SYSTEMY BAZ DANYCH
SQL

TEMAT 25-05: Wyzwalacze.

Autor dokumentu: Wojciech Galiński	sobota, 19 września 2015 r.	351203 Technik informatyk

ŹRÓDŁA WIEDZY: https://dev.mysql.com/doc/.../create-trigger.html, https://dev.mysql.com/doc/.../trigger-syntax.html, http://webmaster.helion.pl/index.php/kurs-mysql.

Zagadnienia obowiązkowe

1. **Wyzwalacz** – (ang. trigger) to specjalna procedura powiązana z określonym zdarzeniem (INSERT, UPDATE DELETE) w tabeli wskazanej w definicji takiego wyzwalacza.

   Zalety i ograniczenia wyzwalaczy opisane są w podpunkcie „Wyzwalacze” pod adresem: http://webmaster.helion.pl/index.php/kursmysql-funkcje-procedury-skladowane-i-wyzwalacze.

   Wyzwalacza nie można uruchomić samodzielnie – taka procedura jest automatycznie uruchamiana przez SZBD podczas wykonania następujących poleceń języka SQL:
   - ***INSERT*** – wstawianie nowych rekordów;
   - ***UPDATE*** – aktualizowanie istniejących rekordów;
   - ***DELETE*** – usuwanie istniejących rekordów.
2. **Używanie wyzwalaczy** – składa się na to:
   - ***tworzenie wyzwalacza*** – służy do tego instrukcja „***CREATE TRIGGER***”. Oto wzorzec (BEGIN i END używamy w przypadku zastosowania wielu instrukcji w ciele wyzwalacza):

```
DELIMITER //  -- zmieniamy separator instrukcji
CREATE TRIGGER nazwa_wyzwalacza CZAS_ZDARZENIA ZDARZENIE
BEGIN
  instrukcje;
END //
DELIMITER ;  -- przywracamy domyślny separator instrukcji
```

   gdzie: *CZAS_ZDARZENIA* – to jedna z wartości: **BEFORE**, **AFTER**; *ZDARZENIE* – to jedna z wartości: **INSERT**, **UPDATE**, **DELETE**.

   Powyższe parametry definiują czas zadziałania wyzwalacza (przed/po różnymi zdarzeniami związanymi ze zmianą zawartości tabeli).

   PRZYKŁAD (zauważ, że poniższy wyzwalacza działa błędnie – znajdź ten błąd):

```
DELIMITER //
CREATE TRIGGER wyzwalacz_dla_insert BEFORE INSERT ON tabela
FOR EACH ROW
BEGIN
  INSERT INTO dziennik(opis_zmiany)
  VALUES  (  CONCAT( new.id, ": => ", IF(new.wartosc IS NULL, "NULL", new.wartosc) )  );
END //
DELIMITER ;
```

   A poniżej przykład wyzwalacza zastępującego w MySQL opcję CHECK z PostgreSQL:

```
DELIMITER $$
CREATE TRIGGER `test_before_insert` BEFORE INSERT ON `Test`
FOR EACH ROW
BEGIN
  IF CHAR_LENGTH( NEW.ID ) < 4 THEN
    SIGNAL SQLSTATE '12345';
    SET MESSAGE_TEXT := 'check constraint on Test.ID failed';
    END IF;
END$$
DELIMITER ;
```

   - ***usuwanie wyzwalacza*** – służy do tego instrukcja „**DROP TRIGGER**”. Oto wzorzec:

```
DROP TRIGGER nazwa_wyzwalacza;
```

   PRZYKŁAD: `DROP TRIGGER wyzwalacz_dla_insert;`
3. **Wyświetlanie listy wyzwalaczy zarejestrowanych w systemie** – służy do tego polecenie:

```
SHOW TRIGGERS;
```

   PRZYKŁAD: `SHOW TRIGGERS;`

   Polecenie to ma dodatkowe opcje, które opisane są w pomocy.

4. **Prawa dostępu do wyzwalaczy** – aby inni użytkownicy mogli korzystać z wyzwalaczy, należy nadać mu prawo „**TRIGGER**” (prawo do używania wyzwalaczy).

   Oto wzorzec:

   ```
   GRANT trigger ON baza_danych.obiekt
   TO nazwa_uzytkownika WITH GRANT OPTION;
   ```

   PRZYKŁAD: `GRANT trigger ON i_miasta.* TO ti41;`

   Oprócz tego można użyć także opcji „**WITH GRANT OPTION**”, która umożliwia przekazywanie innym użytkownikom nadanych praw.

   PRZYKŁAD:
   ```
   GRANT trigger ON *.* TO ti41@localhost WITH GRANT OPTION;
   ```

5. **Przykład dziennika zmian tabeli utworzony za pomocą wyzwalaczy** – oto kod przykładowej tabeli, która pełni funkcję dziennika zmian:

   ```
   CREATE TABLE dziennik(id INT AUTO_INCREMENT PRIMARY KEY, opis_zmiany VARCHAR(250));
   ```

   Następnie tworzymy tabelę, którą będziemy śledzić (opisywać w tabeli dziennik):

   ```
   CREATE TABLE tabela(id INT AUTO_INCREMENT PRIMARY KEY, wartosc FLOAT);
   ```

   W ostatnim kroku należy zdefiniować wyzwalacz dla powyższej tabeli (będzie on działał dla wstawianie, aktualizowania i usuwania danych z tabeli):

   ```
   DELIMITER //
   CREATE TRIGGER trigger_insert AFTER INSERT ON tabela
   FOR EACH ROW
   BEGIN
     INSERT INTO dziennik(opis_zmiany)
     VALUES  (  CONCAT( new.id, ": => ", IF(new.wartosc IS NULL, "NULL", new.wartosc) )  );
   END //
   CREATE TRIGGER trigger_update BEFORE UPDATE ON tabela
   FOR EACH ROW
   BEGIN
     IF new.wartosc != old.wartosc OR new.wartosc IS NULL AND old.wartosc IS NOT NULL
        OR new.wartosc IS NOT NULL AND old.wartosc IS NULL
       THEN
         INSERT INTO dziennik(opis_zmiany)
         VALUES   (   CONCAT( old.id, ": ", IF(old.wartosc IS NULL, "NULL", old.wartosc),
                              " -> ", IF(new.wartosc IS NULL, "NULL", new.wartosc) )   );
     END IF;
   END //
   CREATE TRIGGER trigger_delete BEFORE DELETE ON tabela
   FOR EACH ROW
   BEGIN
     INSERT INTO dziennik(opis_zmiany)
     VALUES  (  CONCAT( old.id, ": ", IF(old.wartosc IS NULL, "NULL", old.wartosc), " =>" )  );
   END //
   DELIMITER ;
   ```

   Teraz można wypróbować działanie dziennika zmian:

   ```
   SELECT * FROM dziennik;  -- dziennik jest pusty
   -- Wstawiamy, aktualizujemy i usuwamy rekordy w tabeli „tabela”
   INSERT INTO tabela(wartosc) VALUES (1), (0.5);
   UPDATE tabela SET wartosc = 2.5; DELETE FROM tabela;
   SELECT * FROM dziennik;  -- dziennik został wypełniony danymi o zmianach w tabeli „tabela”
   ```

   Aby przywrócić bazę danych do stanu przed rozpoczęciem przykładu, należy:

   ```
   -- Usunąć wyzwalacze
   DROP TRIGGER trigger_insert;   DROP TRIGGER trigger_update;   DROP TRIGGER trigger_delete;
   DROP TABLE dziennik;   DROP TABLE tabela;  -- usunąć tabele
   ```

Pytania kontrolne

1. Czym jest wyzwalacz? Jak używamy wyzwalaczy? Jakie zalety i ograniczenia wyzwalaczy?
2. Jak wyświetlamy listę zarejestrowanych wyzwalaczy? Jak ustalamy prawa dostępu do wyzwalaczy?

Zadania

1. Wykonaj przykład dziennika zmian z punktu 5.
2. Spraw, żeby ID zmienianego rekordu oraz poprzednia i nowa wartość znajdowały się w osobnych kolumnach (tabela dziennika zmian ma mieć o 3 pola więcej). Nie usuwaj istniejących pól.