Wprowadzenie do Triggerów (wyzwalaczy) w SQL

Trigger (czyli wyzwalacz) to fragment kodu SQL, który **automatycznie wykonuje się po zajściu określonego zdarzenia** w bazie danych, np. dodania, usunięcia lub zmiany rekordu.

Gdzie można używać triggerów?

Triggerów można używać w wielu systemach baz danych, takich jak:

- **Microsoft SQL Server**
- **PostgreSQL**
- **MySQL**
- **Oracle**
- **SQLite**
- **Firebird**
- **Sybase**
- **InterBase**

Rodzaje triggerów (BEFORE vs AFTER)

BEFORE

Trigger wywołuje się **przed wykonaniem operacji** (INSERT/UPDATE/DELETE). Można np. zmienić wartości pól lub zablokować operację.

AFTER

Trigger uruchamia się **po wykonaniu operacji**. Często używany do logowania zmian, kopiowania danych itp.

Rodzaje zdarzeń:

- `BEFORE INSERT` – przed dodaniem nowego rekordu
- `BEFORE UPDATE` – przed modyfikacją
- `BEFORE DELETE` – przed usunięciem
- `AFTER INSERT`, `AFTER UPDATE`, `AFTER DELETE` – po tych operacjach

Składnia triggera (MySQL/PostgreSQL/SQLite styl)

```
CREATE TRIGGER nazwa_triggera
BEFORE INSERT ON nazwa_tabeli
FOR EACH ROW - To kluczowy fragment – oznacza, że trigger zostanie uruchomiony
                  osobno dla każdego rekordu (wiersza), który jest dodawany,
                  modyfikowany lub usuwany.
BEGIN
    -- kod SQL
END;
```

PRZYKŁAD 1: Automatyczna data rejestracji klienta

Tabela:

```
CREATE TABLE Klient (
    id INT PRIMARY KEY,
    imie VARCHAR(50),
    nazwisko VARCHAR(50),
    email VARCHAR(100),
    data_rejestracji DATETIME
);
```

Cel:

Gdy klient zostaje dodany, automatycznie uzupełnij `data_rejestracji`.

Trigger:

```
CREATE TRIGGER TR_DataRejestracji
BEFORE INSERT ON Klient
FOR EACH ROW
BEGIN
    SET NEW.data_rejestracji = NOW();
END;
```

Sprawdzenie działania – INSERT

```
INSERT INTO Klient (id, imie, nazwisko, email)
VALUES (1, 'Ewa', 'Nowak', 'ewa.nowak@example.com');

SELECT * FROM Klient;
```

Wynik:

```
1 | Ewa | Nowak | ewa.nowak@example.com | 2025-05-20 13:40:00
```

PRZYKŁAD 2: Historia zmian ceny produktu

Tabela:

```
CREATE TABLE Produkt (
    id INT PRIMARY KEY,
    nazwa VARCHAR(100),
    cena DECIMAL(10,2),
    ostatnia_aktualizacja DATETIME,
    poprzednia_cena DECIMAL(10,2)
);
```

Cel:

Gdy cena produktu się zmieni, zapisz poprzednią cenę i datę zmiany.

Trigger:

```
CREATE TRIGGER TR_AktualizacjaCeny
BEFORE UPDATE ON Produkt
FOR EACH ROW
BEGIN
    IF NEW.cena != OLD.cena THEN
        SET NEW.poprzednia_cena = OLD.cena;
        SET NEW.ostatnia_aktualizacja = NOW();
    END IF;
END;
```

Sprawdzenie działania – UPDATE

```
UPDATE Produkt SET cena = 99.99 WHERE id = 1;

SELECT * FROM Produkt;
```

Otrzymasz:

```
1 | Produkt ABC | 99.99 | 2025-05-20 13:42:15 | 120.00
```