRY
2.  SELECT 1/0;
3.  END TRY
4.  BEGIN CATCH
5.      SELECT
6.          ERROR_NUMBER() AS ErrorNumber,
7.          ERROR_SEVERITY() AS ErrorSeverity,
8.          ERROR_STATE() AS ErrorState,
9.          ERROR_PROCEDURE() AS ErrorProcedure,
10.         ERROR_LINE() AS ErrorLine,
11.         ERROR_MESSAGE() AS ErrorMessage;
12. END CATCH;
13. GO
```

Przykład nr 1 na podstawie dokumentacji T-SQL (1/2)



```
IF OBJECT_ID ( 'usp_GetErrorInfo', 'P' ) IS NOT NULL
    DROP PROCEDURE usp_GetErrorInfo;
GO

CREATE PROCEDURE usp_GetErrorInfo
AS
SELECT
    ERROR_NUMBER() AS ErrorNumber
    ,ERROR_SEVERITY() AS ErrorSeverity
    ,ERROR_STATE() AS ErrorState
    ,ERROR_PROCEDURE() AS ErrorProcedure
    ,ERROR_LINE() AS ErrorLine
    ,ERROR_MESSAGE() AS ErrorMessage;
GO
```

Przykład nr 1 na podstawie dokumentacji T-SQL (2/2)

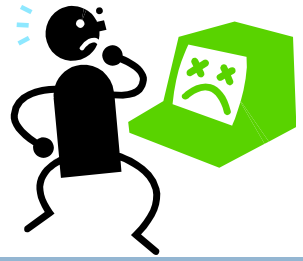


```
1. BEGIN TRY
2.     SELECT 1/0;
3. END TRY
4. BEGIN CATCH
5.     EXECUTE usp_GetErrorInfo;
6. END CATCH;
```

Przykład nr 2 na podstawie dokumentacji T-SQL (1/2)

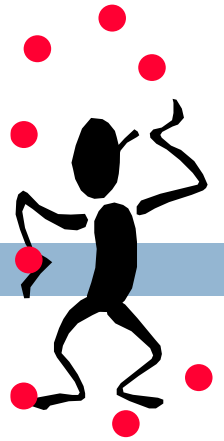

```
1. BEGIN TRANSACTION;


2. BEGIN TRY
3.     DELETE FROM Production.Product
4.     WHERE ProductID = 980;
5. END TRY
```

Przykład nr 2 na podstawie dokumentacji T-SQL (2/2)



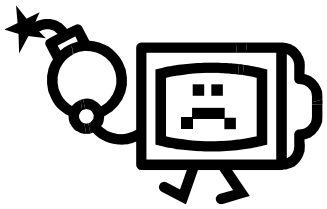

```
1.  BEGIN CATCH
2.      SELECT
3.          ERROR_NUMBER() AS ErrorNumber
4.          ,ERROR_SEVERITY() AS ErrorSeverity
5.          ,ERROR_STATE() AS ErrorState
6.          ,ERROR_PROCEDURE() AS ErrorProcedure
7.          ,ERROR_LINE() AS ErrorLine
8.          ,ERROR_MESSAGE() AS ErrorMessage;
9.      IF @@TRANCOUNT > 0
10.         ROLLBACK TRANSACTION;
11. END CATCH;
12. IF @@TRANCOUNT > 0
13.     COMMIT TRANSACTION;
14. GO
```

Instrukcja THROW



- Wyrzuca wyjątek i przechodzi natychmiast do bloku CATCH obsługującego ten wyjątek
- Jest zbliżona konstrukcyjnie do instrukcji RAISEERROR

Różnice pomiędzy RAISEERROR a THROW



- Wszystkie wyjątki wyrzucane przez THROW mają stopień surowości ustawiony na 16 (nie można go zmienić)
- Nie można formatować informacji w instrukcji THROW, tak jak to ma miejsce w przypadku RAISEERROR (instrukcja printf)
- Numer błędu THROW nie musi być zdefiniowany w wiadomościach błędów (msg_id, error_number)

Ogólna składnia instrukcji THROW

- THROW
- [{ numer błędu | @zmienna_lokalna },
- { wiadomość | @zmienna_lokalna },
- { stan| @zmienna_lokalna }
-] [;]
  - gdzie:
    - numer błędu – liczba całkowita z przedziału <50000; 2147483647>
    - wiadomość – tekst Unicode do 2048 znaków
    - stan – typ tinyint w przedziale <0;255>

Przykłady instrukcji THROW (1/3)

(1)

```
1. THROW 55000, 'Rekord nie istnieje.', 1;
```

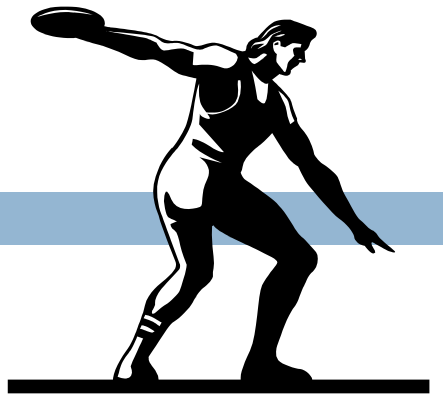
(2)

```
1. DECLARE @wiadomosc NVARCHAR(2048);
2. SELECT @wiadomosc = FORMATMESSAGE(1127);
3. THROW 50001, @wiadomosc, 1;
```

Przykłady instrukcji THROW (2/3)

```
1.  CREATE TABLE dbo.Test
2.  (    ID INT PRIMARY KEY
3.  );
4.  BEGIN TRY
5.      INSERT dbo.Test(ID) VALUES(1);
6.      INSERT dbo.Test (ID) VALUES(1);
7.  END TRY
8.  BEGIN CATCH
9.      PRINT 'Wyjątek w bloku CATCH.';
10.     THROW;
11. END CATCH;
```

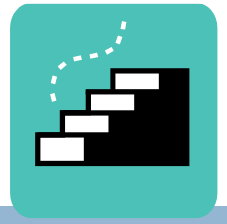
Przykłady instrukcji THROW (3/3)



- W bloku CATCH.
- Msg 2627, Level 14, State 1, Line 1
- Violation of PRIMARY KEY constraint 'PK__TestReth__3214EC272E3BD7D3'. Cannot insert duplicate key in object 'dbo.TestRethrow'.
- The statement has been terminated.

Instrukcja GOTO

Instrukcja GOTO

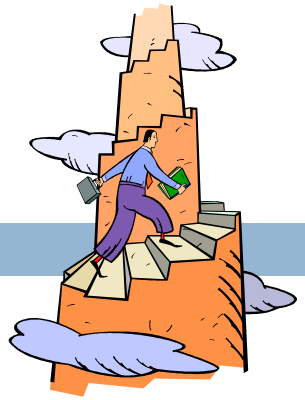


- Instrukcja GOTO pozwala przechodzić pomiędzy liniami kodu, które wcześniej oznaczone są etykietami.

- Definicja etykiety:
  - etykieta:

- Przejście do wcześniej zdefiniowanego bloku:
  - GOTO etykieta

Przykłady instrukcji GOTO



```
1.  DECLARE @Licznik int;
2.  SET @Licznik = 1;
3.  WHILE @Licznik < 10
4.  BEGIN
5.      SELECT @Licznik
6.      SET @Licznik = @Licznik + 1
7.      IF @Licznik = 4 GOTO Etykieta_Jeden
8.      IF @Licznik = 5 GOTO Etykieta_Dwa
9.  END
10. Etykieta_Jeden:
11.     SELECT 'Jestem w etykiecie Jeden.'
12.     GOTO EtykietaTrzy;
13. Etykieta_Dwa:
14.     SELECT 'Jestem w etykiecie Dwa.'
15. Etykieta_Trzy:
16.     SELECT 'Jestem w etykiecie Trzy.'
```

Pytanie?

Prawidłowe wywołanie procedury o nazwie **procedura** z argumentem liczbowym **a** to:

- EXECUTE **nazwa_procedury @a=10;**
- EXECUTE **nazwa_procedury(@10);**
- EXECUTE **nazwa_procedury(a=10);**
- EXECUTE **nazwa_procedury(10);**