SQL Wydanie III

TABLICE INFORMATYCZNE • Marcin Szeliga

Helion

STRUKTURALNY JĘZYK ZAPYTAŃ (SQL)

SQL (ang. *Structured Query Language*) jest jednym z najpopularniejszych języków programowania i najczęściej używanym językiem w pracy z danymi. Sukces ten język SQL zawdzięcza:

- Swojej prostocie. Język SQL jest językiem deklaratywnym, czyli zamiast krok po kroku określać, jak serwer bazodanowy ma wykonać określone polecenie, użytkownik deklaruje, w języku przypominającym potoczny angielski, spodziewany wynik. W przeciwieństwie do języków proceduralnych *SQL pozwala określić wynik, nie sposób jego osiągnięcia.*
- Wydajności i skalowalności. Serwery relacyjnych baz danych są w stanie w czasie rzeczywistym przetwarzać terabajty danych, o ile tylko użyte zostaną poprawne instrukcje języka SQL. Możliwe jest to dzięki wyjątkowej cesze języka SQL — w odróżnieniu od innych języków *SQL umożliwia przetwarzanie wielu rekordów za pomocą pojedynczej instrukcji.* Instrukcje języka SQL przed wykonaniem są interpretowane przez serwer bazodanowy. Wynikiem tej interpretacji jest plan wykonania instrukcji, który następnie jest realizowany przez serwer.

Relacyjny model baz danych

Bazą danych jest zbiór informacji o ściśle określonej strukturze — dane te są przechowywane w tabelach i są uzupełnione o metadane (dane o samych tabelach, np. typach ich kolumn). W modelu relacyjnym dane są przechowywane w wielu odrębnych, ale powiązanych ze sobą tabelach — dwuwymiarowych obiektach zbudowanych z pionowych kolumn i poziomych wierszy, na których przecięciu umieszczone są komórki.

W poszczególnych kolumnach tabeli znajdują się zawsze dane tego samego typu (np. ceny), z kolei w poszczególnych wierszach znajduje się komplet informacji o konkretnych obiektach (np. o towarze). Ważną cechą tabeli jest to, że każdy jej rekord składa się z takiej samej liczby pól (kolumn).

Niezbędna dla opanowania języka SQL jest umiejętność wyobrażenia sobie tabel jako zbiorów rekordów. W zbiorze kolejność elementów nie ma znaczenia, a więc kolejność wierszy i kolumn tabeli również jest bez znaczenia. Ponadto zbiory nie zawierają kopii tego samego elementu. Ponieważ powtórzenie w tabeli tego samego rekordu prowadzi do niespójności danych, przyjęło się dodawać do tabeli specjalną kolumnę, w której zapisuje się identyfikatory poszczególnych wierszy. Taka kolumna nazywa się **kluczem podstawowym** tabeli.

Znaczniki języka

Instrukcje języka SQL zapisuje się za pomocą słów i symboli (znaczników) o ściśle określonym znaczeniu. Znaczniki strukturalnego języka zapytań można podzielić na dziewięć grup.

Dyrektywy sterujące wykonaniem instrukcji

Serwery baz danych ignorują występujące w instrukcjach SQL znaki końca wiersza — dla poprawy czytelności instrukcji przyjęło się zapisywać ich poszczególne klauzule w nowych wierszach. Koniec instrukcji oznacza się symbolem `;`.

Do wywołania procedury składowanej służy dyrektywa `CALL`. Umożliwia ona przekazanie parametrów do wywoływanej procedury i odczytanie wyniku jej działania. W serwerze SQL Server dyrektywa `CALL` została zastąpiona dyrektywą `EXEC` (`EXECUTE`). Oprócz wywołania procedury umożliwia ona wykonywanie ciągu znaków, będącego instrukcją języka SQL (wykonywanie tworzonych dynamicznie instrukcji SQL).

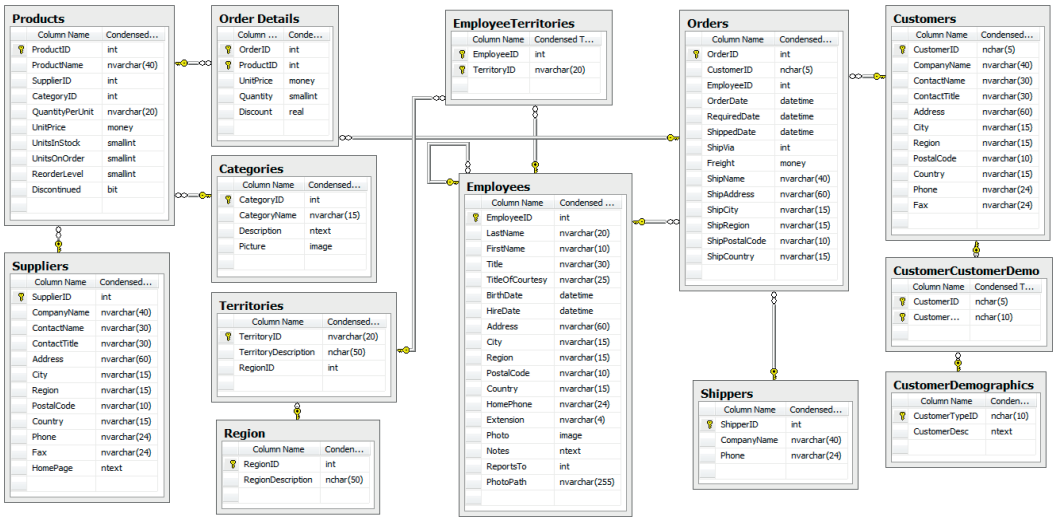
Składnia EXECUTE

```
[[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.

Schemat przykładowej bazy danych Northwind udostępnianej przez firmę Microsoft.



Identyfikatory

Identyfikatory określające obiekty (np. tabele) 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.

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

Komentarze

Komentarze to ciągi znaków ignorowane przez kompilator. Tekst komentarza jest wyróżniany za pomocą znaków `/**/`. Kompilator ignoruje wszystkie znaki znajdujące się między tymi znacznikami.

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

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 (np. w niewłaściwej dla słowa klauzuli) oraz niezgodnie z określonymi dla nich zasadami semantycznymi (ich znaczeniem).

Operatory

Operatory odgrywają 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 takie oto operatory arytmetyczne:

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

Operator ciągów znaków

Jedynym operatorem ciągów znaków jest operator konkatenacji `||` (w serwerze SQL Server operatorem konkatenacji jest znak `+`). Pozostałe operacje na ciągach znaków są przeprowadzane przez odpowiednie funkcje ciągu znaków.

Operatory logiczne

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

- Koniunkcji (`AND`), zwracający wartość `1` (prawda), jeżeli oba wyrażenia są prawdziwe. Operator `AND` jest operatorem dwuargumentowym. W klasycznej logice koniunkcja jest prawdziwa tylko wtedy, gdy oba argumenty są prawdziwe, w pozostałych przypadkach jej wynikiem jest fałsz. Wszystkie kombinacje parametrów tego operatora możliwe w języku SQL zostały przedstawione poniżej.

A	B	A AND B
TRUE	TRUE	TRUE
TRUE	FALSE	FALSE
TRUE	NULL	UNKNOWN
FALSE	TRUE	FALSE
FALSE	FALSE	FALSE
FALSE	NULL	FALSE
NULL	TRUE	UNKNOWN
NULL	FALSE	FALSE
NULL	NULL	UNKNOWN

- Alternatywy (`OR`), zwracający wartość `1` (prawda), jeżeli którekolwiek z wyrażeń jest prawdziwe. Operator `OR` jest operatorem dwuargumentowym. W klasycznej logice alternatywa jest nieprawdziwa tylko wtedy, gdy oba jej argumenty są nieprawdziwe, w pozostałych przypadkach wynikiem alternatywy logicznej jest prawda. Wszystkie kombinacje parametrów tego operatora możliwe w języku SQL zostały przedstawione poniżej.

A	B	A OR B
TRUE	TRUE	TRUE
TRUE	FALSE	TRUE
TRUE	NULL	TRUE
FALSE	TRUE	TRUE
FALSE	FALSE	FALSE
FALSE	NULL	UNKNOWN
NULL	TRUE	TRUE
NULL	FALSE	UNKNOWN
NULL	NULL	UNKNOWN

- Negacji (`NOT`), zwracający wartość `1` (prawda), jeżeli wyrażenie było fałszem, i `0` w przeciwnym razie. Operator `NOT` jest operatorem jednoargumentowym. W klasycznej logice jego wynikiem jest zaprzeczenie (negacja) argumentu. W języku SQL może on również zwrócić wartość `UNKNOWN`.

A	NOT A
TRUE	FALSE
FALSE	TRUE
NULL	UNKNOWN

Wyrażenia

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

Zmienne

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 wykorzystywać ten sam kod do wykonywania operacji na różnych danych wejściowych.

Instrukcja DECLARE

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

Składnia DECLARE

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

gdzie:

- `CURSOR` określa, że zmienna będzie kursorem.

Instrukcja SET

Nadaje wartość zdefiniowanej zmiennej.

Składnia SET

```
SET @zmienna = wyrażenie
```

Przykład SET

```
DECLARE @znajdź VARCHAR(30) = '%'
DECLATE @wynik VARCHAR(50)
SET @znajdź = 'Fra%'
SELECT @wynik = Address
FROM Customers
WHERE CompanyName LIKE @znajdź;
```

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.