Bazy danych

Wykład 5_2

Temat: Obiekty kodu T-SQL przechowywane na serwerze

Sławomir Świętoniowski

Plan wykładu

1. **Widoki.**
2. Procedury przechowywane.

Widoki (Views)

Widok (ang. *view*, nazywany także „perspektywą”) – jest zapytaniem SQL, przechowywanym w bazie danych, w postaci obiektu. Stanowi on swego rodzaju „okno na dane”, znajdujące się w bazie.

- Cele stosowania widoków:
  - **dostarczenie użytkownikowi tylko takiej informacji, która jest dla niego przydatna i zrozumiała** (np. dane zagregowane lub złączone z wielu tabel, nie zawierające identyfikatorów autonumerycznych);
  - **zwiększenie wydajności** - serwer przechowuje kompilowany plan wykonania dla widoku (nieindeksowanego), dzięki czemu jego uruchamianie jest szybsze od wywoływania zwykłego zapytania SQL;
  - **kontrola dostępu do danych** – różni użytkownicy mogą mieć prawo uruchamiania różnych widoków w ramach tej samej bazy danych.
- Widoki nie zajmują miejsca na dysku, gdyż nie są fizycznie przechowywane w bazie, lecz tworzone na bieżąco w formie tabel wirtualnych.
- Za pomocą widoku można nie tylko wyświetlać dane, ale także wstawiać je do tabel, modyfikować i usuwać (z pewnymi ograniczeniami).

Zasady tworzenia widoków

- Widok jest zawsze oparty na poleceniu SELECT, które wybiera dane z jednej lub kilku tabel (albo widoków). Polecenie to podajemy po słowie AS.
- Wyrażenie SELECT w definicji widoku nie może zawierać:
  - odwołania do tabeli tymczasowej lub zmiennej tabelarycznej;
  - odwołania do więcej niż 1024 kolumn;
  - klauzuli ORDER BY, chyba że wyrażenie SELECT zawiera także klauzulę TOP;
  - słowa kluczowego INTO;
  - klauzul COMPUTE i COMPUTE BY.
- Widoki muszą być tworzone w bieżącej bazie danych. W poleceniu CREATE VIEW nie można określić nazwy bazy.

Tworzenie i modyfikowanie widoku – składnia

Polecenia CREATE VIEW i ALTER VIEW są praktycznie identyczne, ponieważ każda modyfikacja widoku polega na zamianie jego dotychczasowej zawartości na nową, określoną w poleceniu ALTER VIEW.

CREATE / ALTER VIEW [< database_name > .] [< owner > .] view_name
[(column [,...n])]
[WITH < view_attribute > [,...n]]
AS
select_statement
[WITH CHECK OPTION]

< view_attribute > ::=

{ ENCRYPTION | SCHEMABINDING | VIEW_METADATA }

T-SQL: Tworzenie widoku Przykłady

```
-- Prosty widok wyświetlający przedmioty (bez opiekunów).
CREATE VIEW V_Przedmiot_dane
AS -- Niektóre z wybieranych pól mają zmienioną nazwę – nagłówek kolumny.
SELECT KodPrzedmiotu AS [Kod przedmiotu], Nazwa,
       FormaZajec AS [Forma zajec]
FROM Przedmiot
GO

-- Modyfikacja widoku polega na podaniu nowej zawartości jego kodu T-SQL.
ALTER VIEW V_Przedmiot_dane
AS -- Dodatkowo mają być wyświetlane pola [Semestr] i [ECTS].
SELECT KodPrzedmiotu AS [Kod przedmiotu], Nazwa, Semestr, ECTS,
       FormaZajec AS [Forma zajec]
FROM Przedmiot
GO
```

Zwracanie danych z widoku

- Widok może być traktowany jak wirtualna tabela, której wiersze nie są zapisywane trwale w bazie. Zwrócenie danych z widoku odbywa się przez polecenie "SELECT...".
- Jeżeli chcemy wyświetlić tylko niektóre kolumny, ich nazwy muszą być zgodne z nazwami kolumn zdefiniowanymi w widoku, a nie w tabelach źródłowych.

```
SELECT [Kod przedmiotu], Nazwa
FROM V_Przedmiot_dane
GO
```

Wstawianie danych przez widok

Za pomocą widoku można nie tylko wyświetlać dane, ale też wstawiać je trwale do bazy. Jest to możliwe, jeżeli widok zawiera wszystkie pola wymagane w tabeli (czyli NOT NULL).

```
-- W widoku [V_Przedmiot_dane] są wszystkie pola obowiązkowe, nie ma zaś
-- pola [Opiekun] z dopuszczalną wartością NULL. Możemy zatem wykorzystać
-- ten widok do wstawiania danych.
INSERT V_Przedmiot_dane
-- Ważne jest, aby nazwy pól były zgodne z definicją widoku, a nie docelowej tabeli.
([Kod przedmiotu], Nazwa, Semestr, ECTS, [Forma zajec])
VALUES
('INF708W', 'Windows 2', 7, 4, '20000')
GO
```

Modyfikowanie i usuwanie danych przez widok

Widok może także służyć do modyfikowania i usuwania wierszy na identycznych zasadach, jak w przypadku zwykłych tabel.

```
-- Modyfikowanie danych przez widok.
UPDATE V_Przedmiot_dane
SET ECTS = 5
WHERE [Kod przedmiotu] = 'INF708W'
GO

-- Usuwanie danych przez widok.
DELETE V_Przedmiot_dane
WHERE Semestr = 7
GO
```

Plan wykładu

1. Widoki.

2. Procedury przechowywane.

Procedury przechowywane

Procedura przechowywana (ang. *stored procedure*) – jest to jedno lub więcej poleceń języka SQL, przechowywanym w bazie danych w postaci wykonywalnego obiektu.

Cechy procedur przechowywanych:

- można je wywoływać interakcyjnie z poziomu aplikacji klienta, a także z innej procedury przechowywanej lub wyzwalanej;
- procedury mogą pobierać i zwracać parametry, co zwiększa ich użyteczność i elastyczność (np. w porównaniu z widokami);
- mogą zwracać zbiór wyników (np. wierszy z tabeli) i kod wyjścia;
- procedury są przechowywane na serwerze w formie skompilowanej – razem z planem wykonania (*execution plan*).

Sposoby zwracania danych z procedury przechowywanej

- Procedura przechowywana może zwracać dane na cztery sposoby:
  - zbiór wierszy (record set), będący wynikiem wykonania polecenia SELECT;
  - parametry wyjściowe (OUTPUT), zwracające wartości (np. integer, char) albo wskaźnik na kursor (zmienna typu cursor);
  - kod powrotu (return code, return value) – zawsze jako liczba całkowita; może służyć do wychwytywania i obsługi błędów;
  - globalny kursor, do którego można się odwoływać poza procedurą.

Procedury przechowywane Zalety (1)

- **gwarancja integralności danych:**
  - kodowanie logiki aplikacji w pojedynczym miejscu, co ułatwia zarządzanie regułami biznesowymi i zachowanie ich jednolitości;
  - jeśli aplikacja klienta może zmieniać dane wyłącznie za pośrednictwem procedur, jest mniejsze ryzyko wprowadzenia modyfikacji naruszających spójność danych;
  - ograniczenie błędów programistycznych przy tworzeniu aplikacji wielowarstwowych – łatwiejsze wykonywanie złożonych operacji, przekazywanie mniejszej ilości informacji pomiędzy warstwami;
  - zautomatyzowanie złożonych transakcji o dużym znaczeniu dla integralności danych;
- **modularny charakter programowania** – kod SQL podzielony jest na mniejsze fragmenty, które są łatwiejsze do zarządzania;
- **łatwiejsze utrzymanie i modyfikacja aplikacji wielowarstwowej** – przejrzyste oddzielenie bazy danych od warstwy dostępu do danych;

Procedury przechowywane Zalety (2)

- **wzrost bezpieczeństwa danych** – łatwiejsza ochrona przed atakami (np. przed tzw. *wstrzyknięciem kodu SQL – ang. SQL injection* – zob. poz. nr 1 w spisie literatury);
- **ograniczenie ruchu w sieci** – procedury (nawet te złożone z bardzo wielu poleceń SQL) są wywoływane zdalnie za pomocą pojedynczego polecenia z parametrami;
- **wzrost wydajności przy powtarzających się wywołaniach** – procedury na serwerze są optymalizowane (kompilowane przy pierwszym uruchomieniu i przechowywane potem razem z gotowym planem wykonania);
- **ograniczenie dostępu do tabel tylko do zdefiniowanych czynności** – aplikacja klienta może mieć uprawnienia tylko do procedur przechowywanych, a nie bezpośrednio do tabel.

Procedury przechowywane Wady

- **Transact-SQL jest językiem o ograniczonym możliwościach** – może on być niewystarczający do oprogramowania bardzo złożonych operacji; sytuacja ta ulega jednak zmianie, do implementacji procedur przechowywanych, oprócz standardowego T-SQL, można wykorzystywać także wysoko-poziomowe, obiektowe języki platformy MS .NET (np. C#, VB);
- **gorsza integracja ze środowiskami programistycznymi** – nie wszystkie narzędzia do tworzenia aplikacji wielowarstwowych pozwalają na kontrolę wersji i usuwanie błędów (ang. *debugging*) na poziomie procedur przechowywanych (chociaż np. *MS Visual Studio .NET* ma tę możliwość);
- **mała przenośność** – mimo istnienia specyfikacji tworzenia procedur przechowywanych w standardzie ANSI SQL 99, przez większość producentów systemów baz danych nie jest ona w pełni przestrzegana; dlatego kod procedur przechowywanych najczęściej nie daje się bezpośrednio przenosić np. z bazy MS SQL Server do Oracle.

T-SQL: Tworzenie i modyfikowanie procedury – składnia

Podobnie, jak w przypadku widoków, składnia polecenia tworzenia (CREATE) i modyfikowania (ALTER) procedury przechowywanej jest identyczna, ponieważ każda modyfikacja procedury polega na zamianie jego dotychczasowej zawartości na nową, określoną w poleceniu ALTER PROCEDURE.

```
CREATE / ALTER PROC [ EDURE ] procedure_name [ ; number ]
    [ { @parameter data_type }
    [ VARYING ] [ = default ] [ OUTPUT ]
    ] [ ,...n ]

    [ WITH  -- Opcja WITH ENCRYPTION służy do zaszyfrowania kodu T-SQL.
    { RECOMPILE | ENCRYPTION | RECOMPILE , ENCRYPTION } ]

    [ FOR REPLICATION ]

    AS sql_statement [ ...n ]
```

Tworzenie procedury przechowywanej – przykład

```
-- Dodawanie nowego przedmiotu.
CREATE PROC Przedmiot_wstawianie -- Unikalna nazwa procedury.
@Par_KodPrzedmiotu char(7), -- Lista parametrów.
@Par_Nazwa char(128),
@Par_Semestr tinyint,
@Par_ECTS tinyint = 4, -- Wartość domyślna parametru.
@Par_FormaZajec char(10),
@Par_Opiekun int
AS -- Po słowie „AS” następuje ciało procedury – właściwy ciąg poleceń T-SQL.
INSERT Przedmiot
(KodPrzedmiotu, Nazwa, Semestr, ECTS, FormaZajec, Opiekun)
VALUES -- Zamiast stałych wykorzystywane są wartości parametrów procedury.
(@Par_KodPrzedmiotu, @Par_Nazwa, @Par_Semestr, @Par_ECTS,
@Par_FormaZajec, @Par_Opiekun)
GO
```

Przykład wywołania procedury przechowywanej

```
-- Do wywołania procedury używamy słowa EXECUTE (albo EXEC), po którym
-- podajemy parametry i ich wartości. Pomaga to uniknąć pomyłek.
EXECUTE Przedmiot_wstawianie
@Par_KodPrzedmiotu = 'INF407W',
@Par_Nazwa = 'Bazy danych',
@Par_Semestr = 4,
@Par_ECTS = DEFAULT, -- Wymuszenie wartości domyślnej parametru.
@Par_FormaZajec = '20000',
@Par_Opiekun = 1
GO
-- Można także stosować skróconą składnię – podajemy tylko wartości parametrów,
-- koniecznie w takiej kolejności, w jakiej są zadeklarowane w definicji procedury.
-- Taki zapis jest krótszy i wygodniejszy, ale bardziej podatny na błędy.
EXEC Przedmiot_wstawianie 'INF507W', 'Sieciowe bazy danych', 5, 4, '10000', 1
GO
```

Procedura z parametrami wyjściowymi – przykład

```
CREATE PROC Przedmiot_edycja -- Procedura zmienia dane przedmiotu.
@Par_KodPrzedmiotu char(7), -- Parametr służący do identyfikacji przedmiotu.
@Par_Nazwa char(128), -- Pozostałe parametry przekazują nowe wartości pól.
@Par_Semestr tinyint,     -- Konwencja nazewnicza parametrów wejściowych
@Par_ECTS tinyint,        -- „@Par_<nazwa>”, a wyjściowych – „@Out_<nazwa>”.
@Par_FormaZajec char(10),
@Par_Opiekun int,
@Out_LiczbaWierszy int OUTPUT -- Parametr wyjściowy.
AS
UPDATE Przedmiot
SET   Nazwa = @Par_Nazwa, Semestr = @Par_Semestr, ECTS = @Par_ECTS,
      FormaZajec = @Par_FormaZajec, Opiekun = @Par_Opiekun
 WHERE KodPrzedmiotu = @Par_KodPrzedmiotu
 -- Wartość parametru wyjściowego to liczba zmodyfikowanych wierszy.
 SELECT @Out_LiczbaWierszy = @@ROWCOUNT
 GO
```

Przykład wywołania procedury z parametrami wyjściowymi

```
-- Wyłączamy wyświetlanie standardowego komunikatu "N row(s) affected".
-- To ustawienie normalnie obowiązuje dla bieżącego połączenia.
SET NOCOUNT ON
-- Deklarujemy zmienną do pobrania wartości parametru wyjściowego.
DECLARE @Var_Wiersze int
-- Wywołanie procedury. Po zmiennej, podanej jako parametr wyjściowy,
-- musi być umieszczone słowo "OUTPUT".
EXEC    Przedmiot_edycja 'INF407W', 'Bazy danych', 5, 4, '20000', 1,
        @Var_Wiersze OUTPUT
-- Wyświetlenie komunikatu tekstowego. Funkcja CONVERT dokonuje konwersji
-- z jednego typu danych na inny (tutaj z int do varchar(3)).
PRINT ('Zmieniono: ' + CONVERT(varchar(3), @Var_Wiersze) + ' wiersz(y).')
-- Przywracamy wyświetlanie komunikatu "N row(s) affected".
SET NOCOUNT OFF
GO
```

Literatura

1. ANLEY C., *Advanced SQL Injection In SQL Server Applications*, Next Generation Security Software Ltd (http://www.ngssoftware.com), 2002.
2. BEYNON-DAVIES P., *Systemy baz danych – nowe wydanie*, WNT, Warszawa 2003.
3. DATE C. J., DARWEN H., *SQL. Omówienie standardu języka*, WNT, Warszawa 2000 (książka dostępna w bibliotece WSIZ "Copernicus").
4. MICROSOFT, *Books On-Line* – dokumentacja systemu *MS SQL Server*, Microsoft Corp. 1988 – 2000.
5. RANKINS R., JENSEN P., BERTUCCI P., *Microsoft SQL Server 2000. Księga eksperta.*, HELION, Gliwice 2003 (książka dostępna w bibliotece WSIZ „Copernicus").
6. Strona ASP.NET: http://www.asp.net.
7. Strona MSDN: http://msdn.microsoft.com.
8. Strona PHP: http://www.php.net.
9. WAYMIRE R., SAWTELL R., *MS SQL Server 2000 dla każdego*, HELION, Gliwice 2002 (książka dostępna w bibliotece WSIZ "Copernicus").

Bazy danych

Wykład 5_2

Dziękuję za uwagę !