

МІНІСТЕРСТВО ОСВІТИ І НАУКИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»



ЛАБОРАТОРНА РОБОТА № 5

РЕАЛІЗАЦІЯ BACK-END ПРОЕКТУ

Виконав: ст. гр. ІР-24,

Дзень С. А.

Прийняла: к. т. н., стар. вик.

Лагун І. І.

Львів-2024

Завдання:

На базі попередньої роботи (back-end with Flask) слід написати для існуючої БД ряд програмних конструкцій (тригери, процедури, функції, курсори). Для збережуваних процедур забезпечити їхній виклик за допомогою контролерів бекенду.

1. Додати до БД 1 додаткову довільну таблицю і зв'язати з іншою існуючою таблицею зв'язком 1:М. Однак для забезпечення цілісності значень використати тригери замість фізичного зовнішнього ключа.
2. Збережені процедури:
 - a. Забезпечити параметризовану вставку нових значень у довільну таблицю.
 - b. Забезпечити реалізацію зв'язку М:М між 2ма таблицями, тобто вставити в стикувальну таблицю відповідну стрічку за реально-існуючими значеннями (напр. surname, name) в цих основних таблицях.
 - c. Створити пакет, який вставляє 10 стрічок у довільну таблицю БД у форматі < Noname+№> , наприклад: Noname5, Noname6, Noname7 і т.д.
 - d. Написати користувацьку функцію, яка буде шукати Max, Min, Sum чи Avg для стовпця довільної таблиці у БД. Написати процедуру, яка буде у SELECT викликати цю функцію.
 - e. Написати 1 процедуру із курсором для виконання однієї із наступних задач:
 - i. Використовуючи курсор, забезпечити динамічне створення 2х таблиць з іменами що містять штамп часу, структура таблиць ідентична будь-якій структурі таблиці БД. Після чого випадковим чином пострічково скопіювати стрічки із батьківської таблиці або в одну, або в іншу додаткові таблиці. Повторний запуск процедури знову створює нові аналогічні таблиці, в яких випадковим чином знову будуть розкинуті дані з батьківської таблиці.
 - ii. Використовуючи курсор, забезпечити динамічне створення таблиць з назвами+штамп часу, взятими зі стовпця з довільної таблиці БД, з випадковою кількістю стовпців (від 1 до 9). Імена та тип стовпців довільні.
 - iii. Використовуючи курсор, забезпечити динамічне створення баз даних з іменами, взятими зі стовпця з довільної таблиці поточної БД, з випадковою кількістю таблиць для кожної БД (від 1 до 9). Структура таблиць довільна. Імена таблиць відповідають імені БД з порядковим номером від 1 до 9.
3. Написати 3 довільні тригери для таблиць поточної БД, як приклад можна взяти наступні:
 - a. Значення певного стовпця не може закінчувати двома нулями
 - b. Заборонити будь-яку модифікацію даних в таблиці
 - c. Заборонити видалення стрічок з таблиці
 - d. Забезпечити мінімальну кардинальність 6 стрічок для певної таблиці БД
 - e. Забезпечити кардинальність (min=2, max=6) стрічок для певної таблиці БД
 - f. Створити таблицю-журнал, в якій вести логи зі штампом часу при видаленні даних для певної таблиці
 - g. Створити таблицю-журнал, в якій вести логи зі штампом часу при модифікації даних для таблиці
 - h. Для певного стовпця забезпечити формат вводу:
2 довільні букви, окрім M і R + '-' + 3 цифри + '-' + 2цифри
 - i. Для певного стовпця забезпечити формат вводу:
1 буква: A, M чи Z + 5 цифр + 2 довільні букви
 - j. Для певного стовпця допускається ввід лише таких імен: 'Svitlana', 'Petro', 'Olha', 'Taras'.
 - k. Для певного стовпця забезпечити формат вводу:
перша літера у значенні повинна відповідати першій літері значення сусіднього поля у рядку

Посилання на гітхаб:

<https://github.com/sergiyclas/db-lab-4-5.git>

1. Додати до БД 1 додаткову довільну таблицю і зв'язати з іншою існуючою таблицею зв'язком 1:М. Однак для забезпечення цілісності значень використати тригери замість фізичного зовнішнього ключа.

- Створення таблиці logs. Таблиця logs містить сповіщення, пов'язані з transactions, і забезпечує зв'язок через логіку тригерів.

```
CREATE TABLE logs (  
  log_id INT AUTO_INCREMENT PRIMARY KEY,  
  transaction_id INT NOT NULL,  
  log_action VARCHAR(100),  
  log_time TIMESTAMP DEFAULT CURRENT_TIMESTAMP  
);
```

```
DELIMITER $$
```

```
CREATE TRIGGER after_transaction_insert
```

```
AFTER INSERT ON transactions
```

```
FOR EACH ROW
```

```
BEGIN
```

```
  INSERT INTO logs (transaction_id, log_action)
```

```
  VALUES (NEW.transaction_id, 'Transaction inserted');
```

```
  INSERT INTO logs (transaction_id, log_action)
```

```
  VALUES (NEW.transaction_id, 'Transaction reviewed');
```

```
  INSERT INTO logs (transaction_id, log_action)
```

```
  VALUES (NEW.transaction_id, 'Transaction approved');
```

```
END$$
```

```
DELIMITER ;
```

Зв'язку нема:

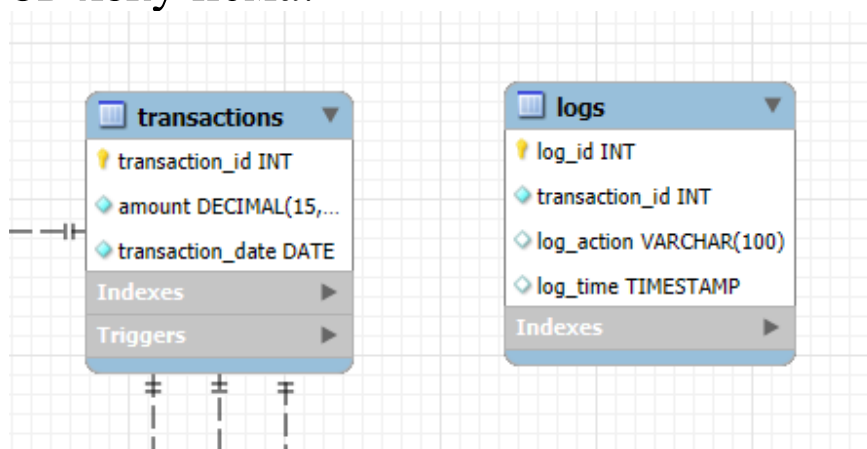


Таблица logs:

GET

http://127.0.0.1:5000/logs

Send

Params

Authorization

Headers (6)

Body

Scripts

Settings




Cookies

Body

Cookies

Headers (6)

Test Results

200 OK · 6 ms · 5.19 KB ·   



Pretty

Raw

Preview

Visualize

JSON

1

[

2

{

3

"log_action": "Transaction inserted",

4

"log_id": 1,

5

"log_time": "Thu, 28 Nov 2024 00:51:00 GMT",

6

"transaction_id": 1

7

},

8

{

9

"log_action": "Transaction reviewed",

10

"log_id": 2,

11

"log_time": "Thu, 28 Nov 2024 00:51:00 GMT",

12

"transaction_id": 1

13

},

14

{

15

"log_action": "Transaction approved",

16

"log_id": 3,

17

"log_time": "Thu, 28 Nov 2024 00:51:00 GMT",

18

"transaction_id": 1

19

},

20

{

21

"log_action": "Transaction inserted",

22

"log_id": 4,

23

"log_time": "Thu, 28 Nov 2024 00:51:00 GMT",

24

"transaction_id": 2

25

},

26

{

Таблица transactions:

GET

http://127.0.0.1:5000/transactions

Send

Params

Authorization

Headers (6)

Body

Scripts

Settings




Cookies

Body

Cookies

Headers (6)

Test Results

200 OK · 6 ms · 1.54 KB ·   




Pretty

Raw

Preview

Visualize

JSON

1

[

2

{

3

"amount": "150.75",

4

"transaction_date": "Wed, 10 Jan 2024 00:00:00 GMT",

5

"transaction_id": 1

6

},

7

{

8

"amount": "200.00",

9

"transaction_date": "Thu, 15 Feb 2024 00:00:00 GMT",

10

"transaction_id": 2

11

},

12

{

13

"amount": "500.50",

14

"transaction_date": "Wed, 20 Mar 2024 00:00:00 GMT",

15

"transaction_id": 3

16

},

17

{

18

"amount": "320.25",

19

"transaction_date": "Thu, 25 Apr 2024 00:00:00 GMT",

20

"transaction_id": 4

21

},

22

{

23

"amount": "410.00",

24

"transaction_date": "Thu, 30 May 2024 00:00:00 GMT",

Зв'язок 1:М:

- виводяться усі логи до вказаної транзакції

GET

http://127.0.0.1:5000/logs/2/transactions

Send

Params

Authorization

Headers (6)

Body

Scripts

Settings

Cookies

Body

Cookies

Headers (6)

Test Results

↺

200 OK

6 ms

624 B

🌐

📄

⋮

Pretty

Raw

Preview

Visualize

JSON

⌵

🔗

📄

🔍

```
1  [
2    {
3      "log_action": "Transaction inserted",
4      "log_id": 4,
5      "log_time": "Thu, 28 Nov 2024 00:51:00 GMT",
6      "transaction_id": 2
7    },
8    {
9      "log_action": "Transaction reviewed",
10     "log_id": 5,
11     "log_time": "Thu, 28 Nov 2024 00:51:00 GMT",
12     "transaction_id": 2
13   },
14   {
15     "log_action": "Transaction approved",
16     "log_id": 6,
17     "log_time": "Thu, 28 Nov 2024 00:51:00 GMT",
18     "transaction_id": 2
19   }
20 ]
```

2. а. Параметризована вставка у довільну таблицю

DELIMITER \$\$

```
CREATE PROCEDURE insert_into_table(  
  IN table_name VARCHAR(255),  
  IN column_list VARCHAR(255),  
  IN value_list VARCHAR(255)  
)  
BEGIN  
  SET @sql = CONCAT('INSERT INTO ', table_name, ' (', column_list, '  
VALUES (', value_list, ')');  
  SELECT @sql AS generated_sql;  
  PREPARE stmt FROM @sql;  
  EXECUTE stmt;  
  DEALLOCATE PREPARE stmt;  
END $$  
DELIMITER ;
```

Проста вставка у таблицю через процедуру
- вставка у таблицю transactions

The image shows two screenshots of a REST client interface. The top screenshot shows a POST request to `http://localhost:5000/insert` with a JSON body containing transaction details. The response is a 201 status code. The bottom screenshot shows a GET request to `http://127.0.0.1:5000/transactions/13` which returns a 200 status code with the transaction details.

Top Screenshot: POST Request

Method: POST
URL: `http://localhost:5000/insert`
Body (JSON):

```
{  "table_name": "transactions",  "column_list": "amount, transaction_date",  "value_list": "'500.50', '2024-12-01'"}
```


Response: 201 CREATED (4 ms, 224 B)

Bottom Screenshot: GET Request

Method: GET
URL: `http://127.0.0.1:5000/transactions/13`
Response: 200 OK (6 ms, 302 B)
Body (JSON):

```
{  "amount": "500.50",  "transaction_date": "Sun, 01 Dec 2024 00:00:00 GMT",  "transaction_id": 13}
```

b. Забезпечити реалізацію зв'язку М:М між 2ма таблицями, тобто вставити в стикувальну таблицю відповідну стрічку за реально-існуючими значеннями (напр. surname, name) в цих основних таблицях

- вставка для зв'язку М:М між таблицями accounts, transactions та стикувальною таблицею TransactionsAccounts

Процедура для зв'язку багато до багатьох для таблиць accounts, transactions і їх стикувальної таблиці TransactionsAccounts.

DELIMITER \$\$

```
CREATE PROCEDURE insert_into_relation_accounts_transactions (  
  IN value1 VARCHAR(50),  
  IN value2 VARCHAR(50)  
)
```

BEGIN

```
  SET @query = CONCAT('INSERT INTO TransactionsAccounts (account_id,  
  transaction_id) ',
```

```
    'SELECT (SELECT account_id FROM accounts WHERE  
  account_number = '', value1, ''), ',
```

```
    '(SELECT transaction_id FROM transactions WHERE  
  transaction_id = '', value2, ''))';
```

```
  PREPARE stmt FROM @query;
```

```
  EXECUTE stmt;
```

```
  DEALLOCATE PREPARE stmt;
```

END;

\$\$

DELIMITER ;

Ця процедура вставляє зв'язок між конкретними акаунтами з таблиці accounts та транзакціями з таблиці transactions в таблицю TransactionsAccounts за їхнім значенням.

POST http://localhost:5000/acc/tran

Params Authorization Headers (8) **Body** Scripts Settings Cookies

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL JSON Beautify

```
1 {
2   "value1": "ACC10001",
3   "value2": "9"
4 }
```

Body Cookies Headers (6) Test Results 201 CREATED • 10 ms • 253 B

Pretty Raw Preview Visualize JSON

```
1 {
2   "message": "Transaction linked successfully"
3 }
```

GET http://localhost:5000/customers/1/accounts

Params Authorization Headers (6) **Body** Scripts Settings Cookies

☒ none ☐ form-data ☐ x-www-form-urlencoded ☐ raw ☐ binary ☐ GraphQL

This request does not have a body

Body Cookies Headers (6) Test Results 200 OK • 6 ms • 537 B

Pretty Raw Preview Visualize JSON

```
1 [
2   {
3     "account_id": 1,
4     "account_number": "ACC10001",
5     "balance": "5000.00",
6     "customer_id": 1
7   },
8   {
9     "account_id": 2,
10    "account_number": "ACC20002",
11    "balance": "1000.00",
12    "customer_id": 1
13  }
14 ]
```

GET http://127.0.0.1:5000/accounts/1/transactions

Params Authorization Headers (6) **Body** Scripts Settings Cookies

Body Cookies Headers (6) Test Results 200 OK • 7 ms • 736 B

Pretty Raw Preview Visualize JSON

```
1 [
2   {
3     "amount": "200.00",
4     "fee_amount": "10.00",
5     "fee_date": "Thu, 15 Feb 2024 00:00:00 GMT",
6     "status": "Pending",
7     "status_date": "Thu, 15 Feb 2024 00:00:00 GMT",
8     "transaction_date": "Thu, 15 Feb 2024 00:00:00 GMT",
9     "transaction_id": 2
10  },
11  {
12    "amount": "380.25",
13    "fee_amount": "4.25",
14    "fee_date": "Wed, 25 Sep 2024 00:00:00 GMT",
15    "status": "Completed",
16    "status_date": "Wed, 25 Sep 2024 00:00:00 GMT",
17    "transaction_date": "Wed, 25 Sep 2024 00:00:00 GMT",
18    "transaction_id": 9
19  }
20 ]
```


с. Створити пакет, який вставляє 10 стрічок у довільну таблицю БД у форматі <Noname+№> , наприклад: Noname5, Noname6, Noname7 і т.д.

- Процедура для вставки 10 записів у таблицю customers, з іменами Noname1, Noname2...

DELIMITER \$\$

```
CREATE PROCEDURE insert_multiple_rows_customers (  
    IN start_num INT,  
    IN end_num INT  
)  
BEGIN  
    DECLARE i INT;  
    SET i = start_num;  
    WHILE i <= end_num DO  
        SET @query = CONCAT('INSERT INTO customers (customer_name)  
VALUES ('Noname', i, ''));  
        PREPARE stmt FROM @query;  
        EXECUTE stmt;  
        DEALLOCATE PREPARE stmt;  
        SET i = i + 1;  
    END WHILE;  
END$$
```

DELIMITER ;

Процедура вставляє записи у таблицю customers з іменами Noname1, Noname2 і так далі. Цей процес повторюється, поки не буде вставлено end - start записів.

POST http://127.0.0.1:5000/customers/noname Send

Params Authorization Headers (8) Body Scripts Settings Cookies

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL JSON

```
1 {
2   "start": 1,
3   "end": 10
4 }
```

Body Cookies Headers (6) Test Results 201 CREATED • 13 ms • 224 B •

Pretty Raw Preview Visualize JSON

```
1 {
2   "Success": true
3 }
```

GET http://127.0.0.1:5000/customers Send

Params Authorization Headers (6) Body Scripts Settings Cookies

Body Cookies Headers (6) Test Results 200 OK • 6 ms • 2.72 KB •

Pretty Raw Preview Visualize JSON

```
68 {
69   "customer_id": 12,
70   "customer_name": "Customer 12",
71   "email": "customer12@mail.com",
72   "phone": "+380992345678"
73 },
74 {
75   "customer_id": 13,
76   "customer_name": "Noname1",
77   "email": null,
78   "phone": null
79 },
80 {
81   "customer_id": 14,
82   "customer_name": "Noname2",
83   "email": null,
84   "phone": null
85 },
86 {
87   "customer_id": 15,
88   "customer_name": "Noname3",
89   "email": null,
90   "phone": null
91 },
92 {
93   "customer_id": 16
```

d. Написати користувацьку функцію, яка буде шукати Max, Min, Sum чи Avg для стовпця довільної таблиці у БД. Написати процедуру, яка буде у SELECT викликати цю функцію.

- Функція та процедура для таблиці transactions

1) Функція для обчислення таких значень, як MAX, MIN, SUM, AVG для стовпця таблиці transactions.

DELIMITER //

```
CREATE FUNCTION calculate_column_transactions (  
  IN column_name VARCHAR(50),  
  IN operation VARCHAR(10)  
) RETURNS DECIMAL(15,2)  
BEGIN  
  SET @query = CONCAT('SELECT ', operation, '(', column_name, ') FROM  
transactions');  
  PREPARE stmt FROM @query;  
  EXECUTE stmt INTO @result;  
  DEALLOCATE PREPARE stmt;  
  RETURN @result;  
END;
```

DELIMITER ;

Функція виконує операцію (MAX, MIN, SUM, AVG) для конкретного стовпця таблиці transactions і повертає результат.

2) Процедура для виклику функції з таблиці transactions

DELIMITER //

```
CREATE PROCEDURE select_with_function_transactions (  
  IN column_name VARCHAR(50),  
  IN operation VARCHAR(10)  
)  
BEGIN  
  SET @query = CONCAT('SELECT calculate_column_transactions(''  
column_name, '', '', operation, '')');  
  PREPARE stmt FROM @query;  
  EXECUTE stmt;  
  DEALLOCATE PREPARE stmt;  
END;
```

DELIMITER ;

Процедура викликає функцію `calculate_column_transactions`, передаючи параметри для обчислення необхідної операції для стовпця таблиці `transactions`.

The screenshot displays a REST client interface. The top section shows a POST request to `http://127.0.0.1:5000/transactions/count` with a JSON body:

```
1 {
2   "column_name": "amount",
3   "operation": "MAX"
4 }
```

The bottom section shows the response, which is a JSON array containing a single object:

```
1 [
2   {
3     "result": "650.75"
4   }
5 ]
```

Below the REST client, a SQL query is shown:

```
3 • SELECT MAX(amount), MIN(amount), AVG(amount), SUM(amount) FROM online_banking.transactions
```

The query results are displayed in a table with the following data:

	MAX(amount)	MIN(amount)	AVG(amount)	SUM(amount)
▶	650.75	150.75	393.687500	4724.25

е. Написати 1 процедуру із курсором для виконання однієї із наступних задач:

і. Використовуючи курсор, забезпечити динамічне створення 2х таблиць з іменами що містять штамп часу, структура таблиць ідентична будь-якій структурі таблиці БД. Після чого випадковим чином пострічково скопіювати стрічки із батьківської таблиці або в одну, або в іншу додаткові таблиці. Повторний запуск процедури знову створює нові аналогічні таблиці, в яких випадковим чином знову будуть розкинуті дані з батьківської таблиці.

DELIMITER //

CREATE PROCEDURE `CreateRandomTransactionTablesAndCopyData`()

BEGIN

DECLARE done INT DEFAULT FALSE;

DECLARE transId INT;

DECLARE transAmount DECIMAL(15, 2);

DECLARE transDate DATE;

**DECLARE transaction_cursor CURSOR FOR SELECT transaction_id,
amount, transaction_date FROM transactions;**

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

**SET @table1_name = CONCAT('transaction_data_',
DATE_FORMAT(NOW(), '%Y%m%d_%H%i%s'));**

**SET @table2_name = CONCAT('transaction_data_',
DATE_FORMAT(NOW(), '%Y%m%d_%H%i%s'), '_copy');**

**SET @create_table1_sql = CONCAT('CREATE TABLE ', @table1_name, ' (
transaction_id INT NOT NULL,
amount DECIMAL(15, 2) NOT NULL,
transaction_date DATE NOT NULL,
PRIMARY KEY (transaction_id)
) ENGINE=InnoDB;');**

**SET @create_table2_sql = CONCAT('CREATE TABLE ', @table2_name, ' (
transaction_id INT NOT NULL,
amount DECIMAL(15, 2) NOT NULL,
transaction_date DATE NOT NULL,
PRIMARY KEY (transaction_id)
) ENGINE=InnoDB;');**

**PREPARE stmt1 FROM @create_table1_sql;
EXECUTE stmt1;
DEALLOCATE PREPARE stmt1;**

```
PREPARE stmt2 FROM @create_table2_sql;
EXECUTE stmt2;
DEALLOCATE PREPARE stmt2;

OPEN transaction_cursor;

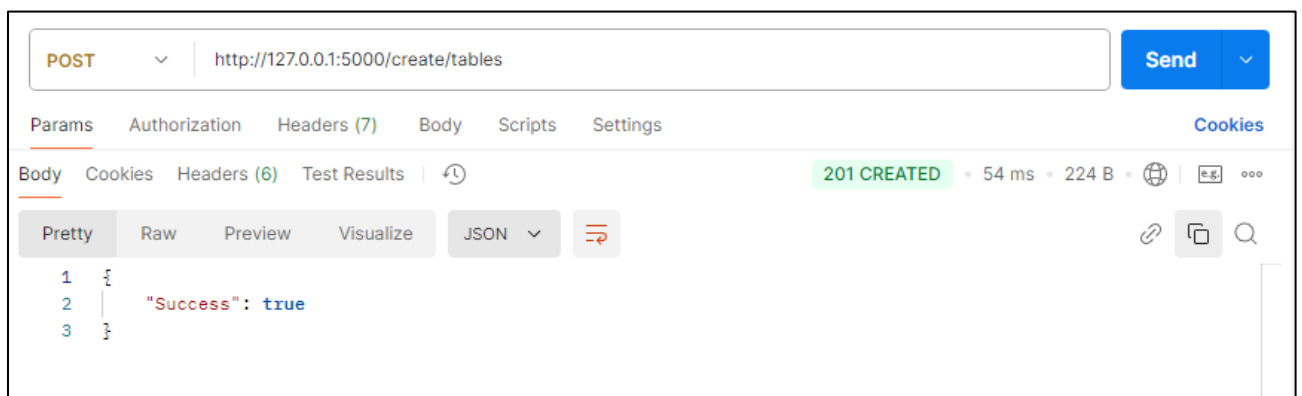
read_loop: LOOP
    FETCH transaction_cursor INTO transId, transAmount, transDate;

    IF done THEN
        LEAVE read_loop;
    END IF;

    IF (RAND() < 0.5) THEN
        SET @insert_sql = CONCAT('INSERT INTO ', @table1_name,
            ' (transaction_id, amount, transaction_date) VALUES (',
            transId, ', ', transAmount, ', ', QUOTE(transDate), ');');
    ELSE
        SET @insert_sql = CONCAT('INSERT INTO ', @table2_name,
            ' (transaction_id, amount, transaction_date) VALUES (',
            transId, ', ', transAmount, ', ', QUOTE(transDate), ');');
    END IF;

    PREPARE stmt3 FROM @insert_sql;
    EXECUTE stmt3;
    DEALLOCATE PREPARE stmt3;
END LOOP;

CLOSE transaction_cursor;
END //
DELIMITER ;
```



Підтвердження створення таблиць та їх заповнення:

SCHEMAS

Filter objects

labor_sql

online_banking

Tables

accounts

authorizations

cards

customers

fees

logs

needs

payment_templates

status_transactions

transaction_data_20241128_034024

transaction_data_20241128_034024_copy

transactions

transactionsaccounts

Views

Stored Procedures

Functions

restaurant_delivery

sys

1 • `SELECT * FROM online_banking.transaction_data_20241128_034024;`

Result Grid

Filter Rows:

Edit:

Export/Import:

Wrap Cell Content:

	transaction_id	amount	transaction_date
▶	1	150.75	2024-01-10
	2	200.00	2024-02-15
	5	410.00	2024-05-30
	6	650.75	2024-06-10
	9	380.25	2024-09-25
	11	310.75	2024-11-10
*	NULL	NULL	NULL

1 • `SELECT * FROM online_banking.transaction_data_20241128_034024_copy;`

Result Grid

Filter Rows:

Edit:

Export/Import:

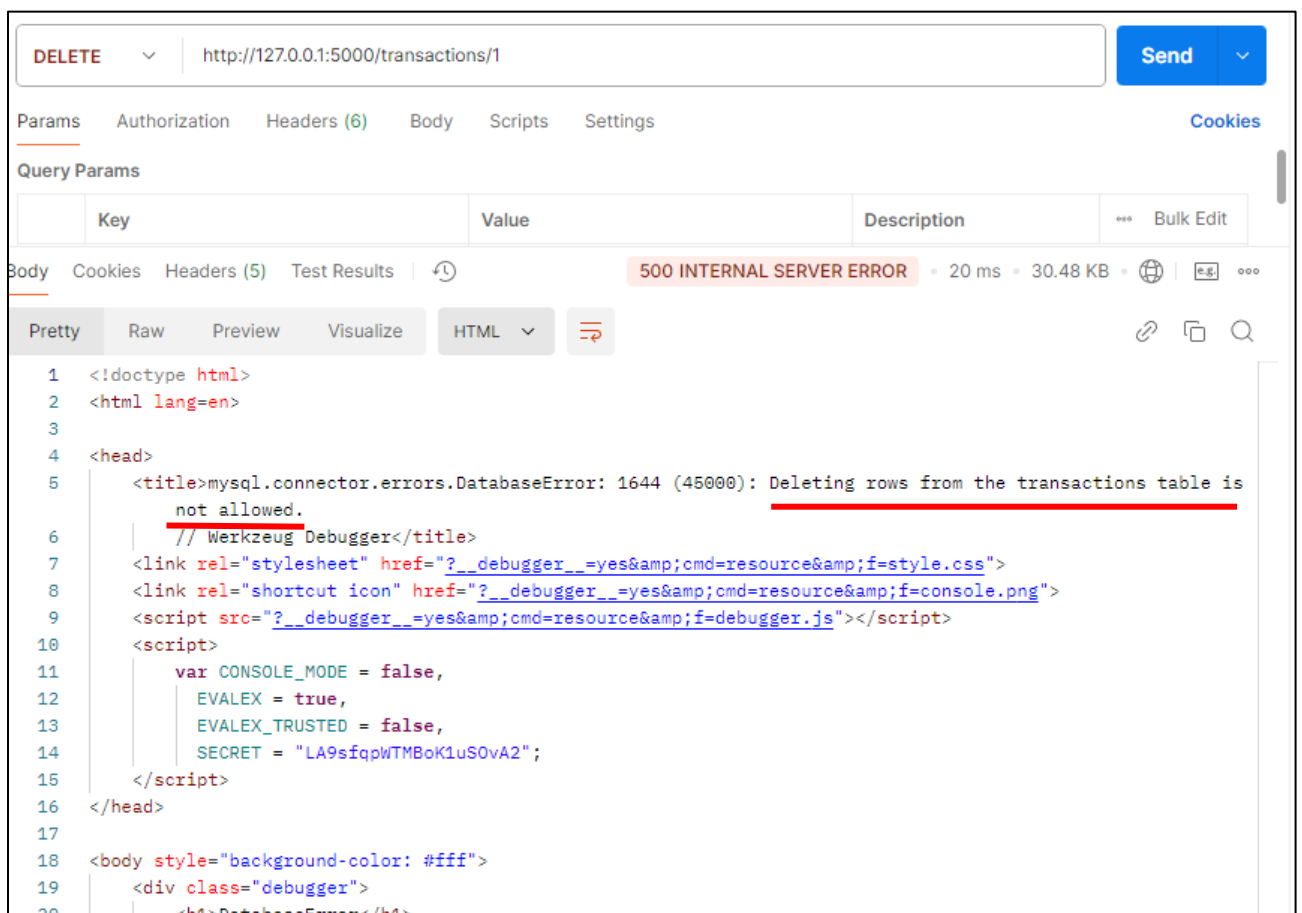
Wrap Cell Content:

	transaction_id	amount	transaction_date
▶	3	500.50	2024-03-20
	4	320.25	2024-04-25
	7	220.00	2024-07-15
	8	530.50	2024-08-20
	10	490.00	2024-10-30
	12	560.50	2024-12-15
*	NULL	NULL	NULL

3. Написати 3 довільні тригери для таблиць поточної БД, як приклад можна взяти наступні:

1) Заборонити видалення рядків із таблиці **transactions**

```
DELIMITER //
CREATE TRIGGER prevent_delete_transactions
BEFORE DELETE ON transactions
FOR EACH ROW
BEGIN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'Deleting rows from the transactions table is not
allowed.';
END;
//
DELIMITER ;
```



2) Забезпечити, щоб значення стовпця (**email**) у таблиці **customers** не закінчувалося двома нулями

DELIMITER //

CREATE TRIGGER prevent_invalid_email

BEFORE INSERT ON customers

FOR EACH ROW

BEGIN

IF NEW.email LIKE '%00' THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE_TEXT = 'Email cannot end with two zeros.';

END IF;

END;

//

DELIMITER ;

POST http://127.0.0.1:5000/customers

Params Authorization Headers (8) Body ● Scripts Settings Cookies Beautify

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   "customer_name": "IDUNNO",
3   "email": "i.dunno@gmail.com00",
4   "phone": "+380999999999"
5 }
```

body Cookies Headers (5) Test Results 500 INTERNAL SERVER ERROR 14 ms 30.36 KB

Pretty Raw Preview Visualize HTML

```
1 <!doctype html>
2 <html lang=en>
3
4 <head>
5   <title>mysql.connector.errors.DatabaseError: 1644 (45000): Email cannot end with two zeros.
6   // Werkzeug Debugger</title>
7   <link rel="stylesheet" href="??_debugger__=yes&cmd=resource&f=style.css">
8   <link rel="shortcut icon" href="??_debugger__=yes&cmd=resource&f=console.png">
9   <script src="??_debugger__=yes&cmd=resource&f=debugger.js"></script>
10  <script>
11    var CONSOLE_MODE = false,
12        EVALEX = true,
13        EVALEX_TRUSTED = false,
14        SECRET = "LA9sfqpWTMBok1uS0vA2";
15  </script>
16 </head>
```

3) Для певного стовпця допускається ввід лише таких імен: 'Svitlana', 'Petro', 'Olha', 'Taras'.

```
CREATE TABLE allowed_names (  
  id INT PRIMARY KEY AUTO_INCREMENT,  
  name VARCHAR(50) NOT NULL  
);
```

```
DELIMITER //
```

```
CREATE TRIGGER validate_allowed_names
```

```
BEFORE INSERT ON allowed_names
```

```
FOR EACH ROW
```

```
BEGIN
```

```
  IF NEW.name NOT IN ('Svitlana', 'Petro', 'Olha', 'Taras') THEN
```

```
    SIGNAL SQLSTATE '45000'
```

```
    SET MESSAGE_TEXT = 'Only the names Svitolana, Petro, Olha, or Taras  
are allowed.';
```

```
  END IF;
```

```
END;//
```

```
DELIMITER ;
```

POST http://127.0.0.1:5000/allowed_names

Body:

```
{  
  "name": "Ivan"  
}
```

500 INTERNAL SERVER ERROR • 18 ms • 29.9 KB

HTML body:

```
<!doctype html>  
<html lang=en>  
  
<head>  
  <title>mysql.connector.errors.DatabaseError: 1644 (45000): Only the names Svitolana, Petro, Olha, or Taras are allowed.  
  // Werkzeug Debugger</title>  
  <link rel="stylesheet" href="?__debugger__=yes&cmd=resource&f=style.css">  
  <link rel="shortcut icon" href="?__debugger__=yes&cmd=resource&f=console.png">
```

POST http://127.0.0.1:5000/allowed_names

Body:

```
{  
  "name": "Svitlana"  
}
```

201 CREATED • 12 ms • 216 B

JSON body:

```
{  
  "id": 1  
}
```

Висновок

Було реалізовано бекенд на Flask+Python з підключенням до MySQL.

Реалізовані методи GET, POST, PUT, DELETE. Також усі типи зв'язків: 1:1, 1:M, M:M. Були реалізовані тригери, процедури та функції для нашого бекенду.