

Paweł Rajba

pawel@cs.uni.wroc.pl

<http://www.itcourses.eu/>

SQL Server

T-SQL: Programowanie

Agenda

- Zmienne i operatory
- Instrukcje sterujące
- CASE i bazujące na nim wyrażenia
- Przegląd funkcji wbudowanych
- Wsady
- Procedury składowane, tymczasowe
- Zmienne tabelowe i tabele tymczasowe
- Funkcje użytkownika i systemowe
- Pobieranie informacji o kodzie

Zmienne i operatory

- Zmienne lokalne
 - deklaracja: `declare @nazwa typ [, @nazwa typ] ...`
 - ustawienie: `set @zmienna=wartosc`
 - wypisanie: `print @zmienna`
- Operatory
 - Arytmetyczne: `*`, `/`, `%`, `+`, `-`
 - Łączenie napisów: `+`
 - Logiczne: `not`, `and`, `or`

Instrukcje sterujące

- BEGIN..END
- IF..ELSE
- RETURN [n]
- WAITFOR { DELAY 'time' | TIME 'time' }
- WHILE
- ..BREAK
- ..CONTINUE

Wyrażenia bazujące na CASE

- CASE
 - jest skrótem na if..else if..else ...
 - jest także wyrażeniem, stąd możemy użyć np. w
 - instrukcji SELECT
 - na liście wybierającej
 - w klauzulach: WHERE, GROUP BY, ORDER BY
 - instrukcji INSERT
 - na liście wartości INSERT
 - instrukcji UPDATE
 - w klauzulach: SET i WHERE
 - instrukcji DELETE
 - w klauzuli WHERE

Wyrażenia bazujące na CASE

- COALESCE(w1, w2, w3, ..., wN)
 - zwraca pierwsze różne od NULL wyrażenie
 - jeśli wszystkie są NULL, zwraca NULL
 - równoważne

```
CASE
  WHEN w1 IS NOT NULL THEN w1
  ...
  WHEN wN IS NOT NULL THEN wN
ELSE NULL
END
```

Wyrażenia bazujące na CASE

- NULLIF(w1, w2)
 - równoważne
CASE
 WHEN w1=w2 THEN NULL
 ELSE w1
END
- ISNULL(w1, w2)
 - równoważne
CASE
 WHEN w1 IS NULL THEN w2
 ELSE w1
END

DEMO

- 01-zmienne
- 02-wyrażenia

Format Daty

- Format daty
 - SET DATEFORMAT {
mdy|dmy|ymd|ydm|dym|myd }
 - określa sposób traktowania napisów np. '7/4/2005'
 - Format ISO
 - jest traktowane jednoznacznie niezależnie od ustawienia dateformat

Funkcje

- Konwersja typów
 - CAST(wyrażenie AS typ_danych)
 - CONVERT(typ_danych[(len)], wyrażenie [, styl])
 - styl można określić dla dat i liczb
- Do konwersji z typów zmiennoprzecinkowych
 - STR(wyrażenie [, dlugosc [, miejsca_dziesietne]])

Tabela konwersji typów

	To:																																
From:	binary	varbinary	char	varchar	nchar	nvarchar	datetime	smalldatetime	date	time	datetimeoffset	datetime2	decimal	numeric	float	real	bigint	int(INT4)	smallint(INT2)	tinyint(INT1)	money	smallmoney	bit	timestamp	uniqueidentifier	image	ntext	text	sql_variant	xml	CLR UDT	hierarchyid	
binary																																	
varbinary																																	
char																																	
varchar																																	
nchar																																	
nvarchar																																	
datetime																																	
smalldatetime																																	
date																																	
time																																	
datetimeoffset																																	
datetime2																																	
decimal																																	
numeric																																	
float																																	
real																																	
bigint																																	
int(INT4)																																	
smallint(INT2)																																	
tinyint(INT1)																																	
money																																	
smallmoney																																	
bit																																	
timestamp																																	
uniqueidentifier																																	
image																																	
ntext																																	
text																																	
sql_variant																																	
xml																																	
CLR UDT																																	
hierarchyid																																	

- Explicit conversion
- Implicit conversion
- Conversion not allowed
- * Requires explicit CAST to prevent the loss of precision or scale that might occur in an implicit conversion.
- Implicit conversions between xml data types are supported only if the source or target is untyped xml. Otherwise, the conversion must be explicit.

Funkcje

- Daty i czasu
 - DATEADD(składnik, liczba, datagodzina)
 - dodaje ustalony składnik do datagodzina
 - wartości składnika:
 - year, month, day, dayofyear, quarter
 - hour, minute, second, millisecond
 - week, weekday
 - DATEDIFF(składnik, datagodzina1, datagodzina2)
 - zwraca różnicę pomiędzy datami
- Uwaga: typ DATETIME2 jest nowszy i daje większą precyzję
 - <http://stackoverflow.com/questions/1334143/sql-server-datetime2-vs-datetime>

Funkcje

- Daty i czasu
 - DATENAME(skłádnik, datagodzina)
 - zwraca nazwę reprezentującą skłádnik z datagodzina
 - DATEPART(skłádnik, datagodzina)
 - zwraca liczbę reprezentującą skłádnik z datagodzina
 - DAY(dg), MONTH(dg), YEAR(dg)
 - zwracają odpowiednio dzień, miesiąc i rok z datagodzina
 - GETDATE
 - zwraca bieżący czas i datę
 - GETUTCDATE
 - zwraca bieżący czas GMT

Funkcje

- Matematyczne
 - ABS, CEILING, FLOOR, ROUND
 - SIN, COS, TAN
 - PI, EXP, LOG, LOG₁₀
 - POWER, SQRT, SQUARE
 - SIGN, RAND

Funkcje

- Napisowe
 - UPPER, LOWER
 - TRIM, LTRIM, RTRIM
 - REPLACE(t₁, t₂, t₃) – wymienia w t₁, t₂ na t₃
 - REPLICATE(t, n)
 - wypisuje napis t, n razy, jeśli n<0, zwraca NULL
 - REVERSE(napis)
 - SPACE(liczba)
 - zwraca napis będący pewną liczbą spacji lub NULL, gdy liczba<0

Funkcje

- Napisowe

- STUFF – usuwa fragment napisu
- SUBSTRING – zwraca fragment napisu
- LEN – zwraca długość napisu
- LEFT, RIGHT – zwraca pewien fragment z lewej/prawej strony napisu

Funkcje

■ Systemowe

- `col_length('tabela', 'kolumna')`
 - maksymalna ilość znaków w kolumnie
- `datalength(wyrażenie)`
 - rozmiar typu danych wyniku
- `db_id('nazwa_bazy')`
 - id podanej bazy danych lub bazy bieżącej
- `db_name(1)`
 - nazwa podanej bazy danych lub bazy bieżącej

Funkcje

■ Systemowe

- `host_id()` – id procesu programu, z którego łączymy się do SQL Servera
- `host_name()` – nazwa stacji roboczej
- `user_id()` – id użytkownika bazy danych
- `user_name()` – nazwa użytkownika bazy
- `object_id(nazwa_obiektu)` – zwraca id obiektu
- `object_name(id_obiektu)` – zwraca nazwę obiektu

Funkcje

- Globalne

- Globalne funkcje typu @@

- Przykładowe:

- @@ERROR – numer błędu ostatniego polecenia
- @@ROWCOUNT – liczba wierszy przetworzonych przez ostatnie polecenie
- @@VERSION – wersja SQL Servera
- @@TRANCOUNT – liczba aktywnych transakcji w bieżącej sesji

DEMO

- o3a-daty1
- o3b-daty2

Wsady

- Instrukcje we wsadzie traktowane są jako jedna całość, jeden „program”
- Stosowanie wsadów zwiększa wydajność
 - np. wykonanie 200 insertów w jednym wsadzie może być nawet 10x szybsze, niż wykonanie tych insertów osobno

Wsady

- Przy tworzeniu wsadów są pewne ograniczenia:
 - wyrażenia create default, create procedure, create rule, create trigger, create view muszą być we wsadzie samodzielne
 - w jednym wsadzie nie można dodać lub zmienić kolumny w tabeli (ALTER TABLE), a potem się do nich odwoływać
 - jawnie trzeba podawać wywołanie za pomocą słowa EXEC (poza pierwszym)

DEMO

- o4a-wsady
- o4b-wsady

Wsady

- Błędy we wsadach
 - Mamy dwa rodzaje błędów:
 - Błędy kompilacji (compile)
 - Błędy wykonania (run-time)

DEMO

- 05-errors

Wsady

- Znaczenie polecenia GO
 - Interpretowany przez klientów (np. Mgmt Studio) znacznik końca wsadu
- SQL Server nigdy nie widzi GO

Wsady

- Znaczenie polecenia GO
 - Quiz: co zrobi poniższy kod
 - ```
select count(*) from authors
/*
go
select count(*) from titles
go
select count(*) from sales
go
*/
select count(*) from jobs
go
```

# Wsady

- Znaczenie polecenia GO
  - W takich sytuacjach należy używać komentarzy jednowierszowych: --

# Dynamic SQL

- Polega wykonaniu zapytania SQL będące w postaci napisu
  - Zwykle tworzonego dynamicznie w innych kodzie SQL, stąd „dynamic SQL”
- Można przekazywać parametry
- Do wykonania zapytania służy polecenie
  - `sp_executesql`
- Nie należy tego mechanizmu nadużywać
  - Jest kompilowane w runtime

# Dynamic SQL

- Składnia sp\_executesql:

```
sp_executesql [@stmt =] statement
[
 { , [@params =] N'@parameter_name data_type [OUT | OUTPUT][,...n]' }
 { , [@param1 =] 'value1' [,...n] }
]
```

- Do poczytania

- <https://msdn.microsoft.com/en-us/library/ms188001.aspx>
- [http://www.sommarskog.se/dynamic\\_sql.html](http://www.sommarskog.se/dynamic_sql.html)

# DEMO

---

- o6-dynamic-sql



# Procedury składowane

- Główne różnice między wsadem a procedurą:
  - wsad jest przesyłany od klienta do serwera i potem na serwerze wykonywany
  - procedura jest tylko na serwerze i tam jest bezpośrednio wykonywana
  - do procedury możemy przekazywać parametry, do wsadu oczywiście nie możemy ich przekazywać
- Do utworzenia procedury służy polecenie **CREATE PROCEDURE**

# Procedury składowane

- CREATE PROCEDURE, składnia
  - create procedure nazwa  
    [@parametr typ=default, ... [output]]  
as  
begin  
    instrukcje  
end
- Słowo output oznacza przekazywanie parametru przez zmienną

# Procedury składowane

- Zagnieżdżanie procedur, rekurencja
  - maksymalny poziom zagnieżdżenia do 32
  - zmienna @nestlevel – poziom zagnieżdżenia
- Wywołania sekwencyjne
  - ilość praktycznie bez ograniczeń

# DEMO

---

- 07a-procedury
- 07b-rekurencja

# Procedury tymczasowe

- Są tworzone w bazie tempdb
- Są trzy rodzaje takich procedur
  - prywatne
  - globalne

# Procedury tymczasowe

- Prywatne procedury tymczasowe
  - Nazwa takiej procedury zaczyna się od znaku #
  - Procedura jest dostępna tylko w ramach połączenia, w którym została utworzona (nie ma konfliktu nazw)
  - Po zakończeniu połączenia procedura jest usuwana

# Procedury tymczasowe

- Globalne procedury tymczasowe
  - Nazwa takiej procedury zaczyna się od znaku ##
  - Tworzona jest jedna kopia takiej procedury i jest ona dostępna dla wszystkich
  - Niezależnie od uprawnień, wszyscy mogą ją wykonywać

# Zmienne tabelowe

- Zmienne tabelowe
  - Są to zmienne, które reprezentują tabele
  - Typu table nie można
    - przekazywać jako parametrów procedur i funkcji
    - wykorzystać jako typu kolumny w tabeli
- Kilka uwag dotyczących typu table
  - Przy definicji listy kolumn tabeli można wykorzystać: [NOT] NULL, PRIMARY KEY, UNIQUE, CHECK, DEFAULT
  - Zmienne tabelowe nie mogą brać udziału w powiązaniach klucza obcego (w obie strony)



# Zmienne tabelowe

- Kilka uwag dotyczących typu table c.d.
  - Na zmiennych można wykonywać operacje SELECT, INSERT, UPDATE, DELETE
    - przy czym nie można użyć konstrukcji SELECT.. INTO zmienna ..
    - oraz konstrukcji INSERT INTO .. EXEC procedura
  - Dla zmiennych nie można utworzyć indeksu przy użyciu CREATE INDEX

# Tabele tymczasowe

- Nieco większe możliwości oferują tabele tymczasowe
  - Np. można dodawać indeksy
- Nazwa zaczyna się od
  - # (lokalne)
  - ## (globalne)
- Ciekawe rozważania do przeczytania:
  - <https://www.mssqltips.com/sqlservertip/1556/differences-between-sql-server-temporary-tables-and-table-variables/>
  - <https://www.simple-talk.com/sql/t-sql-programming/temporary-tables-in-sql-server/>
  - <http://stackoverflow.com/questions/2920836/local-and-global-temporary-tables-in-sql-server>
  - <http://sqlmag.com/t-sql/temporary-tables-local-vs-global>

# Typ tabelowy

- Za pomocą CREATE TYPE można także utworzyć typ tabelowy
- Przykładowo
  - `CREATE TYPE LocationTableType AS TABLE ( LocationName VARCHAR(50), CostRate INT );`
- Konstrukcja umożliwia przekazywanie do procedur tabel
  - ... ale tylko w trybie READONLY

# DEMO

---

- o8a-zmienne-tabelowe
- o8b-tabele-tymczasowe
- o8c-typ-tabelowy

# Funkcje użytkownika

- Funkcje skalarne
  - Funkcje, które zwracają wartość skalarną
  - Mogą przyjmować max 1024 parametry dowolnych typów poza
    - rowversion, timestamp, cursor, table
  - Muszą mieć oczywiście instrukcję RETURN
  - Do utworzenia takiej funkcji używamy polecenia CREATE FUNCTION

# Funkcje użytkownika

- Funkcje skalarne
  - Składnia CREATE FUNCTION  
CREATE FUNCTION [własciciel.]nazwa  
([{ @parametr [AS] typ [=default]} [,...n]]) RETURNS  
typ\_zwracanej\_wartosci  
AS  
BEGIN  
instrukcje  
RETURN wyrażenie\_skalarne  
END
  - Wywołanie musi być z podaniem właściciela
    - np. dbo.przeterminowane('owoce')

# Funkcje użytkownika

- Funkcje tabelowe
  - Zwracają jako wynik zestaw wierszy
  - Można je traktować jak widoki z parametrem
  - Są dwa rodzaje: bezpośrednie i wielopoleceniowe
  - Do utworzenia takich funkcji używamy polecenia  
CREATE FUNCTION

# Funkcje użytkownika

- Funkcje tabelowe bezpośrednie
  - Składnia  
CREATE FUNCTION [własciciel.]nazwa  
([{ @parametr [AS] typ [=default]} [,...n]])  
RETURNS TABLE  
[AS]  
RETURN [ ( ] wyrażenie\_select [ ) ]
  - Jak wynika ze składni funkcja zawiera tylko jedną instrukcję i jest to RETURN
  - Przy wywoływaniu nie trzeba podawać właściciela



# Funkcje użytkownika

- Funkcje tabelowe wielopoleceniowe
  - Składnia  
CREATE FUNCTION [własciciel.]nazwa  
([{ @parametr [AS] typ [=default]} [,...n]]) RETURNS  
@zmienna TABLE <definicja\_tabeli>  
[AS]  
BEGIN  
    instrukcje  
    RETURN  
END
  - Instrukcje powinny wstawić wiersze do @zmienna i to one będą stanowić zwracany wynik

# Funkcje użytkownika

- Uwagi
  - W funkcjach można wykonywać tylko określony zestaw instrukcji (lista w dokumentacji)
  - W funkcjach nie można wywoływać funkcji „niedeterministycznych” (lista tych funkcji w dokumentacji)
  - Jeśli chcemy skorzystać z wartości parametru domyślnego należy w jego miejsce podać default

# Funkcje systemowe

- Funkcje systemowe tzn. dostępne z każdej bazy danych
- Funkcje tabelowe
  - Odwołanie do tabelowej funkcji systemowej odbywa się poprzez dodanie przed nazwą znaków ::
  - Przykład
    - `select * from ::fn_helpcollations()`
- Inne funkcje
  - np. `fn_serverid('nazwa')`

# DEMO

---

- og-funkcje

# Pobranie informacji o kodzie

- Większość informacji jest przechowywana w tabeli master..syscomments
- Niektóre kolumny tabeli syscomments
  - colid
    - określa numer wiersza danej procedury
    - jest typu smallint (czyli może być do 32767 wierszy)
  - encrypted
    - jeśli zaszyfrowane, wartość 1, jeśli nie, wartość 0
  - compressed
    - jeśli skompresowane, wartość 1, jeśli nie, wartość 0
  - text
    - tekst wiersza procedury

# Pobranie informacji o kodzie

- Za pomocą procedury `sp_helptext` możemy zobaczyć kod procedur czy funkcji
  - Jeśli przy tworzeniu kodu funkcji (procedury) użyjemy opcji `ENCRYPTION`, takie możliwości już nie będzie
    - po zaszyfrowaniu nie ma możliwości odszyfrowania
- Informacje o kodzie można także znaleźć w `INFORMATION_SCHEMA.X`
  - `X=ROUTINES,PARAMETERS,ROUTINE_COLUMNS`

# DEMO

---

- 10-informacja-o-kodzie