Funkcje pierwotne obsługiwane przez serwer bazy danych

Serwer MySQL udostępnia szereg funkcji pierwotnych, dzięki którym można zrealizować wiele zadań, zamiast samodzielnie pisać skomplikowane programy. Wśród funkcji pierwotnych MySQL można wyróżnić:

- funkcje sterowania przebiegiem wykonania zapytania
- funkcje tekstowe operujące na ciągach znaków.
- funkcje liczbowe operujące na liczbach.
- funkcje daty i czasu operujące na danych typu data i godzina.
- funkcje konwersji służące do zmiany typu danych.
- funkcje kryptologiczne.

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 sterujące wykonaniem zapytania

Wśród funkcji sterujących przebiegiem wykonania zapytania do najpopularniejszych należą trzy funkcje: **IF()**, **IFNULL()** , **NULLIF()** oraz **CASE()**.

1. Funkcja IF()

Funkcja **IF()** umożliwia warunkowe użycie wyrażeń języka SQL. Składnia:

IF (warunek, wartośćGdyPrawda, wartośćGdyFałsz)

Na przykład poniższe zapytanie zwraca informację jak zarabia nasz pracownik:

```
SELECT NazwiskoPracownika, ImiePracownika,
IF(PlacaPracownika > 2000, 'Zarabia dobrze', 'Zarabia źle') AS Płaca
FROM pracownicy;
```

Zapytanie z funkcją **IF()** może także operować na wyrażeniach wyliczeniowych:

```
SELECT NazwiskoPracownika, ImiePracownika,
IF(PlacaPracownika < 2000, PlacaPracownika*1.2, PlacaPracownika)
↳ AS "Płaca po podwyżce"
FROM pracownicy;
```

Funkcja **IF()** najczęściej jest stosowana w sekcji **SELECT** gdyż pozwala automatycznie wypisywać wartości, które musiałby być wpisywane ręcznie.

2. Funkcja IFNULL()

Funkcja ta służy do zastąpienia wartości **NULL** inną wartością. Funkcja **IFNULL( )** ma składnię:

```
IFNULL (NazwaPola, wartośćGdyPoleMaWartoscNull)
```

Na przykład:

```
SELECT NazwaProducenta,
IFNULL(TelefonProducenta, 'Numer nieznany') AS Telefon
FROM producenci;
```

3. Funkcja NULLIFF()

Funkcja zwraca wartość **NULL** gdy obydwa sprawdzane przez nią argumenty mają tą samą wartość. W przeciwnym wypadku zwracana jest wartość pierwszego parametru. Zatem możemy ją wykorzystać, aby zamienić określoną wartość w tabeli na wartość **NULL**.

```
SELECT NazwiskoKlienta, NULLIF(MiastoKlienta, 'Katowice') AS "Miasta Klientów "
FROM klienci;
```

4. Funkcja CASE()

Działa, jak wielokrotnie użyta instrukcja **IF()**. Składnia :

```
CASE
    WHEN warunek THEN Wynik
    WHEN warunek THEN Wynik
    ELSE  Wynik
END
```

Na przykład:

```
SELECT NazwiskoPracownika, ImiePracownika, DataZatrudnieniaPracownika,
CASE
    WHEN DataZatrudnieniaPracownika < '2013-01-01'  THEN  'Starszy asystent'
    WHEN DataZatrudnieniaPracownika < '2014-01-01'  THEN  'Asystent'
    ELSE  'Kontraktowy'
END AS "Status pracownika "
FROM pracownicy;
```

Funkcje tekstowe

Argumentem funkcji tekstowych są ciągi znaków (dane typu **char**, **varchar** lub **text**). Typ danych zwracanych przez funkcje tekstowe jest podstawą do ich dalszego podziału: wyróżniamy funkcje tekstowe zwracające wartość znakową i funkcje tekstowe zwracające wartość liczbową.

1. Funkcje tekstowe zwracające tekst

Funkcja CONCAT()

Funkcja dokonuje połączenia łańcuchów podanych jako jej argumenty. Składania:

CONCAT(łancuch1, łańcuch2, …)

Przykład:

```
SELECT CONCAT(klienci.ImieKlienta, ' ',klienci.NazwiskoKlienta) AS Klient,
sprzedaze.DataSprzedazy
FROM klienci INNER JOIN sprzedaze
USING (IDklienta);
```

Jeżeli którykolwiek z podanych elementów ma wartość **NULL**, zwracana jest również wartość **NULL**.

Funkcja CONCAT_WS()

Funkcja dokonuje połączenia łańcuchów, rozdzielając je separatorem. Składnia:

CONCAT_WS(separator, łancuch1, łańcuch2, …)

W odróżnieniu od funkcji **CONCAT** wartość **NULL** jest tu ignorowana. Separator o wartości **NULL** powoduje, że funkcja zwraca **NULL**;

```
SELECT CONCAT_WS(' ~ ', producenci.NazwaProducenta, rowery.NazwaRoweru) AS
Rowery
FROM producenci INNER JOIN rowery
USING (IDproducenta);
```

Funkcja UPPER()

Za pomocą tej funkcji wszystkie litery przekazanego jej argumentu zostaną zamienione na duże litery:

```
SELECT UPPER(NazwiskoKlienta), ImieKlienta
FROM klienci;
```

Funkcja LOWER()

Działa odwrotnie do funkcji **UPPER().**

Funkcja LEFT()

Funkcja zwraca podłańcuch o zadanej długości z podanego łańcucha licząc od lewej. Składnia funkcji:

LEFT('łańcuchZnaków', liczbaZnaków)

Na przykład poniższe zapytanie zwraca inicjały pracowników:

```
SELECT CONCAT(LEFT(ImiePracownika,1), LEFT(NazwiskoPracownika,1))
AS  "Inicjały pracowników", Imiepracownika, NazwiskoPracownika
FROM pracownicy;
```

Funkcja RIGHT()

Funkcja zwraca podłańcuch o zadanej długości z podanego łańcucha licząc od prawej strony (od końca). Na przykład poniższe zapytanie zwraca Imiona i nazwiska klientów, których identyfikatory kończą się cyfrą 1:

```
SELECT ImieKlienta, NazwiskoKlienta
FROM klienci
WHERE RIGHT(IDklienta ,1) = 1;
```

Funkcja TRIM()

Funkcja usuwa spacje na początku jak i na końcu łańcucha znaków. Składnia:

TRIM('łańcuchZnaków')

Funkcja LRIM()

Funkcja usuwa spacje na początku łańcucha znaków. Składnia:

LRIM('łańcuchZnaków')

Funkcja RTRIM()

Funkcja usuwa spacje na końcu łańcucha znaków. Składnia:

RTRIM('łańcuchZnaków')

Funkcja REPLACE()

Funkcja w podanym łańcuchu znaków wyszukuje podany ciąg i zastępuje go innym podanym ciągiem. Składnia:

REPLACE('łańcuchZnaków', 'staryCiąg', 'nowyCiąg')

Jeżeli trzeci argument nie zostanie podany, z podanego łańcucha zostaje wycięty ciąg podany jako drugi argument funkcji. Przykład użycia funkcji:

```
SELECT REPLACE(NazwaProducenta, 'BMX', 'Romet')
AS  "Nowe nazwy producentów", MiastoProducenta, UlicaProducenta,
NrDomuProducenta
FROM producenci;
```

Funkcja SUBSTRING()

Funkcja zwraca określoną liczba znaków z łańcucha tekstowego, począwszy od podanej pozycji. Składnia:

```
SUBSTRING('łańcuchZnaków', pozycja, długość)
```

Jeżeli nie podamy liczby zwracanych znaków, zwrócone zostaną wszystkie znaki występujące po pozycji określonej przez drugi parametr. Podanie ujemnej wartości drugiego parametru spowoduje, że znaki będą liczone od prawej do lewej. Przykład:

```
SELECT SUBSTRING(NazwaWojewodztwa, 1,3) AS "Skróty nazw województw”
FROM wojewodztwa;
```

Funkcja SPACE()

Działanie funkcji powoduje zwrócenie określonej liczby spacji. Przykład:

```
SELECT CONCAT(ImieKlienta, SPACE(5), NazwiskoKlienta) AS Klient
FROM klienci;
```

Funkcja REVERSE()

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

```
REVERSE('łańcuchZnaków')
```

Funkcja REPEAT()

Funkcja zwraca podany łańcuch znaków, powtórzony określoną liczbę razy;

```
REPEAT('łańcuchZnaków', liczbaPowtórzeń)
```

Funkcja PASSWORD()

Funkcja szyfruje podany łańcuch znaków. Proces szyfrowania jest nieodwracalny. Funkcja jest przeznaczona do wstawiania wartości do kolumny haseł w tabeli **user**. Składnia:

```
PASSWORD('łancuchZnaków')
```

Przykład:

```
UPDATE user SET password=PASSWORD('hasło')
WHERE user = 'root';
```

2. Funkcje operujące na łańcuchach zwracające liczby

Funkcja LENGTH()

Funkcja zwraca długość łańcucha podanego jako parametr jej wywołania. Jeżeli wywołamy funkcję z wartością **NULL**, funkcja zwróci wartość **NULL**. Składnia:

```
LENGTH('łańcuchZnaków')
```

Przykład:

```
SELECT NazwaRoweru
FROM rowery
WHERE LENGTH(NazwaRoweru) <=5;
```

Funkcja INSTR()

W wyniku działania funkcji otrzymamy pozycję, na której w łańcuchu podanym jako pierwszy parametr, została znaleziona fraza podana jako drugi parametr. Jeżeli szukana fraza nie zostanie znaleziony, funkcja zwróci **0**. Składnia:

```
INSTR('łańcuchZnaków', 'szukanaFraza')
```

Przykład:

```
SELECT NazwiskoKlienta, INSTR(NazwiskoKlienta, 'ski') AS Wynik
FROM Klienci;
```

Funkcja STRCMP()

Funkcja porównuje dwa łańcuchy i zwraca **0**, jeżeli są one równe, **-1** jeżeli pierwszy poprzedza drugi w kolejności alfabetycznej, i **1** jeżeli pierwszy występuje w kolejności alfabetycznej po drugim. Składnia:

```
STRCMP('łańcuchZnaków1', 'łańcuchZnaków2')
```

Przykład:

```
SELECT NazwiskoKlienta, STRCMP(NazwiskoKlienta, 'Nowak') AS Wynik
FROM Klienci;
```

Funkcje liczbowe

Funkcje te mają za zadanie wykonywanie operacji matematycznych. Większość z nich pobiera i zwraca liczby zmiennoprzecinkowe.

ABS()

Funkcja zwraca wartość bezwzględną z podanego argumentu. Składnia:

ABS(liczba)

ACOS()

Funkcja zwraca arcus cosinus z podanej liczby. Składnia:

ACOS(liczba)

ASIN()

Funkcja zwraca arcus sinus z liczby. Składnia:

ASIN(liczba)

ATAN()

funkcja zwraca arcus tangens z liczby. Składnia:

ATAN(liczba)

CEILING()

Funkcja zaokrągla liczbę zmiennoprzecinkową do najbliższej, większej liczby całkowitej. Składnia:

CEILING(liczba)

Przykład:

```
SELECT NazwaRoweru, CenaJednostkowa AS "Stara cena",
CEILING(CenaJednostkowa/3) AS "Nowa cena"
FROM rowery;
```

SIN()

Funkcja zwraca sinus z kąta wyrażonego w radianach. Składnia:

SIN(liczba)

COS()

Funkcja zwraca cosinus z kąta wyrażonego w radianach. Składnia:

COS(liczba)

COT()

Funkcja zwraca cotangens z kąta wyrażonego w radianach. Składnia:

COT(liczba)

TAN(liczba)

Funkcja zwraca tangens z kąta wyrażonego w radianach. Składnia:

TAN(liczba)

DEGREES()

Funkcja zamienia radiany na stopnie. Składnia:

DEGREES(liczba)

FLOOR()

Funkcja zaokrągla liczbę zmiennoprzecinkową do najbliższej mniejszej liczby całkowitej. Składnia:

FLOOR(liczba)

GREATEST()

Funkcja zwraca największą wartość z podanej listy. Składnia:

GREATEST(liczba1, liczba2, …)

LEAST()

Funkcja zwraca najmniejszą wartość z podanej listy. Składnia:

LEAST(liczba1, liczba2 ,...)

PI()

Bezargumentowa funkcja zwracająca kilka pierwszych cyfr liczby π;

POW()

Funkcja podnosi pierwszy argument do potęgi podanej jako jej drugi argument. Składnia:

POW(liczba1, liczba2)

RADIANS()

Funkcja zamienia stopnie na radiany. Składnia:

RADIANS(liczbaStopni)

ROUND()

Funkcja zaokrągla liczbę zmiennoprzecinkową do całkowitej lub opcjonalnie do podanej liczby cyfr po przecinku. Składnia:

ROUND(liczba, [dokładność])

SIGN()

Funkcja sprawdza znak liczby i zwraca **-1**, jeżeli liczba jest ujemna, lub **1**, jeżeli jest dodatnia. Składnia:

SIGN(liczba)

SQRT()

Funkcja zwraca pierwiastek kwadratowy z liczby. Składnia:

SQRT(liczba)

TRUNCATE()

Funkcja powoduje obcięcie liczby do określonej liczby cyfr po przecinku. Pierwszy parametr jest liczbą do obcięcia, drugi wskazuje, do ilu pozycji chcemy liczbę skrócić. Ujemny drugi argument powoduje dodanie określonej liczby zer z lewej strony przecinka. Składnia:

TRUNCATE(liczba, ile)

RAND()

Bezargumentowa funkcja zwraca pseudolosową liczbę z przedziału od **0** do **1**. Serwer MySQL oblicza tę wartość dla każdego wiersza wyniku, co pozwala użyć funkcji **RAND()** do losowego wybierania danych z tabeli. Składnia:

RAND()

Przykład:

```
SELECT *
FROM klienci
WHERE IDklienta = TRUNCATE(RAND( )*10,0);
```

MOD()

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

MOD(liczba1, liczba2)

FORMAT()

Funkcja zwraca przekazaną jako pierwszy argument wywołania liczbę z określoną przez drugi argument liczbą miejsc po przecinku. Składnia:

FORMAT(liczba1, liczba2)

Funkcje daty i czasu

Są to funkcje zwracające daty lub czas bądź umożliwiające formatowanie danych tego typu, jak i ich przetwarzanie.

CURDATE()

Funkcja bezargumentowa zwracająca bieżącą datę.

CURTIME()

Funkcja bezargumentowa zwracająca bieżący.

CURRENT_TIMESTAMP()

Funkcja bezargumentowa zwracająca bieżącą datę i czas.

NOW()

Funkcja bezargumentowa zwracająca bieżącą datę i czas.

DATE_ADD()

Funkcja umożliwia obliczenie wyniku zmiany daty (pierwszy argument) o ilość jednostek czasu (interwał czasowy) określoną przez drugi argument. Składnia:

DATE_ADD(data/czas, interval x typ)

Typy i wymagane przez nie formaty zostały przedstawione w tabeli poniżej.

Typ	*Format*
DAY	dni
DAY_HOUR	dni godziny
DAY_MINUTE	dni godziny:minuty
DAY_SECOND	dni godziny:minuty:sekundy
HOUR	godziny
HOUR_MINUTE	godziny:minuty
HOUR_SECOND	godziny :minuty: sekundy

Typ	*Format*
MINUTE	minuty
MINUTE_SECOND	minuty: sekundy
MONTH	miesiące
SECOND	sekundy
YEAR	Lata
YEAR_MONTH	lata-miesiące

Przykład użycia funkcji:

```
SELECT NOW() AS  "Dzisiejsza data",
DATE_ADD(CURDATE(), INTERVAL 4 MONTH) AS "Data wyliczona";
```

Na przykład zapytanie, które oblicza datę końca rocznej gwarancji na zakupiony w naszej hurtowni rower:

```
SELECT rowery.NazwaRoweru,
CONCAT(klienci.ImieKlienta, klienci.NazwiskoKlienta) AS Kupujący,
sprzedaze.DataSprzedazy AS "Data sprzedaży",
DATE_ADD(sprzedaze.DataSprzedazy, INTERVAL 12 MONTH) AS "Data końca gwarancji"
FROM klienci INNER JOIN sprzedaze USING (IDklienta) INNER JOIN
szczegolysprzedazy USING (IDsprzedazy) INNER JOIN rowery USING(IDroweru);
```

DATE_SUB()

Funkcja odejmuje określony interwał czasowy od podanej daty (czasu). Składnia:

```
DATE_SUB(data/czas, interval x typ)
```

Odejmowany od podanej daty interwał czasowy określamy w ten sam sposób, co w przypadku funkcji **DATE_ADD()**. Przykład:

```
SELECT DATE_SUB('2015-05-04', INTERVAL 100 DAY) AS "Dzień studniówki"
```

DATEDIFF()

Wynikiem działania funkcji jest liczba dni, które dzielą daty podane jako argumenty funkcji. Składnia:

```
DATEDIFF(data1, data2)
```

Na przykład, aby sprawdzić, ile dni upłynęło od dat sprzedaży do dzisiaj:

```
SELECT DataSprzedazy AS "Data sprzedaży",
CURDATE( ) AS "Dzisiejsza data ",
DATEDIFF(CURDATE( ), DataSprzedazy) AS "Do dziś upłynęło dni "
FROM sprzedaze;
```

DATE_FORMAT()

Funkcja formatuje datę zgodnie z podanym formatem. Łańcuch formatujący może zawierać dowolną liczbę kodów rozpoczynających się od znaku procenta, symbolizujących poszczególne elementy daty. Reszta znaków łańcucha zostaje zwrócona w postaci dosłownej. Składnia

DATE_FORMAT(data, format)

Przykładowe kody formatujące zostały wymienione w tabeli poniżej.

Kod	*Opis*	*Przykłady*
%a	Skrócona nazwa dnia tygodnia	SUN...SAT
%b	Skrócona nazwa miesiąca	JAN... DEC
%c	Numer miesiąca nie poprzedzony zerem	1... 12
%D	Dzień miesiąca z angielskim przyrostkiem	1ST, 2ND, 3RD, 4TH, ...
%d	Dzień miesiąca poprzedzony zerem	01...31
%e	Dzień miesiąca	1...31
%M	Nazwa miesiąca	JANUARY...DECEMBER
%m	Numer miesiąca poprzedzony zerem	01...12
%p	AM lub PM	AM, PM
%r	Czas na zegarze 12-godzinnym	01:15:30
%l	Czas na zegarze 24-godzinnym	15:32:00
%W	Nazwa dnia tygodnia	SUNDAY...SATURDAY

Przykład:

```
SELECT DATE_FORMAT(NOW( ),' %W %e %M %Y ');
```

TIME_FORMAT()

Funkcja pozwala formatować dane o czasie. Format tworzy się według tych samych zasad co przy funkcji **DATE_FORMAT()**.

DAYNAME()

Funkcja zwraca nazwę dnia tygodnia dla podanej daty, Składnia:

DAYNAME(data)

Przykład:

```
SELECT DataSprzedazy, DAYNAME(DataSprzedazy) AS "Dzień tygodnia "
FROM sprzedaze;
```

MONTHNAME()

Funkcja zwraca nazwę miesiąca dla podanej daty. Składnia:

MONTHNAME(data)

MONTH()

Funkcja zwraca miesiąc z podanej daty jako liczbę;

MONTH(data)

YEAR()

Funkcja zwraca rok z podanej daty;

YEAR(data)

Funkcje HOUR(), MINUTE(), SECOND()

Funkcje zwracają odpowiednio godziny, minuty i sekundy z czasu przekazanego jako parametr wywołania. Przykład:

```
SELECT CURTIME() AS "Odczytany czas ", HOUR(CURTIME()) AS Godziny,
MINUTE(CURTIME()) AS Minuty, SECOND(CURTIME()) AS Sekundy
```

Funkcje DAYOFMONTH(), DAYOFWEEK(), DAYOFYEAR()

Funkcje **DAYOFMONTH()**, **DAYOFWEEK()**, **DAYOFYEAR()** zwracają odpowiednio:

- numer dnia miesiąca (liczba z zakresu od 1 do 31), Składnia:

  DAYOFMONTH(data)

- numer dnia tygodnia (liczba z zakresu od 1 do 7, gdzie 1 oznacza niedzielę — pierwszy dzień tygodnia, a 7 sobotę)

  DAYOFWEEK(data)

- numer dnia roku (liczba z zakresu od 1 do 365). Składnia:

  DAYOFYEAR(data)

Przykład:

```
SELECT sprzedaze.DataSprzedazy,
DAYNAME(sprzedaze.DataSprzedazy) AS "Dzień słownie",
DAYOFWEEK(sprzedaze.DataSprzedazy) AS "Dzień liczbowo"
FROM sprzedaze;
```

Funkcje konwersji

Aby móc porównywać ze sobą dane, np. wpisywane przez użytkownika, często trzeba je najpierw przekonwertować. Bowiem:

- **dane tekstowe** – mogą np. zawierać tekst lub kombinacje tekstu i liczb (pole adres). Dane tego typu mogą także zawierać również liczby, na których nie są przeprowadzane obliczenia, takie jak numery telefonów, numery katalogowe i kody pocztowe;
- **dane binarne** – mogą zawierać dowolne dane. Mogą to być zarówno długie teksty, jak i grafika czy pliki multimedialne.
- **dane liczbowe** – zawierają dane liczbowe, na których są przeprowadzane obliczenia, przy czym dwa ostatnie typy są zmiennoprzecinkowe, czyli takie dane przechowywane są z określoną dokładnością.
- **dane daty**– przechowują dane dotyczące daty i czasu.
- **dane logiczne** przyjmują wartość **0** oznaczającą fałsz i **1** oznaczające prawdę.

Do konwersji danych służą funkcje konwersji. Poniżej podano najważniejsze z nich.

ASCII()

W wyniku działania funkcji będzie zwrócony kod ASCII znaku podanego jako argument wywołania. Jeżeli jako argument podamy ciąg znaków, za pomocą tej funkcji zostanie obliczony i zwrócony kod ASCII pierwszego znaku w ciągu. Przykład:

```
SELECT ASCII(ImieKlienta) AS "Imię", ASCII(NazwiskoKlienta) AS "Nazwisko"
FROM klienci;
```

CHR()

Działanie funkcji jest przeciwieństwem działania funkcji **ASCII()**. Zamiast zamiany tekstu na liczbę przeprowadza konwersję liczby na odpowiadające jej znaki kodu ASCII. Przykład:

```
SELECT CHR(IDklienta) AS "Identyfikator", NazwiskoKlienta
FROM klienci;
```

BIN()

Funkcja zwraca binarną reprezentację podanej liczby dziesiętnej. Przykład

```
SELECT IDklienta, BIN(IDklienta) AS "ID binarny", NazwiskoKlienta,
FROM klienci;
```

CAST()

Funkcja pozwala przekonwertować (rzutować) dane przekazane jako pierwszy argument wywołania na typ podany jako drugi argument funkcji. Przykład:

```
SELECT CAST(DataSprzedazy AS date)
FROM sprzedaze;
```

Funkcje kryptologiczne

Funkcje kryptologiczne zajmują się zabezpieczeniem informacji.

ENCODE()

Funkcja zwraca szyfrogram podanego ciągu znaków. Do szyfrowania używane jest hasło podane jako drugi parametr wywołania. Przykład:

```
SELECT ENCODE(UlicaKlienta,'1234')
FROM klienci;
```

DECODE()

Funkcja deszyfruje zaszyfrowane funkcją ENCODE() szyfrogramy.

SHA1()

Funkcja wylicza sygnaturę podanego ciągu znaków.

```
SELECT SHA1(UlicaKlienta,'1234')
 FROM klienci;
```

PASSWORD()

Funkcja wylicza i porównuje hasła z bazy z hasłami podanymi przez użytkowników. Do sprawdzania tożsamości należy używać sygnatur haseł, nie samych haseł.

Funkcje specjalne

Oprócz funkcji opisanych wcześniej, każdy serwer bazodanowy obsługuje pewną liczbę funkcji specjalnych, systemowych i metadanych. W mySQL najczęściej wykorzystywane są dwie funkcje tego typu:

LAST_INSERT_ID()

Funkcja zwraca ostatni identyfikator wygenerowany podczas wstawiania przez bieżącego użytkownika wiersza do tabeli. Tego typu informacje są z reguły potrzebne programistom aplikacji klienckich, żeby mogli oni powiązać wstawiony właśnie wiersz z wcześniej zapisanymi w bazie danymi.

LOAD_FILE()

Funkcja pozwala wczytać zawartość pliku do bazy.