MODUŁ 5

Wybrane elementy języka PHP

PHP to, podobnie jak JavaScript, język skryptowy. Jednak w odróżnieniu od JavaScript funkcje napisane w PHP są interpretowane i wykonywane na serwerze. Jeśli plik, który ma zostać wysłany do przeglądarki internetowej w odpowiedzi na żądanie użytkownika zawiera kod PHP, najpierw na serwerze uruchamiany jest interpreter PHP, który wykonuje wszystkie instrukcje PHP, a dopiero potem tak przetworzony plik, nie zawierający już kodu w języku PHP, jest przesyłany do przeglądarki, która wyświetla jego zawartość.
Jeśli na serwerze występuje konieczność komunikacji z bazą danych, odbywa się to także przy pomocy instrukcji napisanych w PHP. Jeśli w kodzie HTML używane są instrukcje języka PHP, plik musi mieć rozszerzenie php. Plik zawierający obraz strony należy, podobnie jak pliki HTML, umieścić w katalogu *public_html* (lub jego podkatalogach).

Lokalizacja programów napisanych w PHP

Ponieważ programy w PHP są wykonywane na serwerze, muszą być tam zapisane. Przykłady zamieszczone w tej lekcji należy więc zapisywać w plikach (lub pliku) umieszczonych w katalogu *public_html* na serwerze ie.uek.krakow.pl i muszą mieć one rozszerzenie php.
Aby zobaczyć efekt w przeglądarce, należy wpisać w niej adres:
ie.uek.krakow.pl/~*login*/nazwa_pliku.php
w miejsce *login* wpisując swój identyfikator.

Przykład:

1. Uruchom program putty i połącz się ze swoim kontem na serwerze ie
2. Wejdź do katalogu *public_html* wydając polecenie:
   cd public_html
3. Uruchom dowolny edytor tekstowy, a tworzony plik nazwij *abc.php* (poniższa instrukcja uruchomi edytor *pico*, ale możesz skorzystać z dowolnego innego)
   pico abc.php
4. W utworzonym pliku umieść dowolny tekst i zapisz plik na dysku (w edytorze pico należy w tym celu przycisnąć *Ctrl+O*, *Enter*)
5. Uruchom przeglądarkę internetową i wpisz w niej adres:
   ie.uek.krakow.pl/~*login*/abc.php
   w miejsce *login* wpisując swój identyfikator na serwerze ie.
   W przeglądarce pojawiła się zawartość pliku. Parser PHP na serwerze wprawdzie się uruchomił, ale ponieważ w utworzonym pliku nie ma instrukcji PHP, więc nie wykonał żadnej operacji

Instrukcje PHP a kod HTML

W pliku zawierającym instrukcje PHP mogą także wystąpić m.in. znaczniki HTML. Powstaje pytanie, w jaki sposób interpreter PHP rozpoznaje kod PHP, a pomija pozostałe składniki kodu? Otóż, kod PHP jest specjalnie oznaczony − znajduje się między znakami `<?` oraz `?>`. Interpreter PHP przetwarza tylko instrukcje umieszczone wewnątrz nich, pozostałe pomija. Wszystko, co znajduje się między `<?` oraz `?>` musi być poprawnym składniowo kodem PHP, to co jest poza tymi znacznikami, ma zwykle format HTML-a.

Przykład (jeśli umieścisz ten kod w pliku a następnie wyświetlisz zawartość tego pliku w przeglądarce, prawdopodobnie pojawi się informacja o błędzie. Między znakami `<?` oraz `?>` nie znajdują się poprawne instrukcje w PHP i to właśnie parser PHP informuje o błędzie):

```
<body>
Tu jest kod w HTML, pomijany przez interpreter PHP
```

```
<?
Tu jest kod w PHP wykonywany przez interpreter PHP
?>
Tu ponownie jest kod w HTML
<?
Tu ponownie jest kod w PHP
?>
</body>
```

Wskaż wszystkie poprawne odpowiedzi dotyczące języka PHP:

- ☐ w pliku zawierającym kod PHP część HTML musi być umieszczona na początku
- ☐ plik zawierający opis strony internetowej, w którym jest także kod PHP musi być umieszczony w katalogu php
- ☑ **instrukcje PHP są wykonywane na serwerze**
- ☐ plik zawierający kod PHP musi mieć rozszerzenie htm, html lub php
- ☐ pliki zawierające opis strony internetowej z kodem PHP muszą być umieszczone w katalogu domowym lub jego podkatalogach
- ☑ **Kod PHP jest umieszczony między znakami <? oraz ?>**
- ☑ **kod zawierający opis strony internetowej przesłany z serwera do przeglądarki nie zawiera instrukcji PHP**
- ☐ **aby program w PHP działał, na serwerze musi być zainstalowany parser PHP**

Wybrane zasady pisania programów w PHP

W poniższej tabeli zamieszczono wybrane zasady pisania programów w języku PHP. Zapoznaj się z nimi

Nazwa zasady	Składnia / przykład
Każda instrukcja musi kończyć się średnikiem	

Komentarze wielowierszowe tworzy się przy pomocy znaków /* oraz */. Komentarzem jest także tekst rozpoczynający się od dwóch znaków *slash* do końca wiersza	`/* To jest komentarz składający się z wielu wierszy */`
	
	`print "Tekst";  // to jest komentarz na końcu wiersza`

Zmienne nie muszą być deklarowane. O tym, że ciąg znaków jest zmienną decyduje znak dolara ($) umieszczony przed tym ciągiem. Co więcej, można dowolnie zmieniać typy zmiennych. Aby przypisać zmiennej wartość należy użyć znaku równości	`$abc="Dowolne znaki";`
	// Zmienna $abc jest tekstem
	
	`$abc=34.8;`
	/* Ta sama zmienna `$abc` jest liczbą. Tym samym zmieniona została zarówno jej wartość, jak i typ. */

Wielkość liter w nazwach zmiennych jest istotna	`$abc` oraz `$Abc` oznaczają **różne** zmienne
Operatory relacyjne	`==` równy `!=` różny `>  >=` większy, większy lub równy `<  <=` mniejszy, mniejszy lub równy
Operatory logiczne	`and` iloczyn logiczny `or` suma logiczna `!` zaprzeczenie (not)
Operator przypisania	=

W poniższym tekście w miejsca oznaczone od (1) do (9) wstaw odpowiednie określenia lub znaki spośród następujących (tekst dotyczy języka PHP):

a) :
b) ;
c) /* oraz */
d) /^ oraz ^/
e) /
f) //
g) %
h) &
i) $
j) <>
k) !=
m) !
n) =
o) ==
p) :=
q) or
r) |
s) not

Jako odpowiedź na pytanie wpisz ciąg znaków (bez spacji, wielkość liter jest nieistotna) oznaczających kolejno wstawiane wyrazy. Przykładowa odpowiedź: cbedqisra
oznacza:1c, 2b, 3e, 4d, itd.

W PHP każda instrukcja musi kończyć się znakiem (1). Komentarz wielowierszowy tworzy się przy pomocy znaków (2). Komentarz na końcu wiersza zaczyna się od (3). Każda zmienna zaczyna się od (4). Operator "różny" to (5), a "równy" to (6). Suma logiczna to operator (7) a zaprzeczenie to (8). Operator przypisania to (9).

ODP: BCFIKOQMN

PHP - instrukcja warunkowa i pętle

Tak jak w każdym języku programowania, również w PHP jest instrukcja warunkowa i pętle. W poniższej tabeli znajdują się informacje na temat składni tych instrukcji w PHP. Zapoznaj się z tymi informacjami

Nazwa instrukcji	Składnia / przykład

Składnia instrukcji warunkowej `if`	`if (warunek) {`
	`ciąg instrukcji }`
	`[else {`
	`ciąg instrukcji }];`
	Przykład:
	`if ($x==23)`
	`print "X jest równe 23";`
	`else`
	`print("X jest różne od 23 i wynosi $x");`
	Przykład:
	`if (!($x==23))`
	`if ($x!=23)`
	W obydwu przypadkach warunek będzie prawdziwy, gdy w zmiennej `$x` będzie liczba różna od 23. W pierwszym należało zagnieździć nawiasy: zarówno warunek instrukcji `if` jak i negowane wyrażenie muszą być w nawiasie

Składnia pętli `for`	`for (zmiennaSterująca=wartośćPoczątkowa;`
	`warunekZakończenia; zmianaLicznika) {`
	`ciąg instrukcji;`
	`};`
	Przykład:
	`for ($i=0; $i<5; $i++)`
	`{`
	`...`
	`};`
	Pętla `for` jest kończona, gdy `warunekZakończenia` przyjmie wartość fałszu (pętla z powyższego przykładu wykona się pięciokrotnie, dla `$i=0,1,2,3,4`)

Składnia pętli `while`	`while (warunek) {`
	`...`
	`};`
	Pętla **while** jest kończona, gdy `warunek` przyjmie wartość fałszu

[redacted]

Wyrażenie:
if ($x!=23)

[redacted]

- jest błędne, gdyż zmienna x jest poprzedzona błędnym znakiem
- będzie prawdziwe, gdy w zmiennej $x będzie liczba równa 23
- **<u>będzie prawdziwe, gdy w zmiennej $x będzie liczba różna od 23</u>**
- jest błędne, gdyż w PHP nie istnieje operator !=

PHP - przykłady

W dalszej części lekcji zamieszczono kilka przykładów dotyczących języka PHP. Możesz je wpisywać w pliku *abc.php* utworzonym na serwerze w katalogu *public_html* i obserwuj efekt w przeglądarce.

Wyświetlenie tekstu na stronie internetowej

Aby z poziomu PHP umieścić tekst na stronie internetowej, należy użyć funkcji *print* lub *echo*, a parametr, czyli tekst, który ma być wyświetlony, należy umieścić w cudzysłowie lub ograniczyć apostrofami.

Przykład:
```
1 <body>
2 Jan Kowalski<br>
3 <?
4 print "Ewelina Malinowska, ";
5 echo 'Wojciech Malinowski';
6 ?>
7 </body>
```

W przeglądarce internetowej pojawi się tekst:
Jan Kowalski
Ewelina Malinowska, Wojciech Malinowski

Napis **Jan Kowalski** zostanie wyświetlony w wyniku interpretacji wiersza 2, który znajduje się w sekcji HTML, zostanie więc pominięty przez parser PHP. Dwa kolejne imiona i nazwiska, umieszczone poniżej, są wynikiem wykonania funkcji *print* i *echo* umieszczonych w kodzie PHP (wiersze 4 i 5).Zauważ, że parametr funkcji *print* został umieszczony w cudzysłowie, a funkcji *echo* w apostrofach. W tym przypadku nie ma to znaczenia.

Znaczniki HTML w kodzie PHP

W kodzie PHP można używać znaczników HTML. Muszą one być wtedy parametrami funkcji *print* lub *echo*. Jeśli łańcuch znakowy jest ograniczony cudzysłowami, wewnątrz nich można używać apostrofów. Jeśli konieczne jest wtedy użycie cudzysłowów, muszą one być poprzedzone znakiem backslash (\).

Przykład:
```
1 <?
2 print "Wpisz nazwę:<br>";
3 echo "<input type=\"text\" name='nazwa' size='30' maxlength='50'>";
4 ?>
```

Przy pomocy funkcji print w przeglądarce wyświetlony zostanie tekst, a występujący po nim znacznik
 spowoduje przejście do nowego wiersza (wiersz 2). Przy pomocy funkcji *echo* w przeglądarce wyświetlone zostanie pole tekstowe formularza (wiersz 3).
Tekst będący parametrem funkcji *echo* został ujęty w cudzysłów (wiersz 3). Wewnątrz można więc bez przeszkód używać apostrofów ale także cudzysłowów, poprzedzając je znakiem \. Powyższe dwie instrukcje będą także poprawne, jeśli wszystkie apostrofy zostaną zamienione na cudzysłowy, a cudzysłowy na apostrofy.

Wyświetlenie w przeglądarce zawartości zmiennej

Ciąg znaków będący parametrem funkcji print i zawierający nazwę zmiennej, której zawartość ma zostać wyświetlona, musi być ujęty w cudzysłów. Łańcuch znaków ograniczony apostrofami jest traktowany dosłownie, a wartość zmiennej nie jest wtedy podstawiana pod jej nazwę.

Przykład:
```
1 <?
2 $x=20;
3 print "Wartość zmiennej jest równa $x<br>";
4 echo 'Wartość zmiennej jest równa $x<br>';
5 ?>
6 Dowolny tekst w nowym wierszu
```

W wyniku wykonania programu w przeglądarce pojawi się:

```
Wartość zmiennej jest równa 20
Wartość zmiennej jest równa $x
Dowolny tekst w nowym wierszu
```
Parametr funkcji *print* został umieszczony w cudzysłowie, a ponieważ wewnątrz niego jest zmienna $x, więc w przeglądarce wyświetlona zostanie jej wartość, czyli liczba 20. Parametr funkcji *echo* został natomiast umieszczony w apostrofach, więc jest traktowany dosłownie, łącznie z napisem $x, który w tym przypadku nie jest interpretowany jako zmienna. W obydwu przypadkach znacznik
 spowoduje przejście do nowej linii, nie ma znaczenia, czy jest ograniczony apostrofami, czy umieszczony w cudzysłowie.

Łańcuchy znakowe łączy się przy pomocy kropki

Aby połączyć łańcuchy znakowe należy użyć kropki.

Przykład:
```
1 <?
2 $x = "Dzisiaj jest ";
3 $y = "ładna pogoda";
4 $z = $x.$y;
5 print $z."<br>a może nie";
6 ?>
```

W przeglądarce pojawi się tekst:
```
Dzisiaj jest ładna pogoda
a może nie
```
Zmiennej *$z* zostaje przypisane połączenie tekstów znajdujących się w zmiennych *$x* i *$y* (wiersze 2-4). Parametrem instrukcji *print* są *$z* wraz z dołączonym do niej tekstem umieszczonym w cudzysłowie, na początku którego znajduje się znacznik `<br>` oznaczający przejście do nowego wiersza (wiersz 5). Jeśli parametrem instrukcji *print* jest zmienna, nie musi być wpisana w cudzysłowie.
Jeśli w instrukcji w wierszu 4 w miejsce kropki wpisany zostałby znak plus, oznaczałoby to dodawanie liczb znajdujących się w zmiennych *$x* i *$y*, a ponieważ nie przechowują one liczb, zmiennej *$z* przypisana zostałaby liczba 0 i ona pojawiłaby się na ekranie. Efekt w przeglądarce byłby więc następujący (sprawdź):
```
0
a może nie
```

Wskaż, co będzie wynikiem wykonania poniższych instrukcji:
```
$x=20;
print '$x';
```

- wystąpi błąd, gdyż niepoprawna jest druga z instrukcji
- wystąpi błąd, gdyż niepoprawna jest pierwsza z instrukcji
- **na ekranie pojawi się napis: $x**
- na ekranie pojawi się liczba 20

Tworzenie i wykorzystanie funkcji w języku PHP

Funkcje w PHP

Tak, jak w każdym języku programowania, również w PHP można tworzyć i wykorzystywać funkcje. W PHP funkcje posiadają następująca składnię:

```
function nazwa (lista parametrów formalnych
{
instrukcje wchodzące w skład funkcji;
return wyrażenie;
}
```

Wyrażenie zwracane przez funkcję jest podawane jako parametr instrukcji *return*. W funkcji może występować wiele instrukcji *return*, funkcja jest kończona po napotkaniu pierwszej z nich. Jeśli funkcja nie zwraca wartości, instrukcja *return* nie musi być użyta.
W celu wykorzystania utworzonej funkcji należy wpisać jej nazwę oraz, w nawiasie, listę parametrów aktualnych, z którymi funkcja ma zostać uruchomiona.

Przykład użycia funkcji w PHP

Poniższy program możesz zapisać w pliku (plik musi mieć rozszerzenie php) umieszczonym w katalogu domowym lub w katalogu *public_html*. Plik nie musi być uruchamiany za pośrednictwem przeglądarki. Wystarczy w systemie Linux będąc w katalogu, w którym znajduje się plik, wydać polecenie:
php *nazwa_pliku*.php
Uruchomiony zostanie parser PHP i efekt działania programu pojawi się na ekranie. Aby plik uruchomić poprzez przeglądarkę, musisz umieścić go w katalogu *public_html*.

W katalogu domowym lub w public_html utwórz plik o nazwie *f.php* i wpisz w nim poniższy kod (pomiń numery wierszy):

```
1 <?
2 function pole_figury ($r, $figura)
3 /* funkcja liczy pole koła lub kwadratu */
4 {
5 if ($figura=='kolo')
6 $pole=pi()*$r*$r;
7 else
8 if ($figura=='kwadrat')
9 $pole=$r*$r;
10 else
11 $pole='brak lub błędny parametr funkcji';
12 return $pole;
13 }
14 $a=10;
15 print pole_figury($a, 'kolo');
16 ?>
```

Omówienie sposobu działania powyższego programu:

Cały program znajduje się między znacznikami PHP (<? oraz ?>). Najpierw zdefiniowana zostaje funkcja o nazwie *pole_figury* (wiersze 2-13). W nagłówku funkcji (wiersz 2) zdefiniowano dwa parametry formalne: *$r* oraz *$figura*. Podczas wykonania funkcji pierwszemu z nich przypisana zostanie liczba, drugiemu napis. Jeśli parametr *$figura* będzie miał wartość 'kolo', (wiersz 5), obliczone zostanie pole koła o promieniu podanym jako pierwszy parametr *$r* (wiersz 6). W przeciwnym przypadku, gdy zmienna *$figura* będzie miała wartość 'kwadrat' (wiersz 8), obliczone zostanie pole kwadratu o boku podanym jako pierwszy parametr *$r* (wiersz 9). W każdym przypadku wynik zostanie przypisany zmiennej *$pole*. Jeśli zmienna *$figura* będzie miała inną wartość (nie 'kolo' i nie 'kwadrat'), zmiennej *$pole* przypisany zostanie tekst (wiersz 11). Funkcja zwraca wartość zmiennej *$pole* (wiersz 12), a więc albo pole koła, albo kwadratu albo napis 'brak lub błędny parametr funkcji'.
W programie, który zaczyna się w wierszu 14, zmiennej *$a* przypisywana jest liczba 10, a następnie uruchamiana jest funkcja *pole_figury* z parametrami aktualnymi: zmienną *$a*, która jest podstawiana za parametr formalny *$r* oraz napisem 'kolo', który jest podstawiany za zmienną *$figura*. Funkcja jest więc uruchamiana z parametrami $r=10 oraz $figura='kolo'. Ponieważ nazwa funkcji w wierszu 15 jest parametrem instrukcji *print*, w wyniku działania programu na ekranie pojawia się

314.1592...
czyli pole koła o promieniu 10.
W wierszu 15 zmień napis 'kolo' na 'kwadrat' a następnie na inny tekst, za każdym razem uruchom program i obserwuj efekt jego działania.

Dołączanie kodu z pliku zewnętrznego

Funkcje zwykle pisze się w celu ich wielokrotnego wykorzystania. Jeśli funkcje lub inne fragmenty kodu, które mają być użyte w danym programie są zapisane w pliku zewnętrznym, należy dołączyć zawartość takiego pliku do tworzonego programu przy pomocy instrukcji **include**. Nazwę dołączanego pliku należy wpisać w cudzysłowie lub w apostrofach. Plik powinien być w tym samym katalogu, co tworzony program, w przeciwnym przypadku nazwę dołączanego pliku należy poprzedzić wskazaniem jego lokalizacji. Dla zachowania porządku, wszystkie pliki dołączane są zwykle na początku pisanego programu, choć dozwolone jest użycie instrukcji **include** w dowolnym miejscu. Należy jednak pamiętać, że elementy zapisane w pliku zewnętrznym są dostępne dopiero po jego dołączeniu.

Dołączanie kodu z pliku zewnętrznego - przykład 1

W katalogu *public_html* utwórz plik o nazwie *zmiennaX.php* i umieść w nim następujący kod:
```
<?
$x=25;
?>
```
W katalogu *public_html* utwórz plik o nazwie *x.php* i wpisz w nim następujący kod:
```
<?
print $x;
include "zmiennaX.php";
?>
```
Następnie w systemie Linux przy aktywnym katalogu public_html wpisz polecenie:
php x.php
Nie ma informacji o błędzie, gdyż program jest składniowo poprawny, ale na ekranie nic się nie wyświetliło. Najpierw wykonywana jest bowiem instrukcja **print $x**, a ponieważ zmienna *$x* jest nieznana, gdyż plik zawierający instrukcję przypisująca jej wartość jest dołączony później, więc taki jest efekt działania programu.
Jeśli jednak w programie *x.php* zamienisz miejscami instrukcje *print* i *include* (zrób to!), na ekranie pojawi się liczba 25. Tym razem zmienna *$x* jest już widoczna, gdyż instrukcja *include* dołączająca plik, w którym przypisywana jest wartość tej zmiennej została użyta wcześniej.

Zasięg instrukcji include

Instrukcja *include* ma zasięg lokalny. Musi być użyta wewnątrz funkcji która korzysta z dołączanego pliku.

Przykład:
Załóżmy, że w pliku *abc.php* jest zdefiniowana funkcja *fx()*. Poniższy sposób dołączenia pliku *abc.php* i wywołania funkcji *fx()* jest niepoprawny, gdyż zawartość pliku *abc.php*, a więc zdefiniowana w nim funkcja *fx()* jest niedostępna wewnątrz funkcji *xyz()*:
```
include "abc.php";
function xyz() {
fx();
...}
```

Natomiast poniższy sposób jest poprawny – plik zawierający funkcję *fx()* jest dołączany wewnątrz funkcji *xyz()*, w której wywoływana jest *fx()*:
```
function xyz() {
include 'abc.php';
fx();
...}
```

Jeśli kod PHP (np. funkcja) znajduje się w pliku zewnętrznym, to należy go dołączyć przy pomocy

instrukcji:

- insert "nazwa_pliku"
- include "nazwa_pliku"
- get "nazwa_pliku"
- join "nazwa_pliku"

PHP - nawiązanie połączenia z bazą danych i wykonanie zapytania

PHP jako język komunikacji z bazami danych

Aby na stronie internetowej mogły być wyświetlane dane pobrane z bazy danych, na serwerze o stałym adresie IP muszą być zainstalowane:

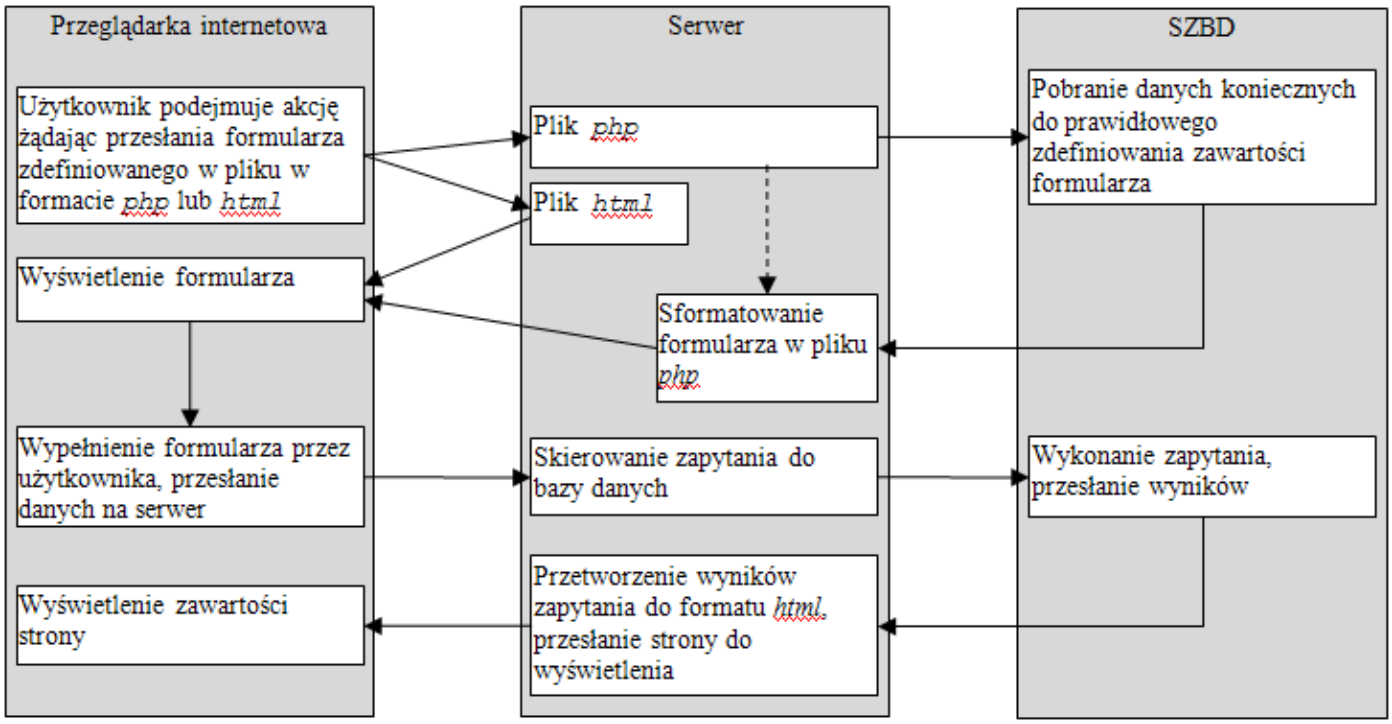
- SZBD (np. PostgreSQL), na którym musi być utworzona baza danych, z której dane będą pobierane, formatowane i wyświetlane w przeglądarce internetowej
- interpreter języka umożliwiającego komunikację z bazą danych (np. PHP)

PHP i PostgreSQL

PHP został wyposażony w funkcje umożliwiając komunikację z większością najpopularniejszych SZBD. Funkcje te dla wszystkich SZBD mają takie same nazwy oraz działają w ten sam sposób, zwykle wymagają także podania tych samych parametrów. Różnią się one między sobą jedynie prefiksem (zakończonym znakiem podkreślenia), który wskazuje na konkretny SZBD. Na zajęciach wykorzystywany będzie SZBD PostgreSQL, trzeba więc wiedzieć, że funkcje PHP służące do komunikacji z tym systemem mają prefix pg_

Schemat działania aplikacji internetowej z dostępem do bazy danych

Projekt przykładowej aplikacji, w której wykorzystano formularz do komunikacji użytkownika Internetu z bazą danych, w której w charakterze oprogramowania klienckiego wykorzystana została przeglądarka internetowa, został zobrazowany na poniższym rysunku.



W przeglądarce użytkownik wpisuje adres strony internetowej zawierającej formularz. Dane potrzebne do prawidłowego wyświetlenia formularza mogą pochodzić z bazy danych lub nie. Jeśli tak, w pliku muszą być instrukcje PHP i musi on mieć rozszerzenie php, jeśli nie, może mieć rozszerzenie htm lub html. Plik zawierający formularz zostaje przesłany z serwera (jeśli wymagane jest pobranie danych do formularza z bazy danych, wcześniej następuje komunikacja z bazą danych) i wyświetlony w oknie przeglądarki. Mogą być z nim związane inne pliki (np. CSS lub skrypty JavaScript) – one także są pobierane i wykorzystywane przez przeglądarki do wyświetlenia zawartości. Następnie użytkownik wypełnia formularz korzystając z dostępnych kontrolek. Po kliknięciu przycisku polecenia *Submit* dane z formularza są przesyłane na serwer. Tam uruchamiana jest funkcja napisana w języku PHP (nazwa pliku zawierającego tę funkcję musi zostać podana w nagłówku formularza) i dane przesłane z formularza są w niej dostępne. Procedura łączy się z bazą danych i kieruje do niej zapytanie w języku SQL: może to być zarówno kwerenda funkcjonalna jak i wybierająca. W bazie danych zapytanie jest wykonywane, dane są zwracane do procedury (jeśli zapytanie jest kwerendą wybierającą) a następnie formatowane i przesyłane do przeglądarki internetowej, gdzie są wyświetlane.

Uporządkuj czynności, które są wykonywane, gdy prawidłowe wyświetlenie danych w przeglądarce wymaga dostępu do bazy danych zarówno w celu wyświetlenia formularza jak i danych pobranych z bazy danych (jedna podana czynność jest zbędna):
a) pobranie danych koniecznych do zdefiniowania zawartości formularza
b) przetworzenie wyniku zapytania do formatu html
c) sformatowanie formularza
d) sformułowanie zapytania przez użytkownika
e) skierowanie zapytania do bazy danych
f) wykonanie zapytania
g) wypełnienie formularza przez użytkownika
h) wyświetlenie formularza
i) wyświetlenie zawartości strony internetowej zawierającej wynik zapytania

Jako odpowiedź na pytanie wpisz ciąg znaków (bez spacji, wielkość liter jest nieistotna) oznaczających kolejno wykonywane czynności. Przykładowa odpowiedź:
cbedghfa
oznacza: najpierw c, potem b, itd.

ODP: ?

Nawiązanie połączenia z bazą danych

Aby można było korzystać z danych przechowywanych w bazie danych, najpierw należy nawiązać połączenie z tą bazą danych. W PHP służy do tego funkcja *connect*, a jeśli SZBD jest PostgreSQL, funkcja nazywa się *pg_connect*.
Funkcja wymaga podania jednego parametru składającego się z czterech elementów:

```
$con=pg_connect("host=adres_serwera dbname=nazwa_bazy_danych
user=nazwa_użytkownika password=hasło");
```

gdzie:
`host` to nazwa serwera, na którym znajduje się baza danych. Jeśli jest ona umieszczona na tym samym serwerze, na którym znajduje się procedura, z której wywoływana jest ta funkcja, można ten parametr pominąć lub nadać mu wartość *localhost*,
`dbname` to nazwa bazy danych, z którą ma zostać nawiązane połączenie,
`user` to nazwa użytkownika w SZBD, który jest właścicielem bazy danych,
`password` to hasło dostępu do bazy danych.
Wynik działania funkcji *pg_connect* jest przypisywany do zmiennej (w przykładzie ma ona nazwę *$con*). Od momentu nawiązania połączenia wszystkie odwołania do bazy danych następują za pośrednictwem tej zmiennej.

Przechowywanie hasła dostępu do bazy danych w osobnym pliku

Hasło dostępu do SZBD (i do bazy danych) będące elementem parametru funkcji *pg_connect* podawane jest w sposób jawny. Nie jest ono jednak przesyłane przez sieć komputerową (instrukcje PHP są wykonywane na serwerze, przed przesłaniem pliku do przeglądarki), nie ma więc niebezpieczeństwa przechwycenia hasła w sieci przez osoby niepowołane. Aby dodatkowo zwiększyć bezpieczeństwo i uniknąć możliwości podejrzenia hasła w trakcie tworzenia kodu, można je ukryć w osobnym pliku postępując w następujący sposób:

- w tym samym katalogu, w którym znajduje się funkcja, w której jest instrukcja nawiązująca połączenie z bazą danych (a więc zwykle w *public_html*) należy utworzyć plik o dowolnej nazwie i rozszerzeniu php (np. *haslo.php*) i wpisać w nim kod:
  ```
  <?php
  $h='xyz';
  ?>
  ```
  w miejsce **xyz** wpisując hasło dostępu do bazy danych. W ten sposób w pliku zostanie utworzona zmienna o nazwie *$h* zawierająca hasło dostępu do SZBD,
- w pliku, w którym wymagana jest komunikacja z bazą danych, za pomocą instrukcji *include* dołączyć plik *haslo.php* powyżej instrukcji służącej do nawiązania połączenia z bazą danych, a jako parametr *password* funkcji *pg_connect* podać nazwę zmiennej *$h*. W efekcie połączenie z bazą danych zostanie nawiązane dzięki poniższym dwóm instrukcjom:
  ```
  include "haslo.php";
  $con=pg_connect("dbname=BD user=użytkownik password=$h");
  ```

Wykonywanie zapytań

Aby skierować zapytanie do bazy danych należy użyć funkcji *pg_exec*. Wymaga ona podania dwóch parametrów:

```
$result=pg_exec($con, $query);
```

gdzie:
`$con` to zmienna, której przypisano połączenie z bazą danych (wymagane jest wcześniejsze użycie funkcji *pg_connect* i przypisanie wyniku jej działania zmiennej, która w tym przypadku została nazwana *$con*),
`$query` to zmienna zawierająca zapytanie w SQL, które ma zostać skierowane do bazy danych. Zapytanie może być dowolnego typu: wybierające (Select) lub funkcjonalne (np. Create Table, Insert Into, Update, Delete From).

Jeśli kwerenda jest wybierająca (Select), dostęp do wybranych w ten sposób danych następuje za pośrednictwem zmiennej *$result*. Zmienna ta jest dwuwymiarową tablicą asocjacyjną. Tablica taka charakteryzuje się tym, że poszczególne jej elementy są indeksowane nie tylko kolejnymi liczbami całkowitymi (w PHP elementy tablicy są indeksowane od 0), jak w przypadku zwykłych zmiennych tablicowych, ale także łańcuchami znakowymi. Jeśli zmienna przechowuje wynik zapytania skierowanego do bazy danych, to indeksy kolumn są nazwami pól zwróconych przez zapytanie.

Wykonanie zapytań - przykład

Reasumując, aby skierować zapytanie do bazy danych należy wykonać trzy instrukcje:

```
1 $con=pg_connect("dbname=nazwa_BD user=użytkownik password=hasło");
2 $query="select * from pracownicy order by nazwisko";
3 $result=pg_exec($con, $query);
```

W pierwszej trzeba nawiązać połączenie z bazą danych (w wierszu 1 pominięto parametr host, oznacza to, że połączenie nawiązywane jest z bazą danych znajdującą się na tym samym serwerze, na którym umieszczona jest aplikacja, w której znajduje się instrukcja wykonująca tę operację). Wynik tego połączenia należy przypisać zmiennej (wierszu 1 jest to zmienna o nazwie *$con*). Następnie zmiennej należy przypisać treść zapytania, które zostanie skierowane do bazy danych (zmiennej *$query* przypisano zapytanie wybierające wszystkie dane z tabeli *pracownicy*, posortowane według pola *nazwisko* – wiersz 2). Na koniec, przy pomocy funkcji *pg_exec*, należy skierować zapytanie do bazy danych (zapytanie znajdujące się w zmiennej *$query* zostało przesłane do bazy danych związanej ze zmienną *$con* – wiersz 3). Wynik zapytania jest przypisywany zmiennej, w przykładzie nosi ona nazwę *$result* (wiersz 3).

1. Połącz się ze swoim kontem na serwerze ie.uek.krakow.pl i wejdź do katalogu *public_html*
2. Utwórz plik tekstowy o nazwie **dane.php**
3. W pliku *dane.php* wpisz powyższe trzy instrukcje. Pamiętaj, aby odpowiednio wskazać parametry w pierwszym wierszu. Pamiętaj też, aby na początku umieścić znaki `<?` a na końcu `?>`
4. Zapisz plik na dysku, uruchom przeglądarkę internetową i wpisz adres: ie.uek.krakow.pl/~*login*/dane.php
   w miejsce *login* wpisując swój identyfikator na serwerze ie.uek.krakow.pl
5. Jeśli w przeglądarce nic się nie pojawiło, oznacza to, że nawiązanie połączenia z bazą danych się powiodło i zapytanie zostało prawidłowo wykonane. W następnej lekcji dowiesz się, jak wyświetlić dane zwrócone przez zapytanie w oknie przeglądarki internetowej. Jeśli w przeglądarce pojawiła się informacja o błędzie, musisz ten błąd poprawić. Pamiętaj, aby program ten zadziałał poprawnie, w bazie danych musi być tabela o nazwie *pracownicy* zawierająca przynajmniej pole *nazwisko*, gdyż z tej tabeli wybierane są dane z bazy danych

Aby nawiązać połączenie z bazą danych oraz wykonać zapytanie należy użyć kolejno funkcji:

- pg_sql, pg_exec
- pg_connect, pg_sql
- **pg_connect, pg_exec**
- pg_exec, pg_connect

PHP - przetwarzanie wyników zapytań wybierających

Plik wejściowy i zawartość bazy danych

W lekcji tej zakładamy, że w katalogu *public_html* na serwerze ie.uek.krakow.pl masz plik o nazwie *dane.php*, który powstał w wyniku wykonania poprzedniej lekcji oraz że plik ten ma następująca zawartość:

```
<?
$con=pg_connect("dbname=nazwa_BD user=użytkownik password=hasło");
$query="select * from pracownicy order by nazwisko";
$result=pg_exec($con, $query);
?>
```

Zakładamy także, że w Twojej bazie danych znajduje się tabela *pracownicy*, a w niej następujące pola:
id_pracownika, *nazwisko*, *imie*, *nip*, *pesel*
Tabela ta jest częścią bazy danych dla wyższej uczelni, która była tworzona na jednych z poprzednich zajęć stacjonarnych.

Odczytanie liczby wierszy i kolumn zwróconych przez zapytanie

Jeśli zapytanie jest wybierające, zmienna zawierająca jego wynik jest dwuwymiarową tablicą (w pliku *dane.php* jest to zmienna *$result*). Aby odczytać, z ilu wierszy i kolumn składa się ta tablica, a więc ile wierszy i kolumn zostało zwróconych przez zapytanie, należy użyć funkcji PHP:
`pg_numrows($result);`
`pg_numfields($result);`
Funkcja *pg_numrows* zwraca liczbę wierszy a *pg_numfields* liczbę kolumn zwróconych przez zapytanie wybierające. W obydwu przypadkach parametrem jest nazwa zmiennej, której przypisano wynik działania funkcji *pg_exec*. Funkcje te są wykorzystywane do sprawdzenia, czy zapytanie zwróciło jakieś dane, może się bowiem zdarzyć tak, że zapytanie jest poprawne składniowo, ale w bazie danych nie ma odpowiednich danych i zwrócona tabela będzie pusta. Funkcje te mogą być także wykorzystane do przetworzenia danych zwróconych przez zapytanie, np. do wyświetlenia ich w przeglądarce.

W poniższym polu wpisz nazwy funkcji PHP, która zwraca liczbę wierszy zwróconych przez zapytanie skierowane do SZBD PostgreSQL, a po przecinku (bez spacji) nazwę funkcji, która zwraca liczbę kolumn zwróconych przez zapytanie.
UWAGA:
wielkość liter jest istotna

odpowiedź: `pg_numrows,pg_numfields`

Wyświetlenie wierszy i kolumn zwróconych przez zapytanie

1. W pliku *dane.php*, przed znakami zamykającymi kod PHP (przed ?>) umieść następujące instrukcje:
```
$liczba_wierszy=pg_numrows($result);
$liczba_kolumn=pg_numfields($result);

print "Zapytanie zwróciło ".$liczba_wierszy." wierszy oraz
".$liczba_kolumn." kolumn<br>";
```

2. W przeglądarce internetowej wyświetl zawartość pliku dane.php wpisując adres: ie.uek.krakow.pl/~*login*/dane.php
   w miejsce login wpisując swój identyfikator na serwerze ie

W przeglądarce pojawiła się informacja o liczbie wierszy i kolumn zwróconych przez zapytanie. W powyższym kodzie zmiennej *$liczba_wierszy* jest przypisywana liczba rekordów zwróconych przez zapytanie a zmiennej *$liczba_kolumn* liczba pól. Następnie instrukcja *print* wyświetla te dane z odpowiednim komentarzem.

Odczytywanie danych zwróconych przez zapytanie

Aby odczytać pojedynczą wartość z tabeli zawierającej wynik zapytania wybierającego należy użyć funkcji *pg_result*:
`pg_result($result, nr_wiersza, nr_kolumny)`
Pierwszym parametrem jest nazwa zmiennej zawierającej wynik zapytania, drugim numer wiersza a trzecim numer kolumny elementu, który chcemy odczytać. Należy pamiętać, że elementy tablicy *$result* są numerowane od zera.

Wyświetlenie pojedynczej wartości z tabeli wynikowej zapytania

Na końcu pliku *dane.php*, bezpośrednio przed znakami ?> wpisz:
`print pg_result($result,0,1)."<br>";`
a następnie przełącz się do przeglądarki i odśwież zawartość wyświetlanego w niej pliku *dane.php*.

Funkcja *pg_result* zwraca wartość elementu tablicy *$result* znajdującego się w pierwszym wierszu (o numerze 0) oraz drugiej kolumnie (o numerze 1) tabeli zwróconej przez zapytanie. Jest to nazwisko pierwszego pracownika. Ponieważ funkcja *pg_result* jest parametrem instrukcji *print*, więc nazwisko to pojawiło się w oknie przeglądarki.

Zmienna zawierająca wynik zapytania jako tablica asocjacyjna

Ponieważ zmienna zawierająca wynik zapytania (*$result*) jest tablicą asocjacyjną, aby wskazać kolumnę, zamiast numeru można użyć jej nazwy.

1. W pliku *dane.php*, przed znakami zamykającymi kod PHP (przed ?>) umieść instrukcję:
   `print pg_result($result,2,pesel)."<br>";`
2. W przeglądarce internetowej odśwież zawartość pliku *dane.php*

W przeglądarce pojawił się PESEL pracownika, którego dane znajdują się w trzecim rekordzie (o numerze 2) tabeli wynikowej zapytania. Korzystając z faktu, że *$result* jest tablicą asocjacyjną, jako trzeci parametr funkcji *pg_result* wpisano nazwę kolumny.

Wskaż wszystkie poprawne odpowiedzi dotyczące funkcji pg_result:

- ☐ wymaga podania trzech parametrów
- ☐ drugim jej parametrem jest numer kolumny
- ☐ zwraca liczbę rekordów zwróconych przez zapytanie
- ☐ zwraca jedną kolumnę z tabeli wynikowej zapytania
- ☐ pierwszym jej parametrem jest tablica dwuwymiarowa
- ☐ zwraca jeden wiersz tabeli wynikowej zapytania
- ☐ drugim jej parametrem jest numer wiersza
- ☐ zwraca jedną wartość
- ☐ wymaga podania dwóch parametrów
- ☐ pierwszym jej parametrem jest nazwa zmiennej zawierającej wynik zapytania

Odczytanie nazwy kolumny z tabeli wynikowej zapytania

Aby odczytać nazwę kolumny z tabeli wynikowej zapytania, należy użyć funkcji:
`pg_field_name($result, nr_kolumny)`
Jako drugi parametr funkcji należy wpisać numer kolumny (kolumny numerowane są od zera), której nazwa ma zostać zwrócona.

1. W pliku *dane.php*, przed znakami zamykającymi kod PHP (przed ?>) umieść instrukcję:

```
print pg_field_name($result,2)."<br>";
```
2. W przeglądarce internetowej odśwież zawartość pliku *dane.php*

W przeglądarce pojawił się napis **imie**, czyli nazwa trzeciej kolumny (o numerze 2) z tabeli wynikowej zapytania.

Odczytanie jednego wiersza z tabeli wynikowej zapytania

Aby odczytać cały wiersz z tabeli wynikowej zapytania, należy użyć funkcji *pg_fetch_array*:
`$row=pg_fetch_array($result, numerWiersza);`
Tablica będąca wynikiem działania funkcji *pg_fetch_array* składa się z jednego wiersza i tylu kolumn, ile pól zostało zwróconych przez zapytanie. Jest to więc zawsze tablica jednowymiarowa. Dostęp do poszczególnych jej elementów następuje więc po podaniu jej nazwy i jednego indeksu umieszczonego w nawiasach kwadratowych:
`$row[nr_kolumny]`
lub, ponieważ jest to tablica asocjacyjna:
`$row[nazwa_pola]`

1. W pliku *dane.php*, przed znakami zamykającymi kod PHP (przed ?>) umieść instrukcje:
```
$row = pg_fetch_array($result,1);
print $row[1]." ".$row[imie]."<br>";
```
2. W przeglądarce internetowej odśwież zawartość pliku *dane.php*

W przeglądarce pojawiło się nazwisko (**Jaskowski**) i imię (**Ireneusz**). Nazwisko zostało zwrócone przez zmienną *$row[1]*, a imię *$row[imie]*. W tym drugim przypadku wykorzystany został fakt, że zmienna *$row* jest tablicą asocjacyjną.

Sprawdzenie poprawności wykonania zapytań funkcjonalnych

Jeśli kwerenda skierowana do bazy danych jest zapytaniem funkcjonalnym, można sprawdzić, ile wierszy zostało zmodyfikowanych w bazie danych w wyniku jej wykonania przy pomocy funkcji *pg_affected_rows*:
`$numRows=pg_affected_rows($result);`
Jeśli po wykonaniu tej operacji wartość zmiennej *$numRows* będzie równa zero, oznacza to, że żaden wiersz w bazie danych nie został zmodyfikowany, a więc jeśli operacja dotyczyła zapytania funkcjonalnego, to nie zostało ono poprawnie wykonane lub w bazie danych nie było takich rekordów, które powinny zostać zmienione.

Sprawdzenie poprawności wykonania zapytań funkcjonalnych - przykład

1. W pliku *dane.php*, przed znakami zamykającymi kod PHP (przed ?>) umieść instrukcje:

```
$query = "insert into pracownicy (nazwisko, imie, nip, pesel)
values ('Wielowiejski', 'Tadeusz', '123-000-32-98',
'90091212121')";
$result=pg_exec($con, $query);
print pg_affected_rows($result);
```

2. W przeglądarce internetowej odśwież zawartość pliku *dane.php*

W przeglądarce pojawiła się liczba 1. Najpierw zmiennej *$query* przypisano instrukcję dodającą jeden rekord do tabeli *pracownicy*. Następnie zapytanie to wykonano, a jego wynik przypisano zmiennej *$result*. Na zakończenie, przy pomocy instrukcji *print*, wyświetlono wynik działania funkcji *pg_affected_rows*, która zwróciła liczbę 1, gdyż zmodyfikowany (dołączony) został jeden rekord do tabeli *pracownicy*.

W poniższym tekście w miejsca oznaczone od (1) do (11) wstaw odpowiednie określenia spośród następujących (niektóre nie zostaną wykorzystane, inne mogą być wykorzystane kilka razy):

a) dwóch
b) trzech
c) czterech
d) pg_affected_rows
e) pg_changed_rows
f) pg_col_name
g) pg_connect
h) pg_exec
i) pg_fetch_array
j) pg_field_name
k) pg_numcols
m) pg_numfields
n) pg_numrows
o) pg_result
p) pg_row

Jako odpowiedź na pytanie wpisz ciąg znaków (bez spacji) oznaczających kolejno wstawiane wyrazy. Przykładowa odpowiedź:
cbefadnmhepo
oznacza:1c, 2b, 3e, 4f, itd.

Do połączenia z bazą danych służy funkcja (1), która wymaga podania jednego parametru składającego się z minimum (2) elementów. Aby wykonać zapytanie należy użyć funkcji (3), która wymaga podania (4) parametrów. Liczbę wierszy zwróconych przez zapytanie zwraca funkcja (5) a liczbę kolumn (6). Jedną wartość z tabeli wynikowej zapytania zwraca funkcja (7), która wymaga podania (8) parametrów. Nazwę kolumny z tabeli wynikowej zapytania zwraca funkcja (9), a cały wiersz można odczytać przy pomocy funkcji (10). Liczbę rekordów zmodyfikowanych w wyniku wykonania zapytania funkcjonalnego zwraca funkcja (11).