SQL

Wydanie II

Marcin Szeliga

Helion

TABLICE INFORMATYCZNE

STRUKTURALNY JĘZYK ZAPYTAŃ

Standard ANSI SQL3 oraz jego implementacje w serwerach SQL Server i PostgreSQL

Znaczniki strukturalnego języka zapytań można podzielić na dziewięć grup:

Dyrektywy

Koniec instrukcji

Serwery baz danych ignorują występujące w instrukcjach SQL znaki końca wiersza. Koniec instrukcji oznacza się symbolem `;`.

SQL Server

Oprócz kończącego instrukcje średnika występuje dyrektywa `GO` — powoduje ona wysłanie do serwera bazy danych i wykonanie wszystkich instrukcji, znajdujących się pomiędzy dyrektywami `GO`.

Wskazówki

1. Dyrektywa `GO` nie może występować w tym samym wierszu, w którym znajduje się jakakolwiek instrukcja języka SQL. W wierszu z dyrektywą `GO` można umieszczać jedynie komentarze.
2. Zakres zmiennych lokalnych ograniczony jest do instrukcji występujących pomiędzy poszczególnymi dyrektywami `GO`.
3. Następujące instrukcje muszą być oddzielone dyrektywą `GO` od pozostałych: `CREATE PROCEDURE`, `CREATE VIEW`, `CREATE TRIGGER`, `CREATE RULE`, `CREATE DEFAULT` oraz (o ile nie użyto dyrektywy `EXEC`) instrukcja wywołania procedur.
4. Dyrektywa `GO` jest poprawnie interpretowana tylko przez programy SQL Server Management Studio i sqlcmd.

Wywołanie procedur

Do wywołania procedury składowanej służy dyrektywa `CALL`. Umożliwia ona przekazanie parametrów do wywoływanej procedury i odczytanie wyniku jej działania.

SQL Server

Dyrektywa `CALL` została zastąpiona dyrektywą `EXEC` (`EXECUTE`). Oprócz wywołania procedury umożliwia ona wykonanie ciągu znaków, będącego instrukcją języka SQL (wykonywanie tworzonych dynamicznie instrukcji SQL).

Składnia:

```
[[EXEC [UTE]]
    {
        [@zwrócony_stan_wykonania =]
            {nazwa_procedury [;liczba] | @nazwa_
            procedury
    }
    [[@parametr =] {wartość | @zmienna [OUTPUT] |
    [DEFAULT]]
        [,...n]
[WITH RECOMPILE]
```

gdzie:

- *@zwrócony_stan_wykonania* jest opcjonalną zmienną lokalną, przechowującą informacje o stanie wykonania wywoływanej procedury lub funkcji,
- *@nazwa_procedury* jest nazwą zmiennej lokalnej, przechowującej nazwę wywoływanego obiektu,
- `OUTPUT` określa, że wywoływana procedura jest funkcją zwracającą jakąś wartość,
- `DEFAULT` przekazuje do procedury domyślne, określone podczas jej tworzenia wartości parametrów wywołania,
- `WITH RECOMPILE` wymusza ponowną kompilację kodu wywoływanej procedury lub funkcji, co wiąże się z obliczeniem nowego planu jej wykonania, bazującego na aktualnych statystykach.

PostgreSQL

Procedury napisane w językach proceduralnych mogą być wywołane dyrektywą `CALL`. Zwracające dane funkcje wywołuje się, podobnie jak w serwerze SQL Server, za pomocą instrukcji `SELECT`, przy czym funkcje zwracające wartości skalarne będą wywoływane w klauzuli `SELECT`, `WHERE`, `ORDER BY`, `GROUP BY` lub `HAVING`, a funkcje zwracające zbiory — w klauzuli `FROM`.

Funkcje

Funkcje języka SQL można podzielić na pięć kategorii:

1. Funkcje sterujące wykonaniem programu.
2. Funkcje skalarne, które zwracają pojedynczą wartość obliczoną na podstawie zera lub większej liczby argumentów skalarnych.
3. Funkcje grupujące, które zwracają pojedynczą wartość dla zbioru argumentów wywołania.
4. Zwracające zbiory danych funkcje tabelaryczne, do których można się odwoływać jak do tabel, w klauzuli `FROM` instrukcji `SELECT`.
5. Funkcje analityczne (funkcje rankingu), które umożliwiają przeprowadzenie obliczeń na zbiorach wierszy powiązanych z wynikiem bieżącego zapytania.

Decyzja o zaimplementowaniu poszczególnych funkcji należy do twórcy serwera baz danych, a zaimplementowane funkcje są opisane w dokumentacji danego serwera. Wyjątkiem jest ujęta w standardzie ANSI funkcja `CASE`.

Funkcja sterująca wykonaniem programu CASE

Sprawdza wartość argumentu wywołania i zwraca jeden z wielu predefiniowanych wyników.

Składnia:

```
CASE argument
    WHEN warunek THEN wynik
        [...n]
    [ ELSE wynik_n ]
END
```

Przykład (zapytanie zwracające opisowe nazwy rocznych dochodów klientów — prawidłowość wyników gwarantuje nam kolejność, w jakiej sprawdzane są poszczególne warunki):

```
SELECT FirstName, YearlyIncome, CASE
  WHEN YearlyIncome < 50000 THEN 'Niski'
  WHEN YearlyIncome < 75000 THEN 'Średni'
  WHEN YearlyIncome < 90000 THEN 'Wysoki'
  ELSE 'Bardzo wysoki'
  END AS Dochód
FROM dbo.CustomersHistory;
```

Identyfikatory

Identyfikatory określające obiekty i umożliwiające odwoływanie się do przechowywanych w nich danych lub do ich metod, zdarzeń czy właściwości muszą być zgodne z przyjętą konwencją nazewniczą:

- Identyfikatory mogą składać się z nie więcej niż 128 znaków.
- Pierwszym znakiem identyfikatora musi być litera alfabetu.
- Identyfikator nie może być słowem zastrzeżonym języka SQL.
- Identyfikator może zawierać litery, cyfry oraz symbole: `@`, `#`, `_`. Wynika z tego, że identyfikatory nie mogą zawierać spacji ani pozostałych symboli specjalnych (PostgreSQL zezwala na stosowanie znaku `$`).

Identyfikatory niezgodne z konwencją nazewniczą muszą być wyróżniane za pomocą cudzysłowów.

SQL Server

Identyfikatory powinny być wyróżniane za pomocą nawiasów kwadratowych.

Komentarze

Komentarze to ciągi znaków ignorowane przez kompilator. Tekst komentarza wyróżniany jest za pomocą znaków `/**/`. Kompilator ignoruje wszystkie znaki (z wyjątkiem dyrektywy `GO`) znajdujące się pomiędzy tymi znacznikami.

Zdefiniowanym w standardzie ANSI znakiem komentarza są dwa myślniki (`--`). Kompilator ignoruje znaki znajdujące się po prawej stronie myślników.

Operatory

Operatory pełnią rolę spójników języka SQL i choć w większości przypadków mogą być zastąpione odpowiednią funkcją, to ich użycie poprawia czytelność kodu programu. Funkcja każdego operatora zależy od kontekstu jego wystąpienia.

Operatory arytmetyczne

W języku SQL występują następujące operatory arytmetyczne:

- iloczyn (`*`),
- iloraz (`/`),
- modulo (`%`),
- suma (`+`),
- różnica (`-`).

Operator ciągów znaków

Jedynym operatorem ciągów znaków jest operator konkatenacji (`||`). Pozostałe operacje na ciągach znaków przeprowadzane są przez odpowiednie funkcje ciągu znaków.

SQL Server

Operatorem konkatenacji jest znak `+`. Ciągi znaków możemy łączyć również za pomocą funkcji `CONCAT`.

Operatory logiczne

Operatory logiczne wykorzystywane są najczęściej do łączenia warunków umieszczonych w klauzulach `WHERE` i `HAVING`. Występują trzy operatory logiczne:

- negacji (`NOT`), zwracający wartość `1` (prawda), jeżeli wyrażenie było fałszem, i `0` w przeciwnym wypadku,
- koniunkcji (`AND`), zwracający wartość `1` (prawda), jeżeli oba wyrażenia są prawdziwe,
- alternatywy (`OR`), zwracający wartość `1` (prawda), jeżeli którekolwiek z wyrażeń jest prawdziwe.

Wskazówki

W związku z występowaniem specjalnej wartości `NULL` w języku SQL obowiązuje logika trójwartościowa, a nie klasyczna logika dwuwartościowa. I tak:

- Wynikiem negacji wartości `NULL` jest wartość `UNKNOWN`.
- Jeżeli choć jeden argument koniunkcji jest wartością `NULL`, jej wynikiem będzie wartość `UNKNOWN`.
- Jeżeli choć jeden argument alternatywy jest wartością `NULL`, jej wynikiem będzie wartość `UNKNOWN`.

Operatory porównania

Do operatorów porównania języka SQL zaliczamy:

- równy (`=`),
- mniejszy niż (`<`),
- większy niż (`>`),
- mniejszy lub równy (`<=`),
- większy lub równy (`>=`),
- różny (`<>`),
- należy do zbioru (`IN`),
- należy do przedziału (`BETWEEN ... AND`),
- jest zgodny ze wzorcem (`LIKE`).

Operatory poziomu bitowego

Język SQL umożliwia wykonywanie podstawowych operacji na bitach:

- mnożenie logiczne (`&`),
- alternatywa logiczna (`|`),
- negacja logiczna (`~`).
- alternatywa wyłączna (operatorem `XOR` w serwerze SQL Server jest `^`, a w PostgreSQL — `#`).

PostgreSQL

Serwer PostgreSQL umożliwia dodatkowo wykonanie następujących operacji na poziomie bitów:

- przesunięcie w lewo (`<<`),
- przesunięcie w prawo (`>>`).

Hierarchia operatorów

Jeżeli w jednym wyrażeniu posłużono się kilkoma operatorami, serwer baz danych obliczy jego wartość w następującej (mogącej wpłynąć na wynik) kolejności:

1. Operatory umieszczone w nawiasach.
2. Mnożenie, dzielenie, modulo.
3. Dodawanie, odejmowanie.
4. Operatory porównania.
5. Operatory poziomu bitowego.
6. Konkatenacja.
7. Negacja.
8. Koniunkcja.
9. Alternatywa.
10. Przypisanie wartości zmiennej (`=`).

Słowa kluczowe

Słowa kluczowe są ciągami znaków, mających ściśle określone znaczenie i interpretowanych przez serwer baz danych w sposób charakterystyczny dla słowa kluczowego. Niedopuszczalne jest używanie słów kluczowych niezgodnie z określonymi dla nich zasadami syntaktycznymi (na przykład w niewłaściwej dla słowa klauzuli) oraz z określonymi dla nich zasadami semantycznymi (ich znaczeniem).

Wyrażenia

Wyrażenia są połączeniem symboli i operatorów, które reprezentują pojedynczą wartość. Typ reprezentowanych danych zależy od typu elementów wyrażenia.

Zmienne

W przypadku systemu SQL Server opisywane znaczniki zostały dodane do słownika języka T-SQL, w przypadku PostgreSQL wchodzą one w skład dodatkowego, osobno instalowanego języka PL/pgSQL.

Zmienne charakteryzują się nazwą, typem i wartością. Dzięki zmiennym możemy:

- przechowywać wartości zwrócone przez funkcje w celu ich późniejszego wykorzystania (w standardzie SQL funkcjonalnym odpowiednikiem zmiennych są podzapytania),
- wielokrotnie wykorzystać ten sam kod do wykonywania operacji na różnych danych wejściowych.

Instrukcja DECLARE

Wykonanie instrukcji spowoduje utworzenie zmiennej o określonej nazwie i typie. Wartość zadeklarowanej zmiennej domyślnie ustawiana jest na `NULL`.

Składnia:

```
DECLARE
    {{@zmienna [AS] typ } | [ = wartość ] }}
        | {@kursor CURSOR}
        | {typ_tabeli}
    } [,...n]
< typ_tabeli > ::=
    TABLE ({< definicja_kolumny > | < ograniczenie
    >} [,...]
          )
```

gdzie:

- `CURSOR` określa, że zmienna będzie kursorem.

Instrukcja SET

Nadaje wartość zdefiniowanej zmiennej.

Składnia:

```
SET @zmienna = wyrażenie
```

Przykład:

```
DECLARE @znajdź VARCHAR(30) = '%'
SET @znajdź = 'Fra%'
SELECT CompanyName, Address, Phone
FROM Customers
WHERE CompanyName LIKE @znajdź
```

Instrukcje sterujące wykonaniem programu

W zgodzie ze standardem ANSI SQL implementacje tego języka posiadają kilka znaczników sterujących wykonaniem programu. Są one wzorowane na językach proceduralnych.

W przypadku systemu SQL Server opisywane znaczniki zostały dodane do słownika języka T-SQL, w przypadku PostgreSQL wchodzą one w skład dodatkowego, osobno instalowanego języka PL/pgSQL.

Instrukcje BEGIN ... END

`BEGIN ... END` wydzielają zbiór instrukcji SQL traktowanych jako funkcjonalna całość (blok instrukcji). W serwerze PostgreSQL instrukcja `BEGIN` rozpoczyna transakcję.

Instrukcja GOTO

Powoduje zmianę sposobu wykonania programu i przejście do instrukcji oznaczonej etykietą. Instrukcje znajdujące się za poleceniem `GOTO` nie zostaną wykonane.

Instrukcja IF ... ELSE

Instrukcja `IF ... ELSE` pozwala na warunkowe wykonanie instrukcji SQL. Instrukcja zostanie wykonana, jeżeli warunek będzie spełniony, w przeciwnym wypadku może być wykonana instrukcja umieszczona w opcjonalnej klauzuli `ELSE`.

Instrukcja RETURN

Powoduje natychmiastowe przerwanie wykonywania procedury lub bloku instrukcji SQL.

Instrukcja WAITFOR

Określa czas, po którym określona instrukcja zostanie wykonana.

Instrukcja WHILE

Polecenie `WHILE` pozwala na wielokrotne (aż do momentu spełnienia określonego warunku logicznego) wykonanie instrukcji SQL. Dodatkowo możemy sterować wykonaniem instrukcji za pomocą słów kluczowych `BREAK` i `CONTINUE`, gdzie:

- `BREAK` powoduje natychmiastowe przerwanie wykonywania instrukcji SQL,
- `CONTINUE` powoduje restart pętli `WHILE`. Instrukcje znajdujące się za poleceniem `CONTINUE` nie zostaną wykonane.

Instrukcje THROW oraz TRY ... CATCH

Dostępna w serwerze SQL instrukcja `THROW` pozwala zgłosić błąd o podanym numerze lub zgłosić własny komunikat błędu. Jeżeli błąd został zgłoszony podczas wykonywania instrukcji znajdującej się w bloku `BEGIN TRY ... END TRY`, sterowanie wykonaniem programu zostanie przekazane do bloku `BEGIN CATCH ... END CATCH`.

TYPY DANYCH

Standard ANSI SQL-3 definiuje następujące typy danych (przedstawione zostały jedynie najczęściej stosowane typy proste):

Typy liczbowe

`BIGINT` reprezentuje liczby całkowite z zakresu od -2^{63} ($-9\,223\,372\,036\,854\,775\,808$) do $2^{63}-1$ ($9\,223\,372\,036\,854\,775\,807$).

`INT` reprezentuje liczby całkowite z zakresu od -2^{31} ($-2\,147\,483\,648$) do $2^{31}-1$ ($2\,147\,483\,647$).

`SMALLINT` reprezentuje liczby całkowite z zakresu od -2^{15} ($-32\,768$) do $2^{15}-1$ ($32\,767$).

`TINYINIT` (SQL Server) reprezentuje liczby całkowite z zakresu od 0 do 255.

`DECIMAL` reprezentuje liczby o określonej skali i precyzji z zakresu od $-10^{38}+1$ do $10^{38}-1$. Typ `NUMERIC` jest synonimem typu `DECIMAL`.

`FLOAT` reprezentuje liczby o zmiennej precyzji z zakresu od $1{,}79E+308$ do $1{,}79E+308$.

`REAL` reprezentuje liczby o zmiennej precyzji z zakresu od $-3{,}40E+38$ do $3{,}40E+38$. Typy zmiennoprzecinkowe charakteryzują się tym, że wartości tak zapisanych danych przechowywane są z pewną skończoną dokładnością.

`BIT` (SQL Server) reprezentuje 0 lub 1. Ten typ często stosowany jest jako odpowiednik typu logicznego niezaimplementowanego w tym serwerze.

Typy daty i czasu

`DATETIME` (SQL Server) reprezentuje datę i czas z zakresu od 1 stycznia 1753 do 31 grudnia 9999, z dokładnością do jednej milisekundy.

`SMALLDATETIME` (SQL Server) reprezentuje datę i czas z zakresu od 1 stycznia 1900 do 6 czerwca 2079, z dokładnością do jednej minuty.

`DATETIME2` (SQL Server) reprezentuje datę i czas z zakresu od 1 stycznia 0 roku do 31 grudnia 9999 z określoną, nie większą niż 100 nanosekund, dokładnością.

`DATE` reprezentuje datę z dokładnością do jednego dnia.

`TIME` reprezentuje czas (SQL Server) i pozwala określić jego dokładność — maksymalna wynosi 100 nanosekund.

`TIMESTAMP` reprezentuje datę i czas.

Typy znakowe

`CHAR` reprezentuje ciąg znaków o określonej długości.

`NCHAR` (SQL Server) reprezentuje ciąg znaków o określonej długości zapisanej w postaci Unicode.

`VARCHAR` reprezentuje ciąg znaków o zmiennej długości.

`NVARCHAR` (SQL Server) reprezentuje ciąg znaków o zmiennej długości zapisanej w postaci Unicode.

`TEXT` reprezentuje ciąg znaków o określonej, nie większej niż $2^{31}-1$ ($2\,147\,483\,647$), długości. W serwerze SQL dane znakowe o długości przekraczającej 8 KB powinny być reprezentowane za pomocą typów `VARCHAR (MAX)` lub `NVARCHAR(MAX)`.

Ciąg dalszy na stronie 2 ➔

Dalsza część książki dostępna w wersji pełnej.