Національний технічний університет України "Київський політехнічний інститут імені Ігоря Сікорського"

КОМП'ЮТЕРНИЙ ПРАКТИКУМ 2.

Реалізація основної структури даних. Реалізація команд додавання даних та виводу всіх доданих елементів.

Підготувала студентка ФБ-34:
Царик Єлизавета

```
i = 0;
tables = {};
values = {};
```

Оголошення глобальних змінних.

Функція для створення таблиць: CREATE

```
def create(c):
    is_var1_correct = is_correct(c[1])
    if not is_var1_correct:
        print('ERROR: variable entered incorrectly.')
        return;
    global tables
    if c[1] in tables:
        print(f'ERROR: cannot create "{c[1]}" again.')
        return;
    columns = words_inside_brackets(c)
    tables[c[1]]=columns
    print(f'Table "{c[1]}" was created.')
    return
```

Функція викликає допоміжну функцію, що дозволяє перевірити, чи змінна є правильною (чи не порушуються вимоги задання змінної). У глобальну змінну tables заноситься створена таблиця. У форматі назва таблиці: список колонок таблиці. Перед цим проводиться перевірка чи не існує таблиці з таким значенням, якщо існує – виводить повідомлення про це. Список колонок таблиці формується за допомогою іншої функції.

Результат роботи:

```
In [5]: %runfile 'C:/Users/User/Desktop/Новый компьютер/КПІ/2025-26/Основи аналізу алгоритмів/lab2/lab2_FB-34_Tsaryk.py' --wdir
Accepts commands (case insensitive): CREATE, INSERT INTO, INSERT, SELECT FROM
Variable names (first character is a letter, other letters/digits/_)
Command is read until ';'
To finish, type 'stop;'

c: create cat (1, 2);
Table "cat" was created.
c: create cat (3, 4);
ERROR: cannot create "cat" again.
```

```
c: create 1cat (1, 2);
ERROR: variable entered incorrectly.
```

```
c: create dog (3, 4);
Table "dog" was created.
```

Функція для перевірки змінної

```
def is_correct(x):
    allowed_all = set("QWERTYUIOPLKJHGFDSAZXCVBNMqwertyuioplkjhgfdsazxcvbnm_1234567890")
    allowed_first = set("QWERTYUIOPLKJHGFDSAZXCVBNMqwertyuioplkjhgfdsazxcvbnm")
    if x[0] not in allowed_first:
        return False
    for ch in x:
        if ch not in allowed_all:
            return False
    return True
```

Функція для виокремлення назв стовпців

```
def words_inside_brackets(tokens):
    result = []
    inside = False
    temp = []

    for tok in tokens:
        if tok.startswith("(") and tok.endswith(")"):
            inner = tok.strip("()")
            inner = inner.translate(str.maketrans('', '', string.punctuation))
            result.extend(inner.split())
        elif tok.startswith("("):
            inside = True
            temp = [tok.lstrip("(")]
        elif tok.endswith(")") and inside:
            temp.append(tok.rstrip(")"))
            joined = " ".join(temp)
            joined = joined.translate(str.maketrans('', '', string.punctuation))
            result.extend(joined.split())
            inside = False
        elif inside:
            temp.append(tok)
    return result
```

Функція для запису у таблицю: INSERT/INSERT INTO

```
def insert(c, is_into):
    if is_into:
        table_name = c[2]
    else:
        table_name = c[1]
    values_in = words_inside_brackets(c);
    global tables
    global values
    global i
    if table_name in tables:
        if len(values_in) == len(tables[table_name]):
            table_name_i = f"{table_name}_{i}"
            values[table_name_i] = values_in;
            i=i+1;
            print(f'Values were inserted into "{table_name}".')
            return
        else:
            print('ERROR: number colums != number values.')
            return;
    else:
        print(f'ERROR: no table with name "{table_name}".')
        return
```

Оскільки команда має дві варіації (з INTO та без) на початку визначаємо на якому місці стоїть назва таблиці, у яку вносяться дані. Далі перевіряється чи є така назва взагалі. Перевіряємо чи сходиться кількість рядків, що існує в таблиці і ті що ми хочемо вставити. Після цього вписуємо список значень (сформований тою ж допоміжною функцією, що і у CREATE) і (назву

таблиці_індекс). Це потрібно, щоб за потреби ми могли не обмежуватись одним записом даних у таблицю.

Результат роботи:

```
c: insert into cat1 (a, b);
ERROR: no table with name "cat1".
c: insert into cat (a, b, c);
ERROR: number colums != number values.
c: insert into cat (a, b);
Values were inserted into "cat".
c: insert into cat (c, d);
Values were inserted into "cat".
c: insert dog (a, d);
Values were inserted into "dog".
```

Функція для вибірки з таблиць: SELECT

```
def select(c, len_c_words):
    if ('join' in c)and('where' in c):
        if 'on' in c:
            name, rows = join_on(c[2], c[4], True, c[6], c[8])
            columns, result = where(name, rows, c[10], c[11], c[12])
            print_table(columns, result)
        else:
            name, rows = join_on(c[2], c[4], False, None, None)
            columns, result = where(name, rows, c[6], c[7], c[8])
            print_table(columns, result)
        return

    elif 'join' in c:
        if 'on' in c:
            name, rows = join_on(c[2], c[4], True, c[6], c[8])
        else:
            name, rows = join_on(c[2], c[4], False, None, None)
        print_table(name, rows)
        return
    elif 'where' in c:
        if check_table(c[2]):
            columns, result = where(tables[c[2]], get_rows(c[2]), c[4], c[5], c[6])
            print_table(columns, result)
        return
    elif (len_c_words == 3):
        if check_table(c[2]):
            rows = get_rows(c[2])
            print_table(tables[c[2]], rows)
        return
    else:
        print('ERROR: failed to recognize command keywords.')
        print('Try: SELECT FROM table_name_1 [JOIN table_name_2 [ON t1_column = t2_column]] [WHERE condition];')
        return
```

Спочатку шукаємо ключові слова, адже ця функція має найбільшу різноманітність. Я розділила їх всі на 5 основних груп:

1) Немає жодних допоміжних слів (довжина = 3)
   SELECT FROM table_name_1
   Для неї просто викликаємо функцію перевірки чи існує така таблиця, функцію, що повертає рядки таблиці, та функцію, що виводиться нашу таблицю.

```
c: select from dog;
['3', '4']
------------
['a', 'd']
c: select from cat;
['1', '2']
------------
['a', 'b']
['c', 'd']
```

2) Має JOIN (можливо ON)
   SELECT FROM table_name_1 JOIN table_name_2 [ON t1_column = t2_column]
   Перевіряємо чи є ON, в залежності від чого використовуємо різні параметри для виклику функції join_on, яка власне і виконує з'єднування таблиць.

```
c: select from dog join cat;
['3', '4', '1', '2']
------------
['a', 'd', 'a', 'b']
['a', 'd', 'c', 'd']
c: select from cat join dog;
['1', '2', '3', '4']
------------
['a', 'b', 'a', 'd']
['c', 'd', 'a', 'd']
c: select from cat join dog on 1 = 3;
['1', '2', '3', '4']
------------
['a', 'b', 'a', 'd']
c: select from cat join dog on 1 = 4;
['1', '2', '3', '4']
------------
```

3) Має WHERE:
   SELECT FROM table_name_1 WHERE condition;
   При чому condition має мати структуру: колонку, яку порівнюємо – знак - з чим порівнюємо.
   Можливі знаки: >, <, =
   Порівнюється довжина рядків.
   Викликається функція where.

```
c: select from cat where 1 > 1;
['1', '2']
------------
c: insert dog (a, abc);
Values were inserted into "dog".
c: select from dog where 3 > 1;
['3', '4']
------------
c: select from dog where 4 > 1;
['3', '4']
------------
['a', 'abc']
c: select from dog where 4 = 123;
['3', '4']
------------
['a', 'abc']
```

4) Має JOIN (можливо ON) та WHERE:
   SELECT FROM table_name_1 JOIN table_name_2 [ON t1_column = t2_column] WHERE condition;
   У цій частині ми перевіряємо чи вжито ON і викликаємо спочатку функцію join_on, потім where.

```
c: select from cat join dog on 1 = 3 where 4 > 1;
['1', '2', '3', '4']
------------
['a', 'b', 'a', 'abc']
c: stop;
```

5) Не підходить під жодну з цих категорій:
   Буде виведено повідомлення про помилку.

Функція для виводу таблиці

```
def print_table(table_name, table_rows):
    if (table_name == None) or (table_rows == None):
        print('ERROR: Columns or rows are not found.')
        return
    print(table_name)
    print('------------')
    for row in table_rows:
        print(row)
```

Функція для знаходження рядків

```
def get_rows(table_name):
    rows = []
    for key, val in values.items():
        if key.startswith(f"{table_name}_"):
            rows.append(val)
    return rows
```

Функція для перевірки таблиці

```
def check_table(table_name):
    if table_name not in tables:
        print(f'ERROR: no table with name "{table_name}".')
        return False;
    return True;
```

Функція join_on

```
def join_on(table_name_1, table_name_2, on, column_1, column_2):
    if (check_table(table_name_1)) and (check_table(table_name_2)):
        rows1 = get_rows(table_name_1)
        rows2 = get_rows(table_name_2)
        cols1 = tables[table_name_1]
        cols2 = tables[table_name_2]

        if not on:
            joined_rows = []
            for r1 in rows1:
                for r2 in rows2:
                    joined_rows.append(r1 + r2)
            joined_cols = cols1 + cols2
            return joined_cols, joined_rows

        if column_1 not in cols1 or column_2 not in cols2:
            print("ERROR: column name not found.")
            return None, None

        i1 = cols1.index(column_1)
        i2 = cols2.index(column_2)

        joined_rows = []

        for r1 in rows1:
            for r2 in rows2:
                if r1[i1] == r2[i2]:
                    joined_rows.append(r1 + r2)
        joined_cols = cols1 + cols2
        return joined_cols, joined_rows
    else:
        return None, None
```

Функція where

```python
def where(columns, rows, col, op, val):
    if col not in columns:
        print(f"ERROR: column '{col}' not found.")
        return None, None

    col_index = columns.index(col)
    val_len = len(val.strip('"'))

    if op not in ['>', '<', '=']:
        print(f"ERROR: unknown operator '{op}'. Use one of: >, <, =")
        return None, None

    result = []
    for row in rows:
        cell_value = row[col_index]
        cell_len = len(cell_value)

        if (
            (op == '>' and cell_len > val_len) or
            (op == '<' and cell_len < val_len) or
            (op == '=' and cell_len == val_len)
        ):
            result.append(row)

    return columns, result
```