SQL – Chakterystyka składni

Instrukcja SQL zaczyna **się poleceniem (słowem kluczowym)** określającym operację, którą zamierzamy wykonać. Po słowie kluczowym mogą znaleźć się **klauzule** (dookreślenia dla słowa kluczowego).

Instrukcje języka SQL, bez względu na ich pogrupowanie czy implementacje, zostaną omówione na przykładzie baz danych **PostgreSQL**, **MySQL**, **Access**.

MySQL jest SZBD bardzo często spotykanym w usługach hostingowych.

PostgreSQL jest najbardziej zaawansowanym serwerem baz danych z publikowanym kodem źródłowym pozwalającym na programowanie proceduralne w języku PL/pgSQL podobnym do języka PL/SQL w bazie Oracle. PostgreSQL jest przy tym obiektowo-relacyjnym systemem zarządzania bazą danych. Cechy te sprawiają, że PostgreSQL jest bardzo atrakcyjnym systemem zarządzania bazą danych, ponieważ może służyć jako doskonały przykład podczas zajęć edukacyjnych.

Access to baza danych dość często wykorzystywana jako składnik pakietu biurowego Microsoft Office i w związku z tym można spotkać ją w biurach i innych mniej zaawansowanych zastosowaniach. Jest chętnie wybierana, ponieważ nie wymaga znajomości języka SQL i jest prosta w obsłudze.

Domeny
Podczas tworzenia tabeli definiujemy również typy atrybutów, np. tworzymy dla wybranej kolumny identyfikator: wiek z zamiarem umieszczenia w niej wieku osób wpisanych na listę promocji towaru.

Wartości, jakie będziemy umieszczać w kolumnie, będą to liczby całkowite z przedziału 18-110 (zakładamy, że produkt adresujemy do osób pełnoletnich). W bazie danych zdefiniujemy taki typ jako **int(3)**, co oznacza maksymalnie trzycyfrowe liczby całkowite. **To, jakie wartości mogą przyjąć umieszczane w kolumnie dane, nazywane jest dziedziną lub domeną**.

Zbiór dopuszczalnych wartości dla co najmniej jednego atrybutu nazywamy domeną.

Definicja wskazuje, iż **w obrębie jednej relacji kilka atrybutów może mieć różne domeny** (każdy atrybut na inną domenę - tekst, liczby całkowite, daty), może też się zdarzyć, że wszystkie atrybuty w relacji będą odnosić się do jednej domeny (jako zbioru dopuszczalnych wartości).

Domena definiuje zbiór dopuszczalnych wartości atrybutu niezależnie od czasu.

Podobnie jak w każdym języku programowania, podstawą elementów języka SQL są pojedyncze znaki. Zaliczamy do nich:
- **wielkie litery A-Z**,
- **małe litery a-z**,
- **cyfry 0-9**,
- **zbiór znaków specjalnych: ()* + >-• / :; = ,<>?_ |**.

Choć implementacje SQL mogą się od siebie różnić, wszystkie muszą obsługiwać zbiór znaków o nazwie SQL_TEXT.

Tabela 14.1. Pięć głównych kategorii składniowych SQL

Lp.	Identyfikatory	Nazwy obiektów	Przykłady		
1	Literały	Stałe			
2	Operatory	Spójniki	Arytmetyczne – dla tych wyrażeń można używać nawiasów ( )	iloczyn	*
				iloraz	/
				modulo	%
				suma	+
				różnica	-
			Znakowe	konkatencja	+ lub ||
				symbol wieloznaczny (zastępujący dowolną liczbę znaków)	%
				symbol wieloznaczny (zastępujący jeden znak)	_
			Logiczne	koniunkcja	AND
				alternatywa	OR
				negacja	NOT
			Porównania	równy	=

		negacja	NOT
	Porównania	równy	=
		mniejszy	<
		większy	>
		mniejszy lub równy	<=
		większy lub równy	>=
		różny	!= lub <>
	Typowe dla SQL operatory relacji	operator przynależności do zbioru przedziału domkniętego	BETWEEN *n* AND *n*
		operator przynależności do zbioru	IN (...)
		operator dopasowania do wzorca	LIKE
		operator dopasowania do wyrażenia regularnego	REGEXP, RLIKE
		operator porównania sprawdzający występowanie znacznika braku wartości NULL	IS NULL

Lp.	Identyfikatory	Nazwy obiektów	Przykłady	
3	Słowa kluczowe	Wyrazy interpretowane przez serwer w określony sposób	Słowa kluczowe to zarezerwowane – podobnie jak w językach programowania – ciągi znaków. Możemy do nich zaliczyć:	
			Instrukcje	CREATE, SELECT
			Klauzule	WHERE, JOIN
			Nazwy typów danych	INTEGER, CHAR
			Nazwy funkcji systemowych	ISNULL, ABS
			Nazwy zarezerwowane do użycia w przyszłości	zależnie od bazy danych
4	Komentarze	Ignorowane	Podwójny znak myślnika (-- –)to komentarz, którego używamy, aby określony wiersz nie był interpretowany.	
			Jeśli chcemy, aby został zignorowany więcej niż jeden wiersz, warto posłużyć się komentarzem, który odnosi się do bloku instrukcji: /* tu znajduje się blok instrukcji */	
5	Operator konkatencji		Operator ten pozwala łączyć atrybut z: • innym atrybutem, • literałem, • wyrażeniem arytmetycznym, • wartością stałą. Na podstawie tej operacji tworzona jest jedna wynikowa kolumna.	

Słowa kluczowe

SK to sformułowanie „zastrzeżone" dla BD, np. nie możemy kolumny nazwać np. **SELECT**. W związku z tym należy zapoznać się nie tylko ze słowami kluczowymi standardu SQL. lecz także implementacjami tego języka funkcjonującymi w SZBD: MySQL. PostgreSQL, Access itp.

Słowo kluczowe **DISTINCT**. Podczas pracy z bazą danych możemy obsługiwać tabele zawierające powtarzające się wartości tego samego atrybutu w kolejnych krotkach. DISTINCT umożliwia eliminowanie duplikatów, wartości kolejnych wierszy oraz kolumn- w zależności od użycia.

Klauzula **ORDER BY**. To klauzula służąca uporządkowaniu zwracanego wyniku. Pamiętać należy, iż stosujemy ją jako ostatnią klauzulę polecenia SELECT. Dane segregowane (sortowane) są domyślnie rosnąco lub według ustawień bazy danych.

Operator **BETWEEN...AND**. Używany jest do sprawdzenia, czy wartość znajduje się w podanym przedziale.

Operator **IN**. Służy do sprawdzenia, czy wartość znajduje się na zdefiniowanej liście. Operator **LIKE** służy do wybierania wartości, które odpowiadają wcześniej ustalonemu wzorcowi. Wzorzec ustalamy za pomocą symboli:

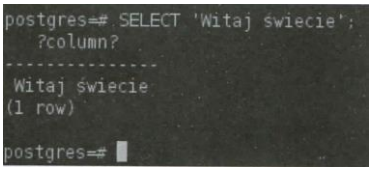
- % - odpowiada dowolnemu ciągowi znaków,
- _(podkreślnik) - odpowiada jednemu dowolnemu znakowi.

Operator **IS NULL** służy do wyszukiwania wartości NULL.

Literały

W językach programowania często zachodzi potrzeba posługiwania się literałami definiowanymi jako stałe dosłowne. Podajmy prosty przykład.
Jeśli napisalibyśmy program, którego celem byłoby wyświetlenie napisu: ***" Witaj świecie" (Hello World)***, posłużylibyśmy się następującym kodem SQL:

SELECT 'Witaj świecie';
W bazie PostgreSQL wynik działania takiej instrukcji



W języku SQL dostępne są następujące rodzaje literałów:

- napisy, ciągi znaków w pojedynczych cudzysłowach, np. 'ulica', '1410N';
- zapisy bitowe poprzedza się literą B i umieszcza w cudzysłowach, np. B '1101';
- napisy szesnastkowe poprzedza się literą X, np. X 'FA';
- dokładne wartości liczbowe zapisywane jako liczba dziesiętna - ze znakiem lub z możliwością zastosowania kropki dziesiętnej, np. -25.3 lub 0.07.

Typy danych

Typ danych określa, jakiego rodzaju informacje mogą być przechowywane w poszczególnych kolumnach tabel lub w zmiennych oraz jakiego typu dane mogą być przekazywane jako parametry wywołania procedury lub funkcji. Listę typów danych zdefiniowanych w standardzie SQL3 zawiera poniższa

Kategoria	Przykładowe typy danych	Opis
Typy liczbowe	INTEGER (INT), SMALLINT	Reprezentują liczby całkowite.
	NUMERIC (DECIMAL)	Reprezentuje liczby o określonej skali i precyzji.
	REAL	Reprezentuje liczby o zmiennej precyzji.
Typy daty i czasu	DATE	Reprezentuje datę.
	TIME	Reprezentuje czas.
Typy znakowe	CHAR	Reprezentuje ciąg znaków o określonej długości.
	VARCHAR	Reprezentuje ciąg znaków o zmiennej długości.
	NCHAR, NVARCHAR	Reprezentują ciąg znaków o stałej lub zmiennej długości zakodowanych w UNICODE.
Typy binarne	BINARY	Reprezentuje ciąg bitów o określonej długości.
	VARBINARY	Reprezentuje ciąg bitów o zmiennej długości.
	BLOB	Reprezentuje duże obiekty binarne.
Dokumenty XML	XML	Reprezentuje całe dokumenty XML.

INTEGER (INT) – 4B
SMALLINT – 2B
BIGINT – 8B
REAL – 1E-37 – 1E+37
DOUBLE – 1E-307 – 1E+308
NUMERIC (8 - PRECYZJA,2 - SKALA)
np.cena

Literały – ciąg znaków użyty pomiędzy cudzysłowami

Typ wyliczeniowy
Kolumna może zawierać również wartości typu wyliczeniowego **ENUM**. W takim wypadku możemy zdefiniować wartości typu jako np. PONIEDZIAŁEK, WTOREK, ŚRODA, CZWARTEK, PIĄTEK, SOBOTA, NIEDZIELA. Aby utworzyć własny typ wyliczeniowy, możemy posłużyć się następującą instrukcją SQL:

```
CREATE TYPE tydzień AS ENUM
('PONIEDZIAŁEK','WTOREK','ŚRODA','CZWARTEK','PIĄTEK',
'SOBOTA','NIEDZIELA');
```

Warto zapamiętać, że wartości typu wyliczeniowego umieszczamy w pojedynczych cudzysłowach w trakcie definiowania typu. Istotne jest również to, iż wartości te są czułe na wielkość liter. Wprowadzenie wartości 'piątek' będzie oznaczało, iż jest to wartość inna niż np. 'Piątek'.

Podczas tworzenia tabeli możemy określić kolumnę nowo utworzonym typem, np.
CREATE TABLE przepracowane_dni (dni tydzień);
Typ wyliczeniowy określa wartość PONIEDZIAŁEK jako najmniejszą, a wartość NIEDZIELA jako największą, umożliwiając porównywanie tych wartości z innymi.

Typ danych geometrycznych

Używa się go do reprezentowania obiektów dwuwymiarowych.
• PUNKT (point) reprezentuje punkt za pomocą dwóch współrzędnych: x,y.
• LINIA (line) definiowana jest za pomocą dwóch punktów: np. A (Xa,Ya) B (Xb,Yb).

Wartość NULL

Zgodnie z jednym z postulatów dra Codda serwery bazodanowe powinny spójnie przetwarzać wartość specjalną **NULL**.

Wartość NULL reprezentuje brakujące, nieznane lub nieistotne dane i jest różna od 0 oraz od pustego ciągu znaków. Na przykład brak ceny produktu nie oznacza, że jest on darmowy, a tylko że jego cena nie została jeszcze ustalona. Z powodu występowania wartości NULL w serwerach bazodanowych obowiązuje logika trójwartościowa, a nie dwuwartościowa. Porównanie wartości NULL z dowolną
inną wartością daje więc w wyniku wartość nieznaną (ang.Unknown), a nie prawdę lub fałsz.

Reguły przetwarzania wartości NULL są następujące:

- Dwie wartości NULL nie są ani sobie równe, ani różne od siebie. Wartość NULL nie jest też równa, mniejsza czy większa od jakiejkolwiek innej wartości. Sensowne wyniki daje jedynie sprawdzanie (za pomocą operatora IS), czy dana wartość jest nieznana .
- Wynikiem wszystkich operacji zawierających NULL jest wartość NULL, co pokazuje poniższy przykład:

```
SELECT NULL/0, 'Ala' + NULL, 5 + NULL, 10 * NULL, NULL + NULL;
-------------------------------------------------------------
NULL          NULL          NULL          NULL          NULL
```

- Wartość NULL jest ignorowana przez wszystkie funkcje grupujące z wyjątkiem funkcji COUNT(*).

Identyfikatory czyli nazwy
Obiekty bazy danych, takie jak baza, tabela, kolumna(atrybutu), muszą mieć niepowtarzalną nazwę, tzn. swój identyfikator, który musi być zgodny ze standardem języka SQL.

Identyfikatory:
• nie mogą mieć więcej niż 128 znaków;
• mogą zawierać litery, cyfry i symbole: @ $ # (pozostałe symbole, takie jak znak spacji, są niedozwolone);
• identyfikatory nie mogą zaczynać się cyfrą, natomiast identyfikatory zaczynające się symbolem @ oznaczają zmienną, a zaczynające się # oznaczają obiekt tymczasowy.

Identyfikatory nie mogą być również słowami kluczowymi języka SQL.

Identyfikatory czyli nazwy
Obiekty bazy danych, takie jak baza, tabela, kolumna(atrybutu), muszą mieć niepowtarzalną nazwę, tzn. swój identyfikator, który musi być zgodny ze standardem języka SQL.

1. nazwa nie może zawierać polskich „ogonków”
2. nazwa nie może zawierać spacji(podkreślniki są najbezpieczniejsze) nazw_ucz nazwUcz
3. nazwa nie może zaczynać się od cyfry
4. nazwa powinna być wpisywana małymi literami

Aby ułatwić sobie pracę, zwłaszcza projektując bazy danych, które w przyszłości miałyby być obsługiwane przez innych ludzi, warto stosować ustalone konwencje nadawania identyfikatorów. Takie postępowanie sprawi, że tworzone tabele i kod SQL będą bardziej czytelne dla innych i dla nas. Nazwy identyfikatorów powinny być możliwie krótkie, jednoznacznie opisujące obiekt.
Gdy identyfikator składa się z kilku wyrazów, powinny być pisane bez spacji, każdy wyraz wielką literą, z wyjątkiem pierwszego, np. **przykładIdentyfikatoraTabeli**.
przy_ident_tab
Gdy tworzymy funkcje, procedury lub wyzwalacze, powinniśmy używać przedrostków:
• udf (user defie function) - funkcja zdefiniowana przez użytkownika;
• usp (user storę procedurę) - procedura zdefiniowana przez użytkownika;
• v (view) - widok;
• tr (trigger) ~ wyzwalacz.

Podczas tworzenia identyfikatorów w PostgreSQL należy zachować ostrożność w dobieraniu wielkości liter. Identyfikatory, w których użyjemy na przemian wielkiej i małej litery, wymagają cudzysłowu. Gdy utworzymy tabelę o nazwie: **kLiEnCi**, będziemy zmuszeni do identycznego wpisania takiej nazwy w cudzysłowie, np.
SELECT * FROM "kLiEnCi";
Gdy w przeciwieństwie do powyższego przykładu dobierzemy konsekwentnie wielkość znaków dla identyfikatora tabeli, stosując wyłącznie małe litery, np.
CREATE TABLE klienci;
będziemy mogli wskazywać na tabelę przy użyciu dużych liter, małych liter lub dowolnej kombinacji dużych i małych liter na przemian, np.
SELECT * FROM kcLiEnCi;
SELECT * FROM klienci;
SELECT * FROM KLIENCI;
Wszystkie trzy powyższe instrukcje zadziałają bez konieczności użycia cudzysłowu na identyfikatorze tabeli.

Wielkość identyfikatorów w MySQL
Podczas tworzenia tabel w MySQL rozróżnianie wielkich i małych liter jest obowiązkowe. Do tabel odwołujemy się zaś, podając identyfikator identyczny pod względem wielkości liter z tym, który ma tabela.
Jeśli nazwiemy tabelę np. klienci, wówczas skuteczne wywołanie tabeli nastąpi jedynie przez umieszczenie takiej samej nazwy w instrukcji, np.
SELECT * FROM klienci;
Analogicznie, jeśli nazwa tabeli to Klienci, wówczas identyczną nazwę będziemy musieli umieścić w instrukcji SQL, np.
SELECT * FROM Klienci;
W przeciwieństwie do PostgreSQL nie używamy cudzysłowu, jedynie zmuszeni jesteśmy do konsekwentnego dobierania nazw i stosowania się do raz przyjętej nomenklatury.