Oprócz tabel i widoków w bazach danych możemy tworzyć własne funkcje, procedury i specjalny typ procedur, które będą automatycznie wywoływanie przez MySQL w memencie wstawiania, usuwania lub modyfikowania określonych danych, czyli wyzwalacze. W tym odcinku dowiesz się, jak tworzyć i jak korzystać z tego typu obiektów.

Materiał z tego odcinka to tylko wstęp do bardzo obszernego tematu, jakim jest tworzenie funkcji procedur i wyzwalaczy. Jeżeli uda mi się w ten sposób przekonać Cię do zalet tego rozwiązania, dokładniejsze informacje znajdziesz m.in. w dokumentacji MySQL-a (http://dev.mysql.com/doc).

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.

Tworzenie funkcji

Wykonanie instrukcji CREATE FUNCTION spowoduje utworzenie funkcji użytkownika. Funkcja z listingu 12.1 oczekuje na jeden parametr — datę — i zwraca odczytany z tej daty rok.

Listing 12.1. Przykładowa funkcja użytkownika i sposób jej wywołania

```
CREATE FUNCTION fn_data

(data DATE)

RETURNS CHAR(20)

RETURN YEAR(data);

Query OK, 0 rows affected (0.01 sec)



SELECT fn_data('2001-1-1');

+---------------------+

| fn_data('2001-1-1') |

+---------------------+

| 2001                |

+---------------------+
```

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.

Kolejny przykład pokazuje nieco zmienioną wersję naszej funkcji. Tym razem funkcja oczekuje na dwa parametry i zwraca rok i nazwę miesiąca oddzielone podanym znakiem (listing 12.2).

Listing 12.2. Rozbudowana funkcja i jej wywołanie w ramach instrukcji odczytującej dane z tabeli

```
CREATE FUNCTION fn_daty
 (data date, delimeter char(1))
RETURNS char(20)
RETURN CONCAT(YEAR(data),delimeter,  MONTHNAME(data));
Query OK, 0 rows affected (0.00 sec)

SELECT date_placed, fn_daty(date_placed,':')
FROM orderinfo;
+-------------+--------------------------+
| date_placed | fn_daty(date_placed,':') |
+-------------+--------------------------+
| 2000-03-13  | 2000:March               |
| 2000-06-23  | 2000:June                |
| 2000-09-02  | 2000:September           |
| 2000-09-03  | 2000:September           |
| 2000-07-21  | 2000:July                |
+-------------+--------------------------+
5 rows in set (0.00 sec)
```

Usuwanie funkcji

Wykonanie instrukcji DROP FUNCTION spowoduje usunięcie wybranych funkcji użytkownika (listing 12.3).

Listing 12.3. Administrator albo osoba mająca nadane odpowiednie uprawnienia może usuwać funkcje użytkownika z bazy

```
DROP FUNCTION fn_data;
Query OK, 0 rows affected (0.00 sec)
```

Procedury składowane

Procedury składowane są zbiorami instrukcji języka SQL zapisanymi pod wspólną nazwą i wywoływanymi jak pojedyncza instrukcja.

Procedury składowane umożliwiają:

1. Przekazywanie parametrów wywołania.

2. Wykonywanie prawie wszystkich instrukcji języka SQL, w tym wywoływania innych procedur składowanych.
3. Zwracanie dowolnej liczby wyników do programu, który wywołał procedurę.
4. Zwracanie informacji o udanej lub niewykonanej procedurze.

Procedury składowane są powszechnie wykorzystywane w celu:
1. implementowania reguł logiki biznesowej,
2. zabezpieczenia obiektów bazy danych przed bezpośrednim dostępem użytkowników,
3. chronienia bazy danych przed atakami polegającymi na iniekcji kodu SQL,
4. poprawienia wydajności często wykonywanych instrukcji,
5. zminimalizowania obciążenia sieci (zamiast wysyłać całe instrukcje języka SQL użytkownik wywołuje jedynie procedurę, wysyłając jej nazwę i przekazując parametry jej wywołania).

Tworzenie procedur składowanych

Wykonanie instrukcji CREATE PROCEDURE spowoduje dodanie nowej procedury składowanej (listing 12.4).

Listing 12.4. W tym przypadku ciało procedury składa się z pojedynczej instrukcji, a więc nie trzeba było umieszczać jej w bloku BEGIN … END. Przykładowa procedura oczekuje na jeden parametr wywołania i nie zwraca żadnych informacji

```
CREATE PROCEDURE przecena

(IN nazwa varchar (64))

UPDATE item

SET sell_price = sell_price*0.9

WHERE description = nazwa;

Query OK, 0 rows affected (0.01 sec)



SELECT description, sell_price

FROM item

WHERE description LIKE 'f%';

+-------------+------------+

| description | sell_price |

+-------------+------------+

| Fan Small   | 8.86       |

| Fan Large   | 11.22      |

+-------------+------------+

CALL przecena ('Fan Small');
```

Query OK, 1 row affected, 1 warning (0.03 sec)

```
SELECT description, sell_price
FROM item
WHERE description LIKE 'f%';
+-------------+------------+
| description | sell_price |
+-------------+------------+
| Fan Small   | 7.97       |
| Fan Large   | 11.22      |
+-------------+------------+
```

Usuwanie procedur składowanych

Nieużywane procedury składowane możemy usunąć poprzez instrukcję DROP PROCEDURE (listing 12.5).

Listing 12.5. Usuwamy procedurę składowaną

```
DROP PROCEDURE przecena;
Query OK, 0 rows affected (0.00 sec)
```

Wyzwalacze

Wyzwalacze są specjalnym typem procedur składowanych powiązanych z wybranymi tabelami i wywoływanych wykonaniem instrukcji języka SQL: INSERT, UPDATE albo DELETE.

Niemożliwe jest bezpośrednie wywołanie wyzwalacza za pomocą dyrektywy CALL.

Instrukcje wykonane w ramach ciała wyzwalacza traktowane są jako fragment transakcji jawnie lub niejawnie rozpoczętej przez użytkownika, który odwołał się do danych przechowywanych w powiązanej z wyzwalaczem tabeli. Wynika z tego, że wyzwalacz może zatwierdzić (wykonując instrukcję COMMIT TRANSACTION) lub wycofać (instrukcją ROLLBACK TRANSACTION) zmiany wprowadzone przez użytkownika.

Podstawowym zastosowaniem wyzwalaczy jest wymuszenie integralności danych, zwłaszcza ich zgodności z regułami logiki biznesowej. Wyzwalacze umożliwiają m.in.:

1. Kaskadowe aktualizowanie danych w powiązanych tabelach.
2. Sprawdzanie poprawności danych na podstawie wartości przechowywanych w dowolnych tabelach (w przeciwieństwie do zawężenia CHECK, za pomocą którego możemy odwołać się jedynie do bieżącej tabeli).
3. Jednoczesne sprawdzanie danych zmodyfikowanych w dowolnej liczbie wierszy tabeli.

4.

Wywoływanie predefiniowanych lub zdefiniowanych przez użytkownika komunikatów błędu.

5.

Monitorowanie aktywności użytkowników.

6.

Modyfikacje danych w bazach niespełniających wymogów trzeciej postaci normalnej. W bazach tego typu prawdopodobnie przechowywane są informacje nadmiarowe (redundantne) i modyfikacja np. numeru telefonu w jednej tabeli może wiązać się z koniecznością zmiany tego numeru w innych tabelach.

Wyzwalacze a ograniczenia i procedury składowane

Obiekty obu tych typów w pewnym podstawowym zakresie mogą pełnić tę samą funkcję — wymuszenia integralności przechowywanych danych. Jednak zakres i typ przeprowadzanych przez nie akcji są różne:

1.

Wyzwalacze w przeciwieństwie do ograniczeń wywoływane są w odpowiedzi na akcje użytkownika. Wynika z tego, że dopiero po wykonaniu instrukcji wyzwalany jest wyzwalacz, natomiast warunki ograniczeń sprawdzane są przed wykonaniem instrukcji języka SQL.

2.

Konsekwencją poprzedniego punktu jest kolejność, w jakiej wywoływane są wyzwalacze i ograniczenia — najpierw sprawdzane są warunki zdefiniowane w ograniczeniach, a po ich pomyślnym sprawdzeniu wywoływany jest wyzwalacz.

3.

Wyłącznie właściciel tabeli może utworzyć powiązany z nią wyzwalacz. Uprawnienie do tworzenia wyzwalaczy nie może zostać nikomu nadane czy przekazane.

4.

Wyzwalacze nie mogą zostać powiązane z widokami oraz tabelami tymczasowymi.

5.

Wyzwalacze mogą przetwarzać jednocześnie wiele wierszy tabeli. Możliwe jest również warunkowe przetwarzanie poszczególnych wierszy.

Tak jak możliwe jest zdefiniowanie dla tabeli dowolnej liczby zawężeń (wyjątkiem jest ograniczenie PRIMARY KEY), możliwe jest również utworzenie dowolnej liczby powiązanych z nią wyzwalaczy.

Natomiast tym, co odróżnia wyzwalacze od procedur składowanych (oprócz sposobu ich wywoływania) jest fakt, że wyzwalacze z reguły nie zwracają żadnych danych.

Tworzenie wyzwalaczy

Aby utworzyć wyzwalacz, należy wykonać instrukcję CREATE TRIGGER. W tym punkcie przyjrzymy się wykorzystaniu wyzwalaczy do monitorowania zmian nazw produktów.

W pierwszej kolejności utworzymy tabelę, w której będziemy przechowywać historię nazw produktów (listing 12.6).

Listing 12.6. Tworzymy tabelę pomocniczą, w której wyzwalacz automatycznie będzie zapisywał interesujące nas dane

```
CREATE TABLE dziennik(

id int auto_increment primary key,

komunikat varchar(255));

Query OK, 0 rows affected (0.10 sec)
```

Następnie tworzymy wyzwalacz uruchamiany instrukcją UPDATE wykonaną dla tabeli item, w ramach którego:

1. Sprawdzamy, czy została zmieniona nazwa produktu.
2. Sprawdzamy, czy nowa albo stara nazwa produktu nie jest nieokreślona.
3. Jeżeli któryś z tych warunków jest spełniony, wstawiamy do tabeli dziennik wiersz z oryginalną i zmienioną nazwą produktu (listing 12.7).

Ponieważ w ciele wyzwalacza poszczególne instrukcje oddziela znak średnika, na czas jego tworzenia należy zmienić znak końca instrukcji — inaczej zostanie on natychmiast zinterpretowany, a nie zapisany jako fragment ciała wyzwalacza.

Listing 12.7. Wyzwalacz zapisujący historię zmian nazw produktów

```
DELIMITER //

CREATE TRIGGER tgr_nazwy

BEFORE UPDATE ON item

FOR EACH ROW

BEGIN

IF new.description != old.description

OR new.description is null and old.description is not null

OR old.description is null and new.description is not null then

INSERT INTO dziennik(komunikat)

VALUES ( CONCAT('nazwa: ', COALESCE(old.description,'NULL'), ' -> '

, COALESCE(new.description,'NULL'))

);

END IF;

END;//

Query OK, 0 rows affected (0.16 sec)
```

Jak wynika z analizy kodu wyzwalacza, w specjalnej tabeli systemowej OLD przechowywane są oryginalne wartości, a w tabeli NEW — wartości zmienione przez użytkownika. Przetestujmy nasze rozwiązanie (listing 12.8).

Listing 12.8. Jakakolwiek zmiana nazwy produktu zostanie odnotowana

```
UPDATE item

SET description = 'Linux DVD'

WHERE item_id=3;

Query OK, 1 row affected (0.03 sec)

Rows matched: 1  Changed: 1  Warnings: 0


SELECT *

FROM dziennik;

+----+------------------------------+

| id | komunikat                    |

+----+------------------------------+

| 1  | nazwa: Linux CD -> Linux DVD |

+----+------------------------------+
```

Usuwanie wyzwalaczy

Aby usunąć wyzwalacz, należy wykonać instrukcję DROP TRIGGER (listing 12.9).

Listing 12.9. Usuwamy przykładowy wyzwalacz

```
DROP TRIGGER tgr_nazwy;

Query OK, 0 rows affected (0.04 sec)
```