Administracja i programowanie pod Microsoft SQL Server 2000

Paweł Rajba

pawel@ii.uni.wroc.pl
http://www.kursy24.eu/

Zawartość modułu 4

- Wsady
- Procedury składowane
- Procedury składowane tymczasowe, startowe
- Zmienne tabelowe
- Funkcje
  - użytkownika
  - systemowe
- Pobieranie informacji o kodzie

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
- Przy tworzeniu wsadów są pewne ograniczenia:
  - wyrażenia create default, create procedure, create rule, create trigger, create view muszą być we wsadzie samodzielne

Wsady

- Ograniczenia c.d.
  - w jednym wsadzie nie można dodać lub zmienić kolumny w tabeli (ALTER TABLE), a potem się do niek odwoływać
  - jawnie trzeba podawać wywołanie za pomocą słowa EXEC (poza pierwszym)
- Przykład
  - sqlserver-p04-01A.sql
  - sqlserver-p04-01B.sql

Wsady

- Błędy we wsadach
  - Mamy dwa rodzaje błędów:
    - Błędy kompilacji (compile)
    - Błędy wykonania (run-time)
  - Przykłady
    - sqlserver-p04-02.sql

Wsady

- Znaczenie polecenia GO
  - Interpretowany przez klientów (np. Query Analyzera) znacznik końca wsadu
  - SQL Server nigdy nie widzi GO i nawet nie wie co to

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: --

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 – przykład składni:
  - ```
    create procedure nazwa[;liczba]
      [@parametr typ=default, ... [output]]
    as
    begin
      instrukcje
    end
    ```
  - Parametr liczba pozwala utworzyć grupę procedur, którą można potem usunąć jednym poleceniem
  - Słowo output oznacza przekazywanie parametru przez zmienną


    ```

# Procedury składowane

- Przykład
  - sqlserver-p04-03.sql

 

# Procedury składowane

- Zagnieżdżanie procedur, rekurencja
  - maksymalny poziom zagnieżdżenia do 32
  - zmienna @nestlevel – poziom zagnieżdżenia
- Wywołania sekwencyjne
  - ilość praktycznie bez ograniczeń
- Przykład
  - sqlserver-p04-04.sql

 

# 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

 

# Startowe procedury

- Są to procedury uruchamiane przy starcie serwera i działają w tle
- Uruchamianych może być kilka procedur, ponieważ ich uruchamianie jest asynchroniczne
- Startowa procedura musi być w bazie master
- Do ustawienia procedury jako startowej służy polecenie
  - sp_procoption nazwa, startup, true
  - lub też można to zrobić w EM, przy przeglądaniu właściwości procedury (jest odpowiedni checkbox)

 

# Startowe procedury

- Poprzez opcje uruchomienia serwera -f można wyłączyć uruchamianie takich procedur
  - EM | Właściwości serwera | Startup Parameters...
- Przykład
  - sqlserver-p04-05.sql

 

# 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
- Przykład
  - sqlserver-p04-06.sql

 

# Funkcje użytkownika

- Funkcje skalarne
  - Takie 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 [wlasciciel.]nazwa
([{ @parametr [AS] typ [=default]} [,...n]])
RETURNS typ_zwracanej_wartosci
AS
BEGIN
   instrukcje
   RETURN wyrazenie_skalarne
END
```

  - Wywołanie musi być z podaniem właściciela
    - np. dbo.przeterminowane('owoce')

Funkcje użytkownika

- Funkcje tabelowe
  - Zwracają jako wynik zestaw wierszy
  - Możne 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 [wlasciciel.]nazwa
([{ @parametr [AS] typ [=default]} [,...n]])
RETURNS TABLE
[AS]
RETURN [ ( ] wyrazenie_select [ ) ]
```

  - Jak wynika ze składni funkcja zawiera tylko jedną instrukcje i jest to RETURN
  - Przy wywoływaniu nie trzeba podawać właściciela

Funkcje użytkownika

- Funkcje tabelowe wielopoleceniowe
  - Składnia

```
CREATE FUNCTION [wlasciciel.]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 użytkownika

- Przykład
  - sqlserver-p04-07.sql

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')

Pobieranie 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

Pobieranie 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
- Przykład
  - sqlserver-p04-08.sql

