

Wyzwalacze — dziennik zmian (audyt)

Czas: 2 × 45 min

praca indywidualna lub w parach

Cel: Nauczenie się tworzenia wyzwalaczy, które automatycznie zapisują historię zmian danych, oraz poznanie ograniczeń wyzwalaczy.

Po zajęciach potrafisz:

- tworzy wyzwalacze BEFORE i AFTER dla INSERT, UPDATE i DELETE
- korzysta z wartości OLD i NEW i porównuje je bezpiecznie dla NULL ($\Leftrightarrow$)
- projektuje tabelę dziennika i zapisuje w niej kto, kiedy i co zmienił
- rozpoznaje ograniczenia wyzwalaczy: błąd 1442 i brak reakcji na kaskadowe usunięcie

Sprzęt i materiały

- komputer w pracowni z XAMPP (MariaDB 10.4) lub MySQL 8.4
- baza sklep zaimportowana z pliku `sklep.sql` (folder Pliki_do_zadan)

Oprogramowanie

- **XAMPP (Apache + MariaDB + phpMyAdmin)** — panel sterowania XAMPP, konsola: przycisk Shell albo `C:\xampp\mysql\bin\mysql.exe -u root`
apachefriends.org/download.html
- **phpMyAdmin** — zakładka SQL do uruchamiania zapytań; wchodzi w skład XAMPP
localhost/phpmyadmin
- **VS Code lub Notepad++** — do pisania pliku `.sql` z zapytaniem
code.visualstudio.com

Bezpieczeństwo

- Zaimportuj świeżą bazę sklep — w zadaniu zmieniasz ceny, klientów i usuwasz zamówienie. Na koniec zaimportuj ją ponownie, żeby kolejne zadania zaczynały od danych wyjściowych.
- Dziennik zmian zawiera dane osobowe (e-maile). W prawdziwym systemie dostęp do tabel audytu ma tylko administrator, a dane przechowuje się tylko tak długo, jak wymagają tego przepisy (RODO).

Przebieg zadania

1 Historia cen

W wyzwalaczu UPDATE dostępne są dwie wersje wiersza: OLD (przed zmianą) i NEW (po zmianie).

```
DELIMITER //
DROP TRIGGER IF EXISTS trg_produkty_cena //
CREATE TRIGGER trg_produkty_cena
AFTER UPDATE ON produkty
FOR EACH ROW
BEGIN
    IF NEW.cena <> OLD.cena THEN
        INSERT INTO historia_cen (id_produkty, stara_cena, nowa_cena, kto, kiedy)
        VALUES (OLD.id_produkty, OLD.cena, NEW.cena, CURRENT_USER(), NOW());
    END IF;
END //
DELIMITER ;
```

1. Zaimportuj `sklep.sql`. Utwórz plik `r2_11_nazwisko.sql`.
2. Z1. Utwórz tabelę `historia_cen` (id AUTO_INCREMENT, id_produkty, stara_cena, nowa_cena, kto VARCHAR(100), kiedy DATETIME) i wyzwalacz `trg_produkty_cena` (kod powyżej). Zmień cenę produktu 1 na 3199.00, a potem tylko jego stan (+1). Ile wierszy jest w historii i dlaczego?
3. Z2. Jednym poleceniem podnieś o 10% ceny akcesoriów (kategoria 3). Ile wierszy dopisał wyzwalacz? Co oznacza FOR EACH ROW?
4. Z3. Wypisz historię zmian z nazwą produktu, starą i nową ceną, zmianą w procentach i użytkownikiem, który zmienił cenę.

2 Audyt tabeli klienci

Jedna tabela dziennika dla trzech operacji. Dane „przed” i „po” zapisujemy jako tekst e-mail; miasto.

```
CREATE TABLE audyt_klienci (
    id INT AUTO_INCREMENT PRIMARY KEY,
    operacja ENUM('INSERT', 'UPDATE', 'DELETE') NOT NULL,
    id_klienta INT UNSIGNED NOT NULL,
    stare VARCHAR(255) NULL,
    nowe VARCHAR(255) NULL,
    kto VARCHAR(100) NOT NULL,
    kiedy DATETIME NOT NULL
);
```

1. Z4. Utwórz tabelę `audyt_klienci` (kod powyżej) i trzy wyzwalacze AFTER: `trg_klienci_ai` (INSERT — tylko nowe), `trg_klienci_au` (UPDATE — stare i nowe) i `trg_klienci_ad` (DELETE — tylko stare). Pomocna funkcja: `CONCAT_WS(';', ' ', NEW.email, NEW.miasto)`.
2. Z5. Utwórz wyzwalacz BEFORE INSERT `trg_klienci_bi`, który przed zapisem zamienia e-mail na małe litery bez spacji na brzegach, a puste miasto (same spacje) na NULL. Dodaj klientkę Ola Nowicka z e-mailem „Ola.Nowicka@Example.PL ” i miastem „ ”. Sprawdź, co zapisało się w tabeli klienci i w dzienniku.
3. Z6. Zmień miasto Oli Nowickiej na Lublin, „zmień” miasto klientki 1 na Kraków (to samo, co już ma), usuń Olę Nowicką i spróbuj usunąć klientkę 1. Wyświetl dziennik. Dlaczego próba usunięcia klientki 1 nie zostawiła wpisu?
4. Z7. Popraw `trg_klienci_au` tak, żeby zapisywał wpis tylko wtedy, gdy e-mail lub miasto naprawdę się zmieniły. Użyj porównania odpornego na NULL: `IF NOT (OLD.email <=> NEW.email AND OLD.miasto <=> NEW.miasto) THEN ...`. Sprawdź: ponownie Kraków dla klientki 1, miasto NULL dla klienta 2, zmiana samego imienia klientki 7.

3 Przegląd i ograniczenia wyzwalaczy

1. Z8. Wyświetl `SHOW TRIGGERS;` oraz z `information_schema.TRIGGERS` kolumny `TRIGGER_NAME`, `ACTION_TIMING`, `EVENT_MANIPULATION`, `EVENT_OBJECT_TABLE`. Znajdź wyzwalacze w phpMyAdmin (tabela → Wyzwalacze).
2. Z9. Utwórz wyzwalacz AFTER INSERT na produkty, który próbuje zaktualizować tabelę produkty (np. `UPDATE produkty SET stan = stan + 0 WHERE id_produktu = NEW.id_produktu`). Dodaj dowolny produkt. Zapisz błąd i czy produkt został dodany. Usuń ten wyzwalacz.
3. Z10. Utwórz tabelę `log_pozycji` (id, id_zamowienia, id_produktu, kiedy) i wyzwalacz AFTER DELETE na pozycje_zamowienia zapisujący usunięte pozycje. Usuń zamówienie 4 (jego pozycje usuwa kaskada ON DELETE CASCADE), a potem bezpośrednio pozycję produktu 7 z zamówienia 1. Ile wpisów jest w log_pozycji?
4. Z11. Usuń wyzwalacz z Z10 (DROP TRIGGER) i sprawdź SHOW TRIGGERS LIKE 'pozycje%'. Zaimportuj ponownie `sklep.sql`.

W sprawozdaniu umieść

- ☐ plik `.sql` z tabelami, wyzwalaczami i testami Z1–Z11
- ☐ zrzuty ekranu historii cen (Z3) i dziennika klientów po Z6 i Z7
- ☐ opis ograniczeń z Z9 i Z10
- ☐ odpowiedzi na pytania kontrolne

Pytania kontrolne

1. Kiedy użyjesz wyzwalacza BEFORE, a kiedy AFTER? Podaj przykład z tego zadania.
2. Które z wartości OLD i NEW są dostępne w wyzwalaczach INSERT, UPDATE i DELETE? Które można zmieniać?
3. Dlaczego w Z7 zwykłe `OLD.miasto <> NEW.miasto` nie wykryje zmiany z „Warszawa” na NULL?
4. Wyzwalacz nie zadziałał przy kaskadowym usunięciu pozycji (Z10). Jak zapewnić pełny audyt usuwanych pozycji?

Kryteria oceny

Kryterium	Punkty
Historia cen — tabela, wyzwalacz i raport (Z1–Z3)	4
Audyt klientów: wyzwalacze AFTER i BEFORE z testami (Z4–Z6)	6
Poprawiony wyzwalacz z porównaniem $\Leftrightarrow$ (Z7)	2
Przegląd i ograniczenia wyzwalaczy (Z8–Z11)	5
Odpowiedzi na pytania kontrolne	3
Razem	20

Sprawozdanie przygotuj na wzorze dołączonym do zadania (plik .docx o tym samym numerze).