Procedury i funkcje w MySQL.

Tworzenie kopii zapasowej w MySQL.

Transakcje - ćwiczenie

Oprócz tabel i widoków w bazach danych możemy tworzyć własne funkcje, procedury i specjalny typ procedur, które będą automatycznie wywoływanie przez MySQL w momencie wstawiania, usuwania lub modyfikowania określonych danych, czyli wyzwalacze.

## Funkcje użytkownika

Jedną z możliwości MySQL-a jest możliwość tworzenia przez użytkowników własnych funkcji. Funkcjom, tak jak procedurom, można przekazać pewną liczbę parametrów, ale funkcja nie tylko wykonuje pewne operacje, ale także zwraca obliczony na podstawie przekazanych parametrów wynik.

Funkcje różnią się od procedur tym, że nie muszą być wskazywane parametry wejścia oraz wyjścia (funkcje bezparametrowe). Ponadto funkcja zwraca jakiś typ danych z dostępnych w SQL.

Ogólna postać:

CREATE FUNCTION nazwa\_f(parametry) RETURNS typ

BEGIN

 -- polecenia

 RETURN wynik

END

**Tworzenie funkcji**

Wykonanie instrukcji CREATE FUNCTION spowoduje utworzenie funkcji użytkownika. Funkcja z przykładu oczekuje na jeden parametr — datę — i zwraca odczytany z tej daty rok.

CREATE FUNCTION fn\_data

(data DATE)

RETURNS CHAR(20)

RETURN YEAR(data);

Osoby znające inne proceduralne języki programowania prawdopodobnie rozpoznają pewne podobieństwa w tworzeniu funkcji — w każdym przypadku trzeba zdefiniować nagłówek (nazwę, listę argumentów i typ zwracanej wartości) i ciało funkcji. Ostatnią instrukcją ciała funkcji powinna być instrukcja RETURN. Funkcja zwróci wartości wymienione po prawej stronie tej instrukcji.

Również można wykonać funkcję w formie graficznej:



Zastosowanie:

[SELECT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/select.html) `data\_start\_rezerwacji`,fn\_data(data\_koniec\_rezerwacji)FROM `rezerwacje`;



Przykład 2:

CREATE FUNCTION fn\_daty

 (data date, delimeter char(1))

RETURNS char(20)

RETURN CONCAT(YEAR(data),delimeter, MONTHNAME(data));

*Podczas korzystania z funkcji mogą wystąpić pewne problemy.
Jednym z ważniejszych jest znak średnika ';', który jest wpisywany podczas tworzenia funkcji/procedury. Jest prosty sposób na obejście tego. Wystarczy, że wpiszemy np. "DELIMITER //", co spokojnie pozwoli nam na zapisanie kodu. Natomiast znak średnika zastąpi '//'. Tak więc warto nie zapominać o tym, a na końcu warto wpisać "DELIMITER ;" tak aby średnik miał swoje dawne "znaczenie".*



Wywołanie na przykładzie bazy hotel:

[SELECT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/select.html) data\_start\_rezerwacji,fn\_daty(data\_start\_rezerwacji,': ') FROM rezerwacje



**Usuwanie funkcji**

Wykonanie instrukcji DROP FUNCTION spowoduje usunięcie wybranych funkcji użytkownika

DROP FUNCTION fn\_data;

**Procedura**

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

Procedury są przydatne wtedy, gdy chcemy wykonać jakieś często powtarzające się operacje.

Postać procedury:

CREATE PROCEDURE nazwa\_procedury (IN zmienna typ, OUT zmienna typ)

BEGIN

 -- polecenia i kod SQL

END

Przykład użycia procedury, obniżenie ceny o 30%

[CREATE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/create-procedure.html) [PROCEDURE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/create-procedure.html) `obnizka30`([IN](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/comparison-operators.html#function_in) `nazwa` [VARCHAR](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/string-types.html)(50)) [NOT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/logical-operators.html#operator_not) DETERMINISTIC NO SQL SQL SECURITY DEFINER [UPDATE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/update.html) produkt [SET](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/set.html) cena = cena \* 0.7 WHERE nazwa = nazwa



Wywołanie procedury:

call obnizka30('mikser');

**Procedura dodawania klienta: na bazie sklepik\_abc:**



[CREATE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/create-procedure.html) [PROCEDURE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/create-procedure.html) `dodaj\_klienta`([IN](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/comparison-operators.html#function_in) `klient\_id` [INT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/numeric-types.html), [IN](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/comparison-operators.html#function_in) `imie` [VARCHAR](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/string-types.html)(50), [IN](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/comparison-operators.html#function_in) `nazwisko` [VARCHAR](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/string-types.html)(50), [IN](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/comparison-operators.html#function_in) `miasto` [VARCHAR](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/string-types.html)(50), [IN](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/comparison-operators.html#function_in) `kraj` [VARCHAR](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/string-types.html)(50), [IN](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/comparison-operators.html#function_in) `telefon` [CHAR](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/string-types.html)(12)) [NOT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/logical-operators.html#operator_not) DETERMINISTIC NO SQL SQL SECURITY DEFINER [INSERT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/insert.html) INTO klienci(klient\_id, imie, nazwisko, miasto, kraj, telefon) [VALUES](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/miscellaneous-functions.html#function_values) (null,imie,nazwisko,miasto, kraj, telefon)

Wywołanie procedury dodaj\_klienta:

CALL dodaj\_klienta(null, 'Mariusz','Musiał', 'Łańcut', 'Polska', 123654123);

Dodany klient, przez procedurę:



### Usuwanie procedur

Nieużywane procedury możemy usunąć poprzez instrukcję **DROP PROCEDURE nazwa\_procedury;**

Często podczas wykonywania pewnych operacji na tabelach w bazie danych, takich jak wstawianie, usuwanie czy aktualizowanie rekordów, trzeba zaktualizować inną tabelę i/lub wykonać pewne obliczenia za każdym razem, gdy modyfikujemy dane. Można wtedy używać triggerów, czyli wyzwalaczy.

**Triggerem (wyzwalaczem)** nazywamy procedurę, która jest połączona z konkretną tabelą w bazie danych i wywołuje się automatycznie jako reakcja na wcześniej określone zdarzenie występujące w tej tabeli.

Do zdarzeń, które powodują wywołanie triggera, zalicza się:

* INSERT - dodanie nowego rekordu,
* UPDATE - zmodyfikowanie istniejącego rekordu,
* DELETE - usunięcie rekordu z bazy danych.

Trigger może być ustawiony do wywołania przed (BEFORE) lub po (AFTER) zdarzeniu, które go wywołuje.

* AFTER DELETE–wykonanie wyzwalacza po operacji usunięcia rekordu
* AFTER INSERT-wykonanie wyzwalacza po dodaniu rekordu
* AFTER UPDATE-wykonanie wyzwalacza po zmodyfikowaniu rekordu
* BEFORE DELETE-wykonanie wyzwalacza przed operacji usunięcia rekordu
* BEFORE INSERT-wykonanie wyzwalacza przed dodaniu rekordu
* BEFORE UPDATE-wykonanie wyzwalacza przed zmodyfikowaniu rekordu

**Przykład: na bazie sklepik\_abc**

Po wstawieniu jakiegoś rekordu do tabeli kategorie, zostanie automatycznie wstawiony rekord do producenta test\_wyzwalacza

[CREATE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/create-trigger.html) [TRIGGER](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/create-trigger.html) `wyzwalacz1` AFTER [INSERT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/insert.html) ON `kategorie` FOR EACH ROW [INSERT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/insert.html) INTO `producent` (`id\_producenta`, `nazwa\_producenta`) [VALUES](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/miscellaneous-functions.html#function_values) (NULL, 'test\_wyzwalacza');

**Dodawanie graficznie zdarzeń w xamppie**



Przykład: przez minutę co 15 sekund będzie się wstawiało do tabeli kategorie rekord test



[CREATE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/create-event.html) [EVENT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/create-event.html) `dodaj\_kategorie` ON SCHEDULE EVERY 15 SECOND STARTS '2021-01-03 13:55:00.000000' ENDS '2021-01-03 13:56:00.000000' ON COMPLETION [NOT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/logical-operators.html#operator_not) PRESERVE ENABLED [DO](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/do.html) [INSERT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/insert.html) INTO kategorie [VALUES](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/miscellaneous-functions.html#function_values) (null,'test');

Należy pamiętać o przestawieniu statusu na WŁĄCZONY –ENABLED.



Rysunek 1. Kategorie po wywołaniu zdarzenia

Pokazane wszystkie kategorie:



Rysunek 2. Kategorie przed wykonaniem zdarzenia

**Transakcje**- zbiór operacji na bazie danych stanowiących logiczną całość – albo żadną z nich.

Ćwiczenie przypominające o transakcjach. Można wykonać na dowolnej bazie. Przetestować, zrozumieć zasadę działania, a następnie usnąć tą tabele konta.

Tworzenie tabeli:

CREATE TABLE konta

(id INTEGER PRIMARY KEY,

id\_osoby INTEGER NOT NULL,

saldo DECIMAL(12,2) NOT NULL);

Wstawianie danych:

INSERT INTO konta VALUES(1,1,200.00);

INSERT INTO konta VALUES(2,2,300.00);

UPDATE `konta` SET saldo=saldo-40 WHERE `id`=1;

UPDATE `konta` SET saldo=saldo+40 WHERE `id`=2;

START TRANSACTION;

UPDATE `konta` SET saldo=saldo-40 WHERE `id`=1;

UPDATE `konta` SET saldo=saldo+40 WHERE `id`=2;

COMMIT;

SELECT \* FROM `konta`;

ODWOLANIE TRANSAKCJI:

START TRANSACTION;

UPDATE `konta` SET saldo=saldo-40 WHERE `id`=1;

UPDATE `konta` SET saldo=saldo+40 WHERE `id`=2;

ROLLBACK;

## **Tworzenie kopii zapasowej bazy danych MySQL**

Bazy danych zazwyczaj są tworzone aby przechowywać i przetwarzać jakieś ważne dane. Dlatego bardzo ważnym jest tworzenie kopii zapasowych pozwalające na zabezpieczenie informacji na wypadek zniszczenia danych lub awarii serwera.

Polecenie mysqldump służy do tworzenia plików tekstowych „dumps” baz danych zarządzanych MySQL. W tych plikach znajdują się wszystkie polecenia SQL, potrzebne do odtworzenia bazy danych od podstaw.

Otwieramy xampp-a, 🡪shell🡪otwieramy okienko wiersza poleceń. Chcemy utworzyć kopię bazy sklepik\_abc. Musimy napisać polecenie:

**mysqldump sklepik\_abc –u root –p > kopia\_sklepikabc.sql**

i wcisnąć ENTER. Dalej musimy podać hasło (u nas jest puste, więc ENTER i to wszystko).



Znajdź w katalogu głównym Xamppa plik kopia\_sklepikabc.sql



## **Przywracanie kopii zapasowej**

Otwieramy xampp-a, 🡪shell🡪otwieramy okienko wiersza poleceń. Chcemy odtworzyć kopię bazy kopia\_sklepikabc.sql. Musimy napisać polecenie:

**mysqldump sklepik\_abc –u root –p < kopia\_sklepikabc.sql**

i wcisnąć ENTER. Dalej musimy podać hasło (u nas jest puste, więc ENTER i to wszystko). Chwilę czekamy, aż zostaną wykonane wszystkie zapytania zapisane w pliku zrzutu.



Oczywiście tworzenie kopii i przywracanie bazy można wykonać graficznie z zakładki Import /Eksport.



**Zadanie do przećwiczenia.**

1. Wykonaj kopie zapasowe 2 swoich baz danych, a następnie dokonaj jakiś zmian w tych bazach, np. usuń rekordy.
2. Przywróć kopie zapasowe tych baz z punktu wyżej.