|  |  |  |  |
| --- | --- | --- | --- |
| Ćwiczenie:  **DDL - TRIGGERS** | Uczelnia/Wydział:  **Akademia Nauk Stosowanych w Nowym Sączu Wydział Nauk Inżynieryjnych** | | |
| Kierunek i grupa:  **2 IS (s) L1/P1** | Przedmiot:  **Bazy Danych – Laboratorium, mgr. Inż. Nikodem Bulanda** | | |
| Data Wykonania:  **10.05.2023 r** | Imię i Nazwisko **Mateusz Bienias** | Data oddania:  **17.05.2023 r.** | Ocena: |

1. **„Stwórz wyzwalacz, który podczas każdorazowego wprowadzenia wypłaty do tabeli salaries zwiększy sumaryczną wartość zmiennej @sum, przechowującej wartość sumaryczną wypłaconych w tej sesji uposażeń. Sprawdzenia dokonaj dla pracownika o numerze 9998. Jeśli nie istniej stwórz go i zadeklaruj przynajmniej 4 wypłaty po czym sprawdź stan zmiennej @sum.”**

// Tworzymy użytkownika z emp\_no = 9998 i wyświetlamy

INSERT INTO employees VALUES

(9998, '2001-09-05', 'Jan', 'Kowalski', 'M', '2016-04-07');

SELECT distinct \* FROM employees LIMIT 20;

  
  
// Tworzymy zmienną SUM I trigger

SET @sum = 0;  
CREATE TRIGGER salary\_sum BEFORE INSERT ON salaries

FOR EACH ROW SET @sum = @sum + NEW.salary;

// Następnie wrzucamy np: 5 rekordów do tabeli salaries dla danego pracownika

INSERT INTO salaries VALUES

(9998, 39000, '2016-04-07', '2017-04-07'),

(9998, 42000, '2017-04-07', '2018-04-07'),

(9998, 42000, '2018-04-07', '2019-04-07'),

(9998, 45000, '2019-04-07', '2020-04-07'),

(9998, 45000, '2020-04-07', '2021-04-07');

// Trigger powinien zadziałać i zliczyć nowo dodane wypłaty pracownika

SELECT @sum AS 'Suma wyplaconych uposazen';



1. **„Wyświetl i porównaj listę triggerów w twoim systemie SZDB.”**

// Listing z zadania:

|'salary\_sum '|'INSERT '|'salaries '|'SET @sum = @sum + NEW . salary '|

'BEFORE '|'2023 -04 -29 02:41:26.880000 '|'ONLY\_FULL\_GROUP\_BY ,

STRICT\_TRANS\_TABLES , NO\_ZERO\_IN\_DATE , NO\_ZERO\_DATE ,

ERROR\_FOR\_DIVISION\_BY\_ZERO , NO\_ENGINE\_SUBSTITUTION '|' root@localhost

'|'utf8mb4 '|' utf8mb4\_0900\_ai\_ci '|' utf8mb4\_0900\_ai\_ci '|

// Polecenie do wyświetlania triggerów

SHOW TRIGGERS;

// Listing po wykonaniu zapytania w moim SZDB:

 salary\_sum | INSERT | salaries | SET @sum = @sum + NEW.salary |

 BEFORE | 2023-05-10 11:20:31.49 |ONLY\_FULL\_GROUP\_BY,

 STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,

 ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION | root@localhost

 | utf8mb4 | utf8mb4\_0900\_ai\_ci | utf8mb4\_0900\_ai\_ci

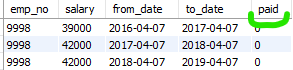
// W zasadzie po za datą listingi niczym się nie różnią

1. **„Rozbuduj tabele salaries o kolumnę paid, przyjmującą wartości boolowskie. Kolumna ta będzie przechowywać flagę potwierdzającą przekazanie pieniędzy pracownikowi, na początek ustaw wszystkie wartości na false. Następnie dokonaj zadeklarowania triggera który będzie przeliczał i aktualizował stan zmiennych @wyplacono i @do\_wyplaty (gdzie @do\_wypłaty będzie stanowić sumę aktualnych zobowiązań, zaś @wyplacono sumę wypłaconych w tej sesji wypłat).”**

// Dodajemy kolumnę „paid”

ALTER TABLE salaries ADD COLUMN

(paid BOOLEAN NOT NULL DEFAULT FALSE);



// Deklarujemy zmienne ‘do\_wyplaty’ I ‘wyplacono’

SET @do\_wyplaty =   
(SELECT **SUM**(salary) FROM salaries WHERE emp\_no = 9998);

SET @wyplacono = 0;

// Sprawdzamy jakie wartości znajdują się początkowo w zmiennych

SELECT @do\_wyplaty; SELECT @wyplacono;

// **Definiujemy trigger dla operacji INSERT na tabeli salaries**

CREATE TRIGGER salary\_insert BEFORE INSERT ON salaries

FOR EACH ROW

SET

  @do\_wyplaty = @do\_wyplaty + IF(NEW.paid=0, NEW.salary, 0),

  @wyplacono = @wyplacono + IF(NEW.paid=1, NEW.salary, 0);

// W tym triggerze sprawdzamy czy nowy wpis zawiera pensję którą już wypłacono czyli gdy paid=1, wtedy dodajemy wartość salary do ‘wyplacono’ w przeciwnym wypadku gdy paid=0 to wartość salary dodajemy do ‘do\_wyplaty’.  
  
// Przykład dla INSERT, gdzie paid = false czyli 0

INSERT INTO salaries VALUES

(9998, 58000, '2021-04-07', '2022-04-07', false);

// Jak widać podniesiono wartość ‘do\_wyplaty’  
 

// Przykład dla INSERT, gdzie paid = true czyli 1

INSERT INTO salaries VALUES

(9998, 58500, '2022-04-07', '2023-04-07', true);

// Jak widać podniesiono wartość ‘wyplacono’

// **Definiujemy trigger dla operacji UPDATE na tabeli salaries**

CREATE TRIGGER salary\_update BEFORE UPDATE ON salaries

FOR EACH ROW

  SET @do\_wyplaty = @do\_wyplaty - IF(OLD.paid=0, OLD.salary, 0) + IF(NEW.paid=0, NEW.salary, 0),

  @wyplacono = @wyplacono - IF(OLD.paid=1, OLD.salary, 0) + IF(NEW.paid=1, NEW.salary, 0);

// W tym triggerze najpierw sprawdzamy sytuację czy poprzednio wartość paid = 0  
a po UPDATE zmieniła się na paid = 1, jeśli tak to oznacza że dana należność została wypłacona pracownikowi, więc musimy odjąć tą kwotę, czyli ‘salary’ od ‘do\_wyplaty’ i dodać do ‘wyplacono’. Kolejna sytuacja to gdy np.: księgowy się pomylił i chce cofnąć naliczoną wypłatę więc w kolejnym UPDATE zmienia paid z *true* na *false* to odejmujemy starą pensję od ‘wyplacono’ i dodajemy nową do ‘do\_wyplaty’.

// Przykład dla UPDATE, gdzie paid = było false a jest true

UPDATE salaries

SET paid = 1

WHERE emp\_no = 9998 AND from\_date = '2017-04-07'   
AND to\_date = '2018-04-07';

// Jak widać wzrosło ‘wyplacono’ a zmalało ‘do\_wyplaty’

// Przykład dla UPDATE, gdzie paid = było true a jest false

UPDATE salaries

SET paid = 0

WHERE emp\_no = 9998 AND from\_date = '2017-04-07'   
AND to\_date = '2018-04-07';

// Jak widać podniesiono wartości wróciły do tych z przez poprzedniego UPDATE

// Można też aktualizować kwoty należności wypłaconych i tych do\_wyplaty, wtedy trigger nie będzie przenosił wartości z jednej zmiennej do drugiej tylko zwiększy ich wartość.

UPDATE salaries

SET salary = 39999

WHERE emp\_no = 9998 AND from\_date = '2021-04-07'   
AND to\_date = '2022-04-07' AND paid = 0;

// Jak widać wzrosła kwota do wypłaty a w wypłacono nic się nie zmieniło

1. **„Zbuduj prosty system logujący, składający się z tabeli i trzech triggerów. Zapisujących informację na temat zmian dokonanych w obszarze tabeli salaries. Pamiętaj należy obsłużyć INSERT, UPDATE   
   i DELETE.”**

// Tworzymy tabelę przechowującą logi

CREATE TABLE **salaries\_log** (

  id BIGINT PRIMARY KEY AUTO\_INCREMENT,

  by\_user **VARCHAR** (255) NOT NULL,

  timestamp\_log TIMESTAMP NOT NULL,

  action\_type **VARCHAR** (255) NOT NULL,

  log\_details TEXT);

// Tworzymy trigger dla operacji INSERT

DELIMITER |

CREATE TRIGGER simple\_logger\_insert AFTER INSERT ON salaries

FOR EACH ROW

BEGIN

  INSERT INTO **salaries\_log** (by\_user, timestamp\_log, action\_type, log\_details)

  VALUES (**CURRENT\_USER**(), **NOW**(), 'INSERT',

**CONCAT**('Dodano rekord. Pensja: ', NEW.salary, ' za okres od: ',

  NEW.from\_date, ' do ', NEW.to\_date, '. Status wypłaty: ', NEW.paid));

END;|

// Przykład gdy dokonamy inserta na salaries

INSERT INTO salaries VALUES

(9998, 58500, '2023-04-07', '2024-04-07', false);



// Tworzymy trigger dla operacji UPDATE

DELIMITER |

CREATE TRIGGER simple\_logger\_update AFTER UPDATE ON salaries

FOR EACH ROW

BEGIN

  INSERT INTO **salaries\_log** (by\_user, timestamp\_log, action\_type, log\_details)

  VALUES (**CURRENT\_USER**(), **NOW**(), 'UPDATE',

**CONCAT**('Zaktualizowano: ', NEW.salary,

        ' za okres od: ', NEW.from\_date,

        ' do ', NEW.to\_date,

        ' dla pracownika ', NEW.emp\_no,

        ' stare wartosci: ', OLD.salary,

        '\_', OLD.from\_date,

        '\_', OLD.to\_date,

        '\_', OLD.emp\_no));

END;|

// Przykład gdy dokonamy update na salaries

UPDATE salaries

SET salary = 42999

WHERE emp\_no = 9998 AND from\_date = '2021-04-07'   
AND to\_date = '2022-04-07';



// Tworzymy trigger dla operacji DELETE

DELIMITER |

CREATE TRIGGER simple\_logger\_delete AFTER DELETE ON salaries

FOR EACH ROW

BEGIN

  INSERT INTO **salaries\_log** (by\_user, timestamp\_log, action\_type, log\_details)

  VALUES (**CURRENT\_USER**(), **NOW**(), 'DELETE',

**CONCAT**('Usunięto ', OLD.salary,

  ' za okres od: ', OLD.from\_date,

  ' do ', OLD.to\_date,

  '. Wyplacono: ', OLD.paid,

  ' dla pracownika: ', OLD.emp\_no));

END;|

// Przykład gdy dokonamy DELETE na salaries

DELETE FROM salaries

WHERE emp\_no = 9998 AND from\_date = '2020-04-07' AND paid = 0;

