Wybrane funkcje serwera MySQL

MySQL może zrobić znacznie więcej niż tylko przechowywanie i pobieranie danych. Możemy także **wykonywać manipulacje na danych** przed pobraniem lub zapisaniem. To tam gdzie MySQL Pojawiają się funkcje. Funkcje to po prostu fragmenty kodu, które wykonują pewne czynności następnie zwróć wynik. Niektóre funkcje akceptują parametry, podczas gdy inne nie akceptują parametrów.

Wbudowane funkcje serwerów baz danych można podzielić na trzy kategorie:

1. Funkcje skalarne — zwracają pojedynczą wartość obliczoną na podstawie zera lub większej liczby prostych argumentów.

2. Funkcje grupujące — zwracają pojedynczą wartość dla zbioru argumentów wywołania.

3. Funkcje typu RowSet — zwracające dane w postaci tabelarycznej, do których odwołujemy się tak jak do tabel.

Na podstawie typu parametrów wywołania funkcje skalarne można podzielić na:

1. Funkcje tekstowe operujące na ciągach znaków.

2. Funkcje liczbowe operujące na liczbach.

3. Funkcje daty i czasu operujące na danych typu data i godzina.

4. Funkcje konwersji służące do zmiany typu danych.

5. Na specjalne wyróżnienie zasługują funkcje kryptologiczne, które pozwalają zaszyfrować, odszyfrować i podpisać wiadomość oraz sprawdzić jej autentyczność.

W języku SQL funkcje można zagnieżdżać do dowolnego poziomu. Funkcje najbardziej wewnętrzne obliczane są w pierwszej kolejności, a na podstawie ich wyników obliczane są funkcje zewnętrzne.

Funkcje tekstowe zwracające tekst

CONCAT()

Funkcja łączy (konkatenuje) przekazane jako parametr i oddzielone przecinakami ciągi znaków. Używaliśmy już tej funkcji, a w następnych listingach użyjemy jej jeszcze kilka razy.

UPPER()

Wynikiem działania funkcji UPPER() jest ciąg znaków podany jako argument, ale składający się wyłącznie z wielkich liter. Za pomocą tej funkcji wszystkie małe litery argumentu zostaną zamienione na wielkie.

LEFT()

Za pomocą funkcji LEFT() z podanego jako argument ciągu znaków zostanie wycięta określona liczba znaków, począwszy od lewej strony

Przykład zagnieżdżania funkcji — najpierw odczytywane są pierwsza litera imienia i nazwiska, a następnie są one łączone w jeden ciąg, np.

SELECT CONCAT(LEFT(imie,1), LEFT(nazwisko,1))

FROM klient;

RIGHT()

Za pomocą funkcji RIGHT() z podanego jako argument ciągu znaków zostanie wycięta określona liczba znaków, począwszy od prawej strony. Np.

ELECT imie, nazwisko

FROM klient

WHERE RIGHT(klient,2) = 'rd';

SUBSTRING()

W wyniku działania funkcji SUBSTRING() zostanie zwrócona określona liczba znaków z łańcucha tekstowego, począwszy od podanej pozycji. Jeżeli nie podamy liczby zwracanych znaków, zwrócone będą wszystkie znaki występujące po pozycji określonej przez drugi parametr.

REVERSE ()

Funkcja REVERSE () zwraca ciąg znaków będący palindromem argumentu wywołania, czyli ciągiem znaków o odwróconej kolejności liter

```
SELECT `Imie`,`Nazwisko`,reverse(`Imie`) as odwrocone FROM `klient`;
```

Np. LENGTH ()

Funkcja LENGTH() jako wynik zwraca długość ciągu znaków podanego jako parametr jej wywołania. Jeżeli wywołamy funkcję LENGTH() z wartością Null, funkcja zwróci wartość Null. Za pomocą poniższej instrukcji wyświetlimy tylko te opisy towarów, które mają długie (dłuższe niż 10-znakowe) nazwy

LENGTH ()

Funkcja LENGTH() jako wynik zwraca długość ciągu znaków podanego jako parametr jej wywołania. Jeżeli wywołamy funkcję LENGTH() z wartością Null, funkcja zwróci wartość Null. Za pomocą poniższej instrukcji wyświetlimy tylko te opisy towarów, które mają długie (dłuższe niż 10-znakowe) nazwy

Funkcje liczbowe

Funkcje liczbowe pozwalają wykonywać dodatkowe (oprócz dodawania, odejmowania czy mnożenia) operacje matematyczne oraz formatować liczby.

ROUND()

Działanie funkcji ROUND() polega na zaokrągleniu liczby do określonej liczby cyfr po przecinku. Pierwszy parametr jest liczbą do zaokrąglenia, drugi wskazuje, do ilu pozycji chcemy ją zaokrąglić. Ujemna wartość powoduje zaokrąglenie liczby z lewej strony przecinka; 0 spowoduje zaokrąglenie do najbliższej liczby całkowitej. Jeżeli drugi parametr nie jest podany, serwer baz danych przyjmuje domyślnie jego wartość jako równą 0

TRUNCATE()

Funkcja TRUNCATE() powoduje obcięcie liczby do określonej liczby cyfr po przecinku. Pierwszy parametr jest liczbą do obcięcia, drugi wskazuje, do ilu pozycji chcemy liczbę skrócić. Ujemna wartość powoduje dodanie określonej liczby zer z lewej strony przecinka. Jeżeli drugi parametr nie jest podany, MySQL przyjmuje domyślnie jego wartość jako równą 0

```
SELECT Truncate(3.1415926535897,4);
```

Truncate(3.1415926535897,4)
3.1415

ABS()

Wynikiem działania funkcji ABS() jest wartość bezwzględna liczby (liczba bez znaku).

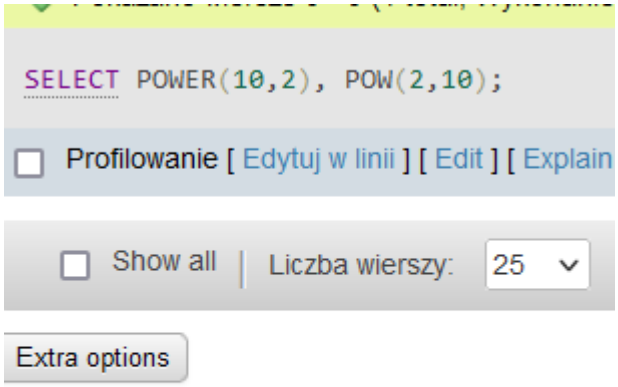
MOD()

Funkcja MOD() zwraca jako wynik wartość reszty po podzieleniu liczby podanej jako pierwszy parametr przez dzielnik podany jako drugi parametr wywołania funkcji. Jeżeli wartość drugiego parametru wynosi 0, zwracana jest liczba podana jako pierwszy parametr

POWER(), POW()

Funkcja POWER() sprawia, że liczba podana jako pierwszy parametr zostanie podniesiona do potęgi podanej jako drugi parametr. Wartości drugiego parametru mogą być mniejsze niż zero

```
SELECT POWER(10,2), POW(2,10);
```



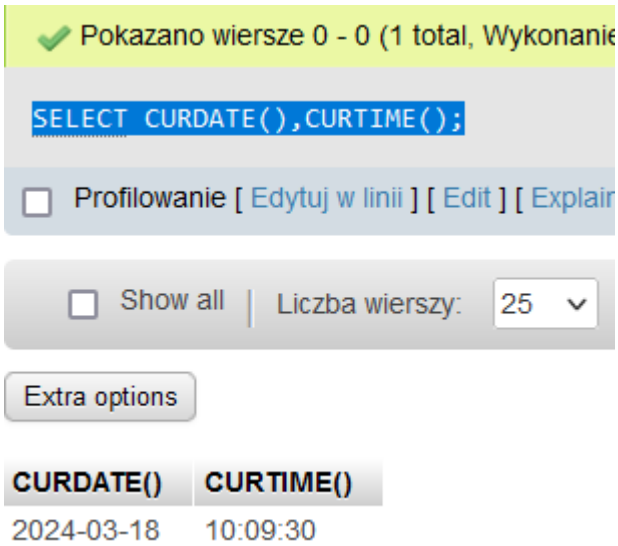
POWER(10,2)	POW(2,10)
100	1024

Funkcje daty i czasu

CURDATE(), CURTIME()

Wynikiem działania funkcji CURDATE() jest bieżąca data, a funkcji CURTIME() — bieżący czas. Częstym zastosowaniem funkcji jest automatyczne wstawianie informacji o czasie utworzenia lub zmodyfikowania danych

```
SELECT CURDATE(),CURTIME();
```



NOW()

Funkcja NOW() zwraca zarówno datę, jak i czas systemowy, NP. SELECT NOW();

DAYOFMONTH(), DAYOFWEEK(), DAYOFYEAR()

Funkcje DAYOFMONTH(), DAYOFWEEK(), DAYOFYEAR() zwracają numer dnia odpowiednio: miesiąca (liczba z zakresu od 1 do 31), tygodnia (liczba z zakresu od 1 do 7, gdzie 1 oznacza niedzielę — pierwszy dzień tygodnia, a 7 sobotę — siódmy dzień tygodnia) i roku (liczba z zakresu od 1 do 365)

Np. `SELECT * FROM zamowienie WHERE DAYOFWEEK(Data_zamowienia) = 2;`

HOUR(), MINUTE(), SECOND()

Funkcje HOUR(), MINUTE(), SECOND() zwracają odpowiednio godziny, minuty i sekundy z czasu przekazanego jako parametr wywołania

SELECT HOUR(CURTIME()), MINUTE(CURTIME()), SECOND(CURTIME());

Pokazano wiersze 0 - 0 (1 total, Wykonanie zapytania trwało 0,0002 sekund

```
SELECT HOUR(CURTIME()), MINUTE(CURTIME()), SECOND(CURTIME());
```

Profilowanie [Edytuj w linii] [Edit] [Explain SQL] [Create PHP code] [Re

Show all | Liczba wierszy: 25 Filter rows: Przeszukaj tę t

Extra options

HOUR(CURTIME())	MINUTE(CURTIME())	SECOND(CURTIME())
10	14	16

DAYNAME(), MONTHNAME()

Funkcje DAYNAME(), MONTHNAME() zwracają nazwę dnia tygodnia i miesiąca daty będącej argumentem wywołania

Pokazano wiersze 0 - 0 (1 total, Wykonanie zapytania trwało 0,

```
SELECT DAYNAME(CURDATE()), MONTHNAME(CURDATE());
```

Profilowanie [Edytuj w linii] [Edit] [Explain SQL] [Create PHP

Show all | Liczba wierszy: 25 Filter rows: P

Extra options

DAYNAME(CURDATE())	MONTHNAME(CURDATE())
Monday	March

DATE_ADD()

```
SELECT CURDATE(), DATE_ADD(CURDATE(),INTERVAL 3 DAY);
```

Działanie funkcji DATE_ADD() powoduje obliczenie wyniku zmiany daty (podanej jako pierwszy parametr) o ilość jednostek czasu określona przez drugi parametr

```
SELECT CURDATE(), DATE_ADD(CURDATE(),INTERVAL 3 DAY);
```

Profilowanie [Edytuj w linii] [Edit] [Explain SQL] [Create PHP coc

Show all | Liczba wierszy: 25 Filter rows: Przesz

Extra options

CURDATE()	DATE_ADD(CURDATE(),INTERVAL 3 DAY)
2024-03-18	2024-03-21

DATEDIFF()

Wynikiem działania funkcji DATEDIFF() jest liczba dni, które dzielą daty podane jako parametry funkcji.

SELECT id_zamowienia, DATEDIFF(data_zamowienia, date_dostarczenia) AS czas_realizacji FROM zamowienia

Funkcje kryptologiczne

Kryptologia to dziedzina nauki zajmująca się zabezpieczeniem informacji. Obejmuje ona kryptografię — jej dziedziną jest szyfrowanie danych — i opisującą techniki łamania szyfrów kryptoanalizę.

Kodowanie a szyfrowanie

Kodowanie i szyfrowanie nie są tym samym. Kodowanie polega na zapisywaniu wyrażeń jednego języka za pomocą wyrażeń innego języka. W informatyce ten drugi język sprowadza się najczęściej do zerojedynkowych ciągów bitów, stanowiących wewnętrzny język komputera (rysunek 5.1).

Rysunek 5.1. Proces kodowania danych

Kodowanie ma na celu takie przetworzenie pierwotnych danych, aby można było optymalnie je przechowywać lub przesyłać, a nie uniemożliwienie ich odczytania przez niepowołane osoby. Aby odczytać zakodowane dane, wystarczy znać zasadę działania dekodera (algorytm dekodowania).

Natomiast szyfrowanie jest odmianą kodowania, w której wymagamy, aby kod nie ujawniał zawartych w szyfrogramie informacji. W tym celu koder, nazywany szyfratorem lub enkryptorem, do utworzenia szyfrogramu (nazywanego też kryptogramem) wykorzystuje dodatkową informację — klucz. Odszyfrowanie danych wymaga użycia klucza, sama znajomość algorytmu dekodowania do tego nie wystarcza. Klucz powinny znać tylko osoby lub programy, dla których szyfrowane dane są przeznaczone; bezpieczeństwo szyfrogramu zależy od jakości algorytmu szyfrującego i stopnia skomplikowania(głównie długości) klucza. Ogólny schemat szyfrowania przedstawia rysunek 5.2.

Rysunek 5.2. Proces szyfrowania danych

Współcześnie powszechnie stosowane są dwie funkcje skrótu:

1. MD5 (ang. Message Digest Algorithm 5) — funkcja skrótu zdefiniowana w dokumencie RFC 1231, szybka, ale nieuważana już za bezpieczną, zwraca wyniki o długości 128 bitów.

2. SHA-1 (ang. Secure Hash Algorithm) — funkcja mieszania wolniejsza, ale bezpieczniejsza od poprzedniej, tworząca wyniki o długości 160 bitów. Funkcja ta jest standardem stosowanym przez agencje rządowe Stanów Zjednoczonych Ameryki Północnej.

ENCODE(), DECODE()

Funkcja ENCODE() zwraca szyfrogram podanego ciągu znaków. Do szyfrowania używane jest hasło podane jako drugi parametr wywołania (listing 5.37). Funkcja DECODE() deszyfruje zaszyfrowane funkcją ENCODE() szyfrogramy.

`DECODE(zakodowane_hasło, tekst)`
: Rozkodowuje `zakodowane_hasło` używając argumentu `tekst`. `zakodowane_hasło` jest łańcuchem zwróconym przez funkcje `ENCODE()`.

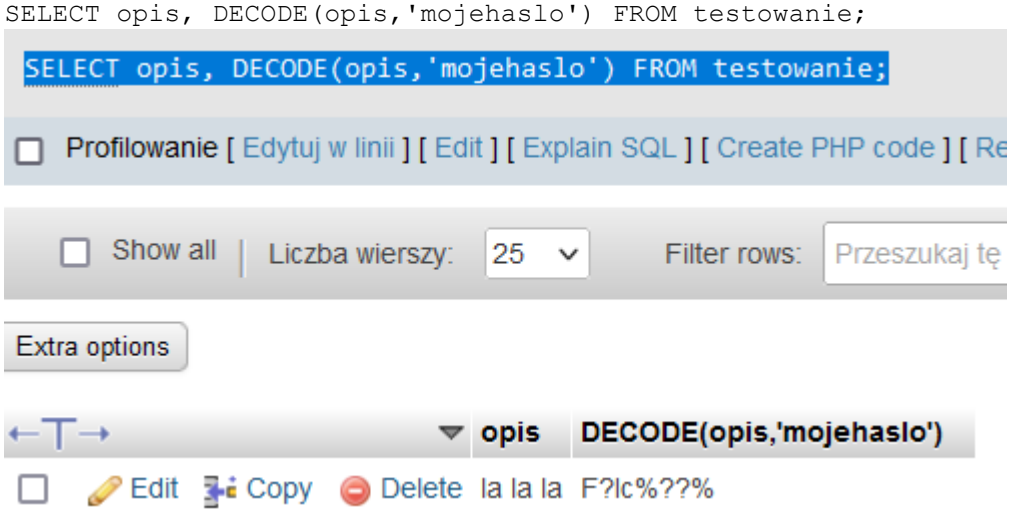
`ENCODE(hasło, tekst)`
: Koduje `hasło` używając argumentu `tekst`. Tak zakodowane hasło można odtworzyć używając funkcji `DECODE()`. Wynikiem jest łańcuch o długości argumentu `hasło`.

```
SELECT DECODE(ENCODE('moj tekst do ukrycia','haslo'),'haslo');

Utwórz tabelę testowanie, id i opis varchar 50. Wstaw do tabeli po 2 rekordy.

SELECT opis, DECODE(opis,'mojehaslo') FROM testowanie;

SELECT opis, DECODE(opis,'mojehaslo') FROM testowanie;
```



Procedura jest podprogramem utworzonym w SQL, która zawiera kod SQL oraz kod sterujący jej wykonaniem.

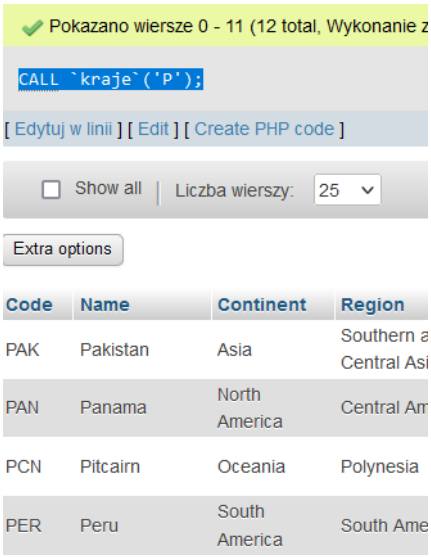
Procedura to kod SQL, który można zapisać i użyć ponowienie w przyszłości. Jeśli często wykonuje się jakąś kwerendę można przechowywać ją w takiej postaci. Do procedur również możemy przekazywać parametry, które są podobne do parametrów funkcji w innych językach programowania.

Przykłady procedur:

Na bazie world utwórz procedurę parametryczną, która wyświetli wszystkie dane z tabeli kraje, których nazwa kraju zaczyna się na literę podaną przez parametr. Nazwij ją kraje.


```
CREATE PROCEDURE `kraje`(IN `litera` CHAR(1)) NOT DETERMINISTIC CONTAINS SQL SQL
SECURITY DEFINER SELECT * FROM country WHERE name LIKE CONCAT(litera,"%");
```

```
CALL `kraje`('P');
```



Przykład 2.

Utwórz procedurę parametryczną wyświetlającą ile jest krajów na danym kontynencie. Parametrem niech będzie kontynent. Procedurę nazwij ilość.

```
SELECT `Continent`,count(`Code`) AS liczba FROM `country` GROUP by Continent;
```

```
ENUM('Asia','Europe','North America','Africa','Oceania','Antarctica','South America')
```


```
CREATE PROCEDURE `ILOSC`(IN `KONTYNENT` ENUM('Asia','Europe','North
America','Africa','Oceania','Antarctica','South America')) NOT DETERMINISTIC CONTAINS
SQL SQL SECURITY DEFINER SELECT `Continent`,count(`Code`) AS liczba FROM `country`
WHERE Continent=KONTYNENT GROUP by Continent;

CALL ILOSC('ASIA');
```