**Лабораторна робота 12.**

**Функції користувача та їх використання у запитах.**

**Мета роботи:** навчитись створювати та виконувати у запитах різні типи користувацьких функцій. Розглянути використання table-return функцій в операторі apply.

**Задачі:**

* Три типи функцій користувача та загальний синтаксис їх створення
* Робота із функціями scalar
* Table-return функції. In-line та multi-statement
* Використання Table-return в операторі apply
* Виконання індивідуального завдання згідно варіанту

1. Архітектура збережених процедур має істотний недолік - їх не можна вказувати безпосередньо в виразах. Щоб використовувати повертається збереженої процедурою значення у виразах, необхідно виконати привласнення повернутого значення змінній, яка потім і підставляється в вираз. Подібний метод не можна назвати зручним.

На даний випадок в T-SQL є можливість створення функцій користувача (user-defined function) (в розпорядженні користувача є також вбудовані (builtin) або, як їх ще називають, системні функції). Функції користувача є самостійними об'єктами бази даних, такими як, наприклад, збережені процедури або тригери. Таким чином, функція користувача розташовується в певній базі даних і доступна тільки в контексті цієї бази даних.

При створенні функцій користувача є ряд обмежень, які слід враховувати. Так, наприклад: визначені користувачем функції не можуть виконувати дії, що змінюють стан бази даних; функції не можуть містити інструкцію OUTPUT INTO, назначенням якої є таблиця; функції не можуть повертати кілька результуючих наборів; обробка помилок у функціях обмежена; функції не можуть викликати збережену процедуру; функції не можуть використовувати динамічний SQL і тимчасові таблиці, але табличні змінні дозволені до використання; рівень вкладеності функцій не має перевищувати 32.

1. Серед функцій користувача виділяють скалярні та такі, що повертають таблицю (набор рядків). Останні ще розділяють на in-line та multi-statement.

Scalar - ці функції характеризуються тим, що повертають звичайне скалярне значення практично будь-якого з типів даних, що підтримує SQL Server Функція може включати багато команд, які об'єднуються в один блок за допомогою конструкції BEGIN ... END.

 Inline – особливістю цієї функції є те, що вона містить всього одну команду - запит SELECT. За допомогою цього запиту формується набір даних, який і повертається користувачеві у вигляді значення типу даних table.

 Multi-statement - повертають користувачеві значення типу даних table, що містить набір даних, однак тіло функції може містити багато команд Transact-SQL. Саме за допомогою цих команд і формується набір даних, повертається після виконання функції.

1. Почнемо із функцій типу scalar. Наведемо загальний синтаксис для створення функції даного типу (рис.1)

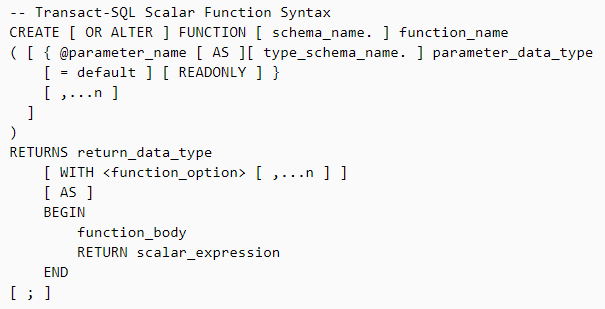


Рис.1. Загальний синтаксис команди створення функції типу scalar

При створенні функції можна вказати схему даних, обов’язково навести унікальне у межах схеми ім’я, та у разі необхідності набір вхідших параметрів із типами даних та значеннями за замовченням. Вихідних параматрів функція не має. Далі вказуємо тип, що буде повертати функція та наводимо саме тіло, в якому обов’язково має бути ключове слово return із вказанням значення або змінної потрібного типу даних.

Далі розглянемо декілька прикладів скалярних функцій. Реалізуємо функцію без параметрів, що повертає середній бал студентів з усіх дисциплін (рис.2)

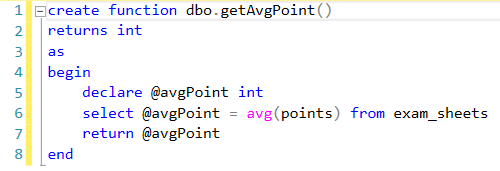


Рис.2. Скалярна функція без параметрів

Виклик даної процедури наводиться на рис.3.

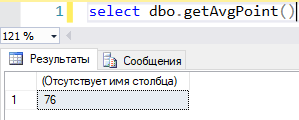


Рис.3. Виклик скалярної функції без параметрів

Додамо до процедури вхідний параметр – код студента, та повернемо середній бал студента із цим кодом (рис.4)

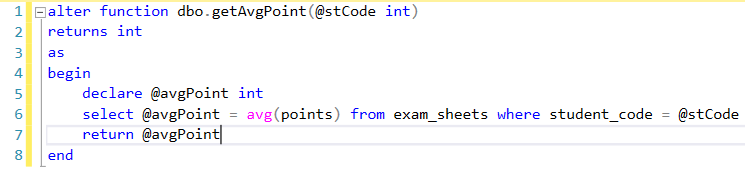


Рис.4. Скалярна функція із параметром

Використаємо процедуру у запитах. Виведемо список студентів групи 308 із середнім балом, впорядкований за спаданням значення середнього балу (рис.5)

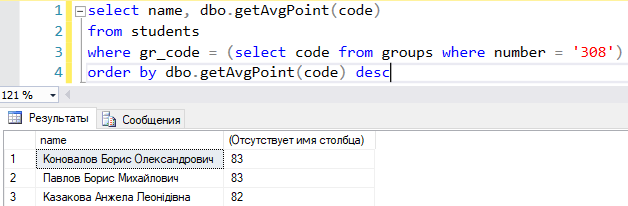


Рис.5. Використання скалярної функції в select та order by

Використовуючи дану функцію, виведемо середній бал студента у групі (рис.6)

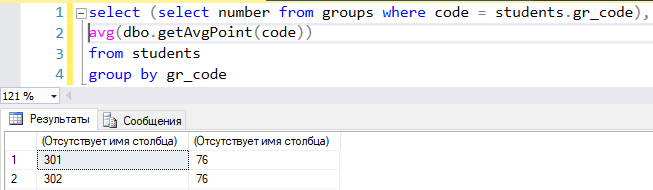


Рис.6. Використання скалярної функції в функціях агрегації

В наступному прикладі використаємо дану функцію для підрахунку кількості студентів із різними середніми балами (рис.7)

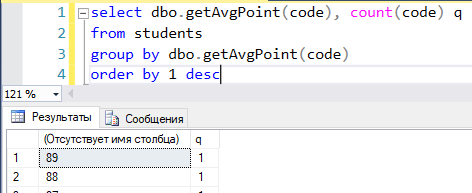


Рис.7. Використання скалярної функції в group by

Розглянемо ще один приклад реалізації скалярної функції, що повертає вік студента в роках на даний момент за його датою народження (рис.8)

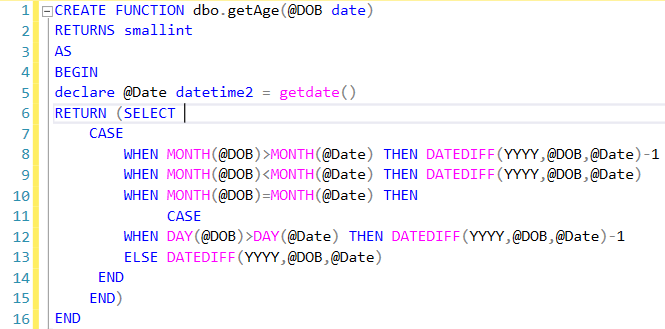


Рис.8. Вік у роках за датою народження

Використаємо дану функцію для отримання студентів, що мають повних 22 роки на поточну дату (рис.9)

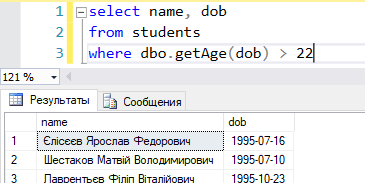


Рис.9. Студенти, старші 22 років

! УВАГА. Пам’ятайте, що фільтрація даних за результатом, що повертає функція користувача не використовує індекси, і тому є як правило дуже повільним. Як наслідок, не рекомендується фільтрувати великі набори даних із використанням функцій користувача. Але якщо даний фільтр буде використаний разом із іншим, що попередньо звузить вибірку, продуктивність запиту не зазнає істотних втрат. Наприклад, як при виборі студентів 308 групи, старших 20 років (рис.10)

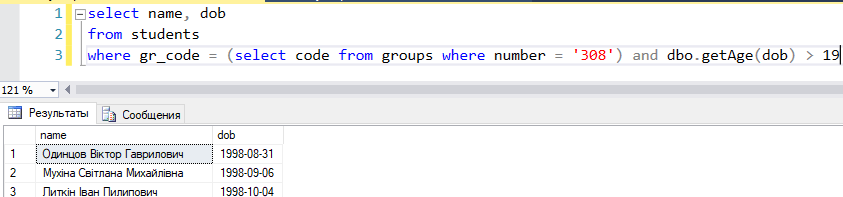


Рис.10. Використання подвійної фільтрації при використанні функцій користувача

1. Розглянемо наступний тип функцій користувача – функції in-line. І почнемо, як завжди, із наведення загального синтаксису створення такої функції (рис.11)

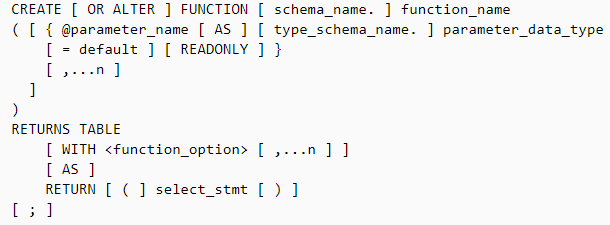


Рис.11. Загальний синтаксис створення функції in-line

Зосередимось лише на відмінностях від синтаксису scalar. По перше, після returns вказується тип даних table, при чому без вказання конкретної структури таблиці, що буде повертатись. Також функція складається із однієї команди – а саме return, що повертає результат роботи запиту select, як таблицю.

Реалізуємо декілька прикладів in-line функцій. Для порівняння із scalar реалізуємо перший приклад (рис.2), що повертає середній бал студентів з усіх дисциплін, за допомогою функції in-line (рис.12)

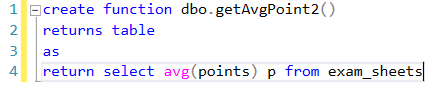


Рис.12. In-line функція без параметрів

Звернення до такої функції також буде дещо відрізнятись від попереднього. Якщо раніше ми звертались до результату функції, як до скалярного значення у розділах select, where, having та ін., то тепер ми будемо звертатись до неї, як до таблиці у розділі from (рис.13)

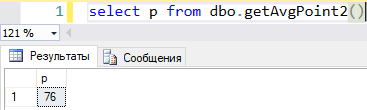


Рис.13. Звернення до функції In-line

Також до In-line функції можна звернутись, як до вкладеного запиту. Наприклад, виберемо студентів, у яких середній бал вище за загальний середній (рис.14)

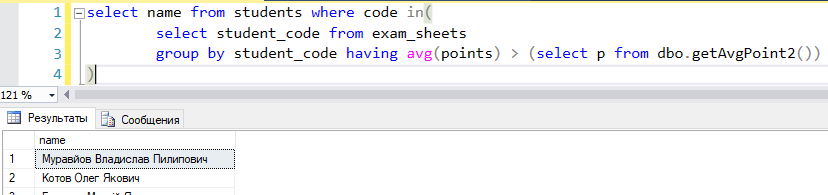


Рис.14. In-line функція, як вкладений запит

Далі розглянемо In-line функції із параметрами, що дозволяють реалізувати більш складні конструкції. Реалізуємо функцію, що повертає дисципліни, які складалися студентами певної групи (рис.15)

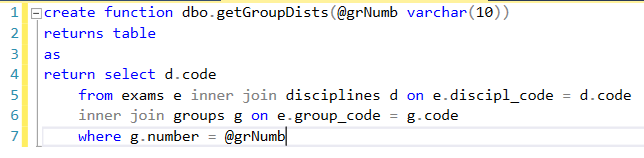


Рис.15. In-line функція із параметром

Далі використовуючи дану функцію, вберемо дисципліни, що складала 308, але не складала 301 (рис.16)

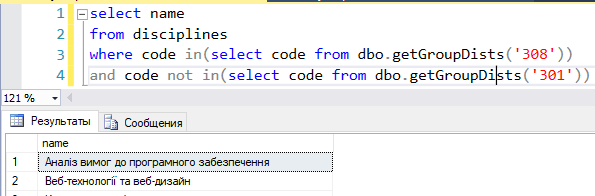


Рис.16. Використання In-line функції із параметром

1. Третій вид функцій користувача – multi-statement. Загальний синтаксис створення функції такого виду наведено на рис.17.

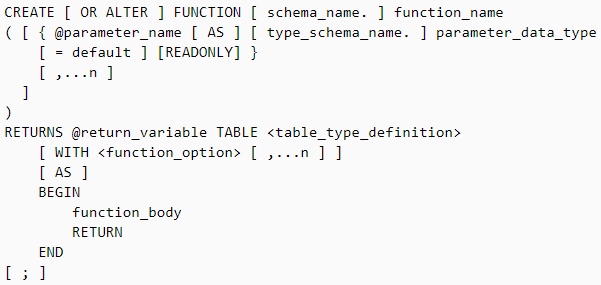
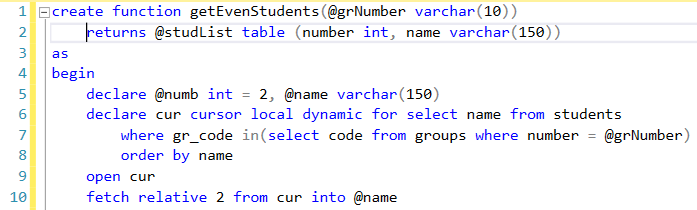


Рис.17. Команда створення функції multi-statement

Знову звернемо увагу лише на відмінності від попередньо розглянутих двох типів. Вони в основному стосуються розділу RETURNS. Тут ми повинні вказати ім’я змінної із типом TABLE, що буде повертатись, як результат роботи функції та навести всі поля даної таблиці із вказанням їх типів даних. Після завершення тіла функції вказуємо лише ключове слово RETURN без параметрів, оскільки функція вже знає, що саме потрібно повернути.

Розглянемо приклад функції, що повертає парних за списком студентів певної групи (рис.18)



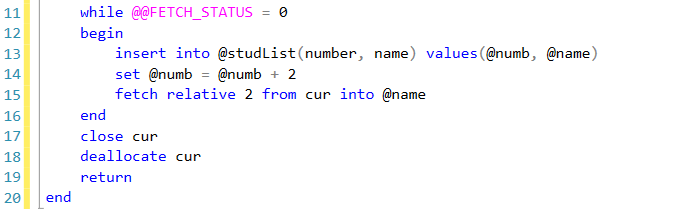


Рис.18. Функція multi-statement

Використовується функція multi-statement так само, як і in-line, оскільки і та і інша повертають таблиці. Приклад виклику функції, наведеної на рис.18, що виводить парних за списком студентів 301 групи, приводиться нижче на рис.19.

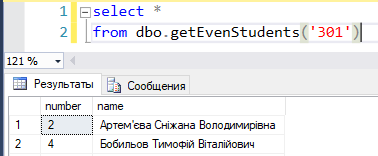


Рис.19. Виконання функції multi-statement

1. Розглянемо ще одну цікаву можливість використання table-returns функцій – наведення їх у лівій частині команди cross apply або outer apply. Реалізуємо приклад, що розглядався у лабораторній роботі 6 на рис.4, та вибирає перших двох студентів за рейтингом для кожної групи. Але тепер із використанням функцій користувача. Для початку створимо функцію, що повертає деяку кількість перших за рейтингом студентів певної групи (рис.20)

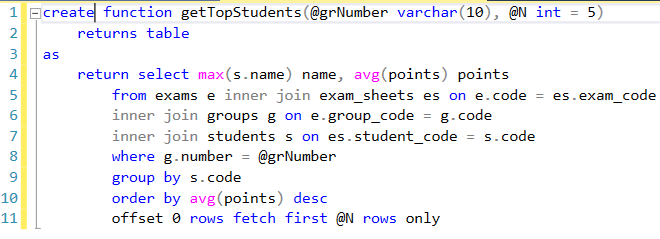


Рис.20. Перші N за рейтингом студентів певної групи

Використаємо дану функцію у запиті для вибірки перших 2-х студентів для кожної групи (рис.21)

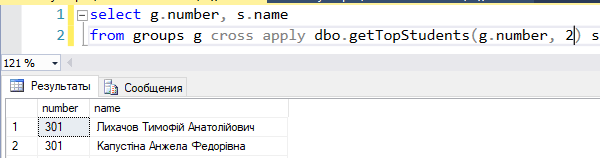


Рис.21. Використання функцій користувача у cross apply

1. Кінець роботи

**Завдання для індивідуального виконання.**

Для своєї БД розробити набір функцій користувача із використанням вхідних параметрів (мінімум по одній scalar, in-line та multi-statement.

Продемонструвати використання функцій у реченнях select, from, where, та group by. Використати результат table-return функції у вигляді вкладеного підзапиту та як частину оператора apply.