МІНІСТЕРСТВО ОСВІТІ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Методичні вказівки до виконання лабораторної роботи за темою «Аналітична обробка оперативних даних засобами мови SQL у середовищі СУБД Microsoft SQL Server»

для студентів, що навчаються за спеціальністю
«122 Комп’ютерні науки»

Затверджено
редакційно-видавничою
радою університету,
протокол № 1 від 15.02.2024 р.

Харків
НТУ ХПІ
2024

Методичні вказівки до виконання лабораторної роботи за темою «Аналітична обробка оперативних даних засобами мови SQL у середовищі СУБД Microsoft SQL Server» для студентів, що навчаються за спеціальністю 122 «Комп’ютерні науки» / Уклад. Орловський Д.Л., Копп А.М. – Харків: НТУ «ХПІ», 2024. – 75 с.

Укладачі Д.Л. Орловський,
А.М. Копп

Рецензент Гринченко М.А.

Кафедра програмної інженерії та інтелектуальних технологій управління

ЗМІСТ

Вступ 4
1 Теоретичні відомості 5
1.1 Загальні відомості про засоби обробки даних мови SQL 5
1.2 З'єднання і теоретико-множинні операції над відношеннями 10
1.3 Обчислення і підведення підсумків в запитах 19
1.4 Використання підзапитів 22
1.5 Створювані користувачем функції та вбудовані функції 25
1.6 Збережені процедури 33
2 Виконання роботи 41
2.1 Побудова та використання створюваних користувачем функцій 41
2.2 Побудова та виконання запитів SELECT-SQL 53
2.3 Побудова та використання збережених процедур 67
2.4 Збереження результатів роботи 70
3 Вимоги до звіту 71
4 Питання для самоперевірки 72
Список літератури 75

ВСТУП

Сучасні інформаційні системи ґрунтуються на використанні баз даних, в яких накопичується різна інформація. Тому зараз розробляються і значно поширюються методи і засоби роботи з базами даних з метою підвищення ефективності роботи людини у різних галузях діяльності. Ці засоби та методи пов’язані з узагальненням і різними додатковими способами обробки даних. Основні ідеї сучасної інформаційної технології базуються на концепції, згідно з якою дані повинні бути організовані в бази даних з метою адекватного відображення реального світу, що змінюється, і задоволення інформаційних потреб користувачів. Ці бази даних створюються і функціонують під управлінням спеціальних програмних комплексів, які називають системами управління базами даних (СУБД).

Методичні вказівки до лабораторної роботи за темою «Аналітична обробка оперативних даних засобами мови SQL у середовищі СУБД Microsoft SQL Server» призначені для студентів, що навчаються за спеціальністю «122 Комп’ютерні науки».

Виконання лабораторної роботи повинно забезпечити закріплення теоретичних знань і практичних навичок, отриманих при вивченні лекційної частини дисциплін, пов’язаних із проектуванням, розробкою та застосуванням баз даних.

В методичних вказівках розглянуті основні питання, пов’язані з теоретичним обґрунтуванням та безпосереднім виконанням лабораторної роботи.

1 ТЕОРЕТИЧНІ ВІДОМОСТІ

1.1 Загальні відомості про засоби обробки даних мови SQL

Мова запитів DQL (Data Query Language) найбільш відома користувачам реляційної бази даних, не дивлячись на те, що він включає всього одну команду SELECT. Ця команда разом зі своїми численними опціями і пропозиціями використовується для формування запитів до реляційної бази даних. Досить часто мову запитів DQL розглядають як складову DML (Data Manipulation Language).

Оператор SELECT – один з найбільш важливих і найпоширеніших операторів SQL. Він дозволяє робити вибірки даних з таблиць і перетворювати до потрібного виду отримані результати. Будучи дуже потужним, він здатний виконувати дії, еквівалентні операторам реляційної алгебри, причому в межах єдиної виконуваної команди. При його допомозі можна реалізувати складні і громіздкі умови відбору даних з різних таблиць.

Оператор SELECT – засіб, який повністю абстрагований від питань представлення даних, що допомагає сконцентрувати увага на проблемах доступу до даних. Приклади його використання наочно демонструють один із засадничих принципів великих (промислових) СУБД : засоби зберігання даних і доступу до них відокремлені від засобів представлення даних. Операції над даними робляться в масштабі наборів цих, а не окремих записів.

Оператор SELECT має наступний формат:

```
SELECT [ALL | DISTINCT ] {*|[ім'я_стовпця
   [AS        нове_ім'я]]} [,..n]
   FROM  ім'я_таблиці [[AS] псевдонім] [,..n]
   [WHERE  <умова_пошуку>]
   [GROUP BY  ім'я_стовпця [,..n]]
   [HAVING <критерії вибору груп>]
   [ORDER BY ім'я_стовпця [,..n]]
```

Оператор SELECT визначає поля (стовпці), які входитимуть в результат виконання запиту. У списку вони розділяються комами і приводяться в такій черговості, в якій мають бути представлені в результаті запиту. Якщо використовується ім'я поля, що містить пропуски або роздільники, його слід взяти в квадратних дужок. Символом * можна вибрати усі поля, а замість імені поля застосувати вираження з декількох імен.

Якщо обробляється ряд таблиць, то (за наявності однойменних полів в різних таблицях) в списку полів використовується повна специфікація поля, тобто. Ім'я_таблиці.Ім'я_поля.

Пропозиція FROM задає імена таблиць і представлень, які містять поля, перераховані в операторові SELECT. Необов'язковий параметр псевдоніма – це скорочення, що встановлюється для імені таблиці.

Обробка елементів оператора SELECT виконується в наступній послідовності:

- FROM – визначаються імена використовуваних таблиць;
- WHERE – виконується фільтрація рядків об'єкту відповідно до заданих умов;
- GROUP BY – утворюються групи рядків, що мають одно і те ж значення у вказаному стовпці;
- HAVING – фільтруються групи рядків об'єкту відповідно до вказаної умови;
- SELECT – встановлюється, які стовпці мають бути присутніми у вихідних даних;
- ORDER BY – визначається впорядкованість результатів виконання операторів.

Порядок пропозицій і фраз в операторові SELECT не може бути змінений. Тільки дві пропозиції SELECT і FROM є обов'язковими, усі інші можуть бути опущені. SELECT – закрита операція, тобто результат запиту до таблиці (або таблиць) завжди є таблицею.

Параметр WHERE визначає критерій відбору записів з вхідного набору. Але в таблиці можуть бути присутніми записи (дублікати), що повторюються. Предикат ALL задає включення у вихідний набір усіх дублікатів, відібраних за критерієм WHERE. Немає необхідності вказувати ALL явно, оскільки це значення діє за умовчанням.

Результат виконання запиту може містити значення, що дублюються, оскільки на відміну від операцій реляційної алгебри оператор SELECT не виключає значень, що повторюються, при виконанні вибірки даних.

Предикат DISTINCT слід застосовувати в тих випадках, коли вимагається відкинути блоки даних, що містять дублюючі записи у вибраних полях. Значення для кожного з приведених в інструкції SELECT полів мають бути унікальними, щоб запис, що утримує їх, зміг увійти до вихідного набору.

Причиною обмеження в застосуванні DISTINCT є та обставина, що його використання може різко уповільнити виконання запитів.

За допомогою параметра WHERE користувач визначає, які блоки даних з приведених в списку FROM таблиць з'являться в результаті запиту. За ключовим словом WHERE слідує перелік умов пошуку, що визначають ті рядки, які мають бути вибрані при виконанні запиту. Існує п'ять основних типів умов пошуку (чи предикатів):

- порівняння: порівнюються результати обчислення одного вираження з результатами обчислення іншого;
- діапазон: перевіряється, чи потрапляє результат обчислення вираження в заданий діапазон значень;
- приналежність множині: перевіряється, чи належить результат обчислень вираження заданій безлічі значень;
- відповідність шаблону: перевіряється, чи відповідає деяке строкове значення заданому шаблону;
- значення NULL: перевіряється, чи містить цей стовпець визначник NULL (невідоме значення).

У мові SQL можна використовувати наступні оператори порівняння:

- = – рівність;
- < – менше;
- > – більше;
- <= – менше або рівно;
- >= – більше або рівно;
- <> – не рівно.

Складніші предикати можуть бути побудовані за допомогою логічних операторів AND, OR або NOT, а також дужок, використовуваних для визначення порядку обчислення вираження. Обчислення вираження в умовах виконується за наступними правилами:

- вираження обчислюється зліва направо;
- першими обчислюються підвирази в дужках;
- оператори NOT виконуються до виконання операторів AND і OR;
- оператори AND виконуються до виконання операторів OR.

Для усунення будь-якої можливої неоднозначності рекомендується використовувати дужки.

Оператор BETWEEN використовується для пошуку значення усередині деякого інтервалу, визначуваного своїми мінімальним і максимальним значеннями. При цьому вказані значення включаються в умову пошуку.

При використанні заперечення NOT BETWEEN потрібно, щоб значення, що перевіряється, лежало поза межами заданого діапазону.

Оператор IN використовується для порівняння деякого значення із списком заданих значень, при цьому перевіряється, чи відповідає результат обчислення вираження одному зі значень в наданому списку. За допомогою оператора IN може бути досягнутий той же результат, що і у разі застосування оператора OR, проте оператор IN виконується швидше.

NOT IN використовується для відбору будь-яких значень, окрім тих, які вказані в представленому списку.

За допомогою оператора LIKE можна виконувати порівняння вираження із заданим шаблоном, в якому допускається використання символів-замінників:

- символ % – замість цього символу може бути підставлена будь-яка кількість довільних символів;
- символ _ замінює один символ рядка;
- [ ] – замість символу рядка буде підставлений один з можливих символів, вказаний в цих обмежувачах;
- [^] – замість відповідного символу рядка будуть підставлені усі символи, окрім вказаних в обмежувачах.

Оператор IS NULL використовується для порівняння поточного значення зі значенням NULL – спеціальним значенням, що вказує на відсутність будь-якого значення. NULL – це не те ж саме, що знак пропуску (пропуск (або пробіл) – допустимий символ) або нуль (0 – допустиме число). NULL відрізняється і від рядка нульової довжини (порожнього рядка).

IS NOT NULL використовується для перевірки присутності значення в полі.

У загальному випадку рядка в результуючій таблиці SELECT-запиту ніяк не впорядковані. Проте їх можна необхідним чином відсортувати, для чого в оператор SELECT вміщується фраза ORDER BY, яка сортує дані вихідного набору в заданій послідовності. Сортування може виконуватися по декількох полях, в цьому випадку вони перераховуються за ключовим словом ORDER BY через кому. Спосіб сортування задається ключовим словом, що вказується у рамках параметра ORDER BY слідом за назвою поля, по якій виконується сортування. За умовчанням реалізується сортування за збільшенням. Явно вона задається ключовим словом ASC. Для виконання сортування в зворотній послідовності необхідно після імені поля, по якому вона виконується, вказати ключове слово DESC. Фраза ORDER BY дозволяє упорядкувати вибрані записи в порядку зростання або убування значень будь-якого стовпця або комбінації стовпців,

незалежно від того, є присутніми ці стовпці в таблиці результату або ні. Фраза ORDER BY завжди має бути останнім елементом в операторові SELECT.

У фразі ORDER BY може бути вказане і більше одного елементу. Головний (перший) ключ сортування визначає загальну впорядкованість рядків результуючої таблиці. Якщо в усіх рядках результуючої таблиці значення головного ключа сортування є унікальними, немає необхідності використовувати додаткові ключі сортування. Проте, якщо значення головного ключа не унікальні, в результуючій таблиці буде присутніми декілька рядків з одним і тим же значенням старшого ключа сортування. В цьому випадку, можливо, доведеться упорядкувати рядки з одним і тим же значенням головного ключа по якому-небудь додатковому ключу сортування.

1.2 З'єднання і теоретико-множинні операції над відношеннями

Розглянемо основні операції над відношеннями, які можуть представляти інтерес з точки зору витягання даних з реляційних таблиць. Це об'єднання, перетин, різниця, розширене декартовий добуток відношень, а також спеціальні операції над відношеннями: вибірка, проекція і з'єднання.

Для ілюстрації теоретико-множинних операцій над відношеннями введемо абстрактні відношення (таблиці) з деякими атрибутами (полями). Їх структури наведені на рисунках 1.1 та 1.2 відповідно.

Відношення R	
R.a1	**R.a2**
A	1
A	2
B	1
B	3
B	4

Рисунок 1.1

Відношення S	
S.b1	**S.b2**
1	h
2	g
3	h

Рисунок 1.2

Запити SQL, за допомогою яких створюються ці таблиці, мають вигляд.

```
CREATE TABLE R
((a1 CHAR(1), a2 INT, PRIMARY KEY(a1, a2))

CREATE TABLE S
((b1 INT PRIMARY KEY, b2 CHAR(1))
```

Операції вибірки і проекції є унарними, оскільки вони працюють з одним відношенням.

Операція вибірки – побудова горизонтальної підмножини, тобто підмножини кортежів, що мають задані властивості.

Операція вибірки працює з одним відношенням R і визначає результуюче відношення, яке містить тільки ті кортежі (рядки) відношення R, які задовольняють заданій умові F (предикату).

Операція вибірки в SQL записується таким чином:

```
SELECT a1, a2
FROM R
WHERE a2=1
```

Операція проекції – побудова вертикальної підмножини відношення, тобто підмножини кортежів, що отримується вибором одних і виключенням інших атрибутів.

Операція проекції працює з одним відношенням R і визначає нове відношення, яке містить вертикальну підмножину відношення R, що

створюється за допомогою витягання значень вказаних атрибутів і виключення з результату рядків-дублікатів.

Операція проекції в SQL записується таким чином:

SELECT DISTINCT b2

FROM S

Декартовий добуток RxS двох відношень (двох таблиць) визначає нове відношення - результат конкатенації (тобто зчеплення) кожного кортежу (кожному запису) з відношення R з кожним кортежем (кожним записом) з відношення S.

RxS={(a, 1, 1, h) (a, 2, 1, h),

((b, 1, 1, h), .. }

Декартовий добуток відношень в SQL має вигляд.

SELECT R.a1, R.a2, S.b1, S.b2

FROM R, S

Результат декартового добутку двох відношень показаний на рисунку 1.3.

R x S			
R.a1	**R.a2**	**S.b1**	**S.b2**
a	1	1	h
a	1	2	g
a	1	3	h
a	2	1	h
a	2	2	g
a	2	3	h
b	1	1	h
b	1	2	g
b	1	3	h
b	3	1	h
b	3	2	g
b	3	3	h
b	4	1	h
b	4	2	g
b	4	3	h

Рисунок 1.3

Якщо одно відношення має N записів і K полів, а інше M записів і L полів, то відношення з їх декартовим добутком міститиме NxM записів і K+L полів. Початкові відношення можуть містити поля з однаковими іменами, тоді імена полів міститимуть назви таблиць у вигляді префіксів для забезпечення унікальності імен полів у відношенні, отриманому як результат виконання декартового добутку.

Проте у такому вигляді (рисунок 1.3) відношення містить більше інформації, чим зазвичай необхідно користувачеві. Як правило, користувачів цікавить лише деяка частина усіх комбінацій записів в декартовому добутку, що задовольняє деякій умові. Тому замість декартового добутку зазвичай використовується одна з найважливіших операцій реляційної алгебри – операція з'єднання, яка є похідною від операції декартового добутку. З точки зору ефективності реалізації в реляційних СУБД ця операція - одна з найважчих і часто входить до числа головних причин, що викликають властиві усім реляційним системам проблеми з продуктивністю.

З'єднання – це процес, коли дві або більше за таблицю об'єднуються в одну. Здатність об'єднувати інформацію з декількох таблиць або запитів у вигляді одного логічного набору даних обумовлює широкі можливості SQL.

У мові SQL для завдання типу з'єднання таблиць в логічний набір записів, з якого вибиратиметься необхідна інформація, використовується операція JOIN в пропозиції FROM.

Формат операції:

FROM ім'я_таблиці_1 {INNER | LEFT | RIGHT}
JOIN ім'я_таблиці_2
ON умова_з'єднання

Існують різні типи операцій з'єднання :

- тета-з’єднання;
- з'єднання по еквівалентності;
- природне з'єднання;

- зовнішнє з'єднання;
- напівз'єднання.

Операція тета-з’єднання визначає відношення, яке містить кортежі з декартових добутків відношень R і S, що задовольняють предикату F. Предикат F має вигляд , де замість може бути вказаний один з операторів порівняння (>,<=, =, <>).

Якщо предикат F містить тільки оператор рівності (=), то з'єднання називається з'єднанням по еквівалентності.

Результат з’єднання двох відношень показаний на рисунку 1.4.

R.a1	**R.a2**	**S.b1**	**S.b2**
a	1	1	h
a	2	2	g
b	3	3	h
b	1	1	h

Рисунок 1.4

Операція тета-з’єднання в мові SQL називається INNER JOIN (внутрішнє з'єднання) і використовується, коли треба включити усі рядки з обох таблиць, що задовольняють умові об'єднання. Внутрішнє з'єднання має місце і тоді, коли в пропозиції WHERE порівнюються значення полів з різних таблиць. В цьому випадку будується декартовий добуток рядків першою і другою таблиць, а з отриманого набору даних відбираються записи, що задовольняють умовам об'єднання.

В умовах з'єднання можуть брати участь поля, що відносяться до одного і тому ж типу даних і таких, що містять один і той же вид даних, але вони не обов'язково повинні мати однакові імена.

Блоки даних з двох таблиць об'єднуються, як тільки у вказаних полях будуть знайдені співпадаючі значення.

Якщо в пропозиції FROM перераховано декілька таблиць і при цьому не вживається специфікація JOIN, а для вказівки відповідності полів

з таблиць використовується умова в пропозиції WHERE, то деякі реляційні СУБД (наприклад, Access) оптимізують виконання запиту, інтерпретуючи його як з'єднання.

Якщо перераховувати ряд таблиць або запитів і не вказувати умови з'єднання, в якості початкової таблиці буде вибрано декартовий (прямий) добуток усіх таблиць.

Тета-з'єднання відношень в SQL має вигляд.

```
SELECT R.a1, R.a2, S.b1, S.b2
FROM R, S
WHERE R.a2=S.b1
```

чи

```
SELECT R.a1, R.a2, S.b1, S.b2
FROM R INNER JOIN S ON  R.a2=S.b1
```

Природним з'єднанням називається з'єднання по еквівалентності двох відношень R і S, виконане по усіх загальних атрибутах, з результатів якого виключається по одному екземпляру кожного загального атрибуту.

Результат природного з'єднання двох відношень показаний на рисунку 1.5.

R.a1	**R.a2 або S.b1**	**S.b2**
a	1	h
a	2	g
b	3	h
b	1	h

Рисунок 1.5

Природне з'єднання відношень в SQL має вигляд.

```
SELECT R.a1, R.a2, S.b2
FROM R, S
WHERE R.a2=S.b1
```

чи

```
SELECT R.a1, S.b1, S.b2
FROM R INNER JOIN S ON  R.a2=S.b1
```

Зовнішнє з'єднання схоже на внутрішнє, але в результуючий набір даних включаються також записи провідної таблиці з'єднання, які об'єднуються з порожньою безліччю записів іншої таблиці.

Яка з таблиць буде такою, що веде, визначає вид з'єднання. LEFT – ліве зовнішнє з'єднання, ведучою є таблиця, розташована зліва від виду з'єднання; RIGHT – праве зовнішнє з'єднання, провідна таблиця розташована праворуч від виду з'єднання.

Лівим зовнішнім з'єднанням називається з'єднання, при якому кортежі відношення R, що не мають співпадаючих значень в загальних стовпцях відношення S, також включаються в результуюче відношення.

Результат лівого зовнішнього з’єднання двох відношень показаний на рисунку 1.6.

R.a1	**R.a2**	**S.b1**	**S.b2**
a	1	1	h
a	2	2	g
b	1	1	h
b	3	3	h
b	4	null	null

Рисунок 1.6

Ліве зовнішнє з'єднання відношень в SQL має вигляд.

```
SELECT R.a1,  R.a2, S.b1, S.b2
FROM R LEFT JOIN S ON R.a2=S.b1
```

Існує і праве зовнішнє з'єднання, зване так тому, що в результуючому відношенні містяться усі кортежі правого відношення.

Крім того, є і повне зовнішнє з'єднання, в його результуюче відношення поміщаються усі кортежі з обох відношень, а для позначення неспівпадаючих значень кортежів в нім використовуються визначники NULL.

Праве зовнішнє з'єднання відношень в SQL має вигляд.

```
SELECT R.a1,  R.a2, S.b1, S.b2
FROM R RIGHT JOIN S ON  R.a2=S.b1
```

Операція напівз'єднання визначає відношення, що містить ті кортежі відношення R, які входять в з'єднання відношень R і S .

Результат напівз'єднання двох відношень показаний на рисунку 1.7.

R.a1	**R.a2**
a	1
a	2
b	3
b	1

Рисунок 1.7

Напівз'єднання відношень в SQL має вигляд.

```
SELECT R.a1, R.a2
FROM R, S
WHERE R.a2=S.b1
```

чи

```
SELECT R.a1,   R.a2
FROM R INNER JOIN S ON R.a2=S.b1
```

Об'єднання (UNION) відношень R і S можна отримати в результаті їх конкатенації з утворенням одного відношення з виключенням кортежів-дублікатів. При цьому відношення R і S мають бути сумісні, тобто мати

однакову кількість полів із співпадаючими типами даних. Інакше кажучи, відношення мають бути сумісні по об'єднанню.

Об'єднанням двох таблиць R і S є таблиця, що містить усі рядки, які є в першій таблиці R, в другій таблиці S або в обох таблицях відразу.

Об'єднання відношень в SQL має вигляд.

```
SELECT R.a1, R.a2
FROM R
UNION
SELECT S.b2, S.b1
FROM S
```

Операція перетину (INTERSECT) визначає відношення, яке містить кортежі, присутні як відносно R, так і відносно S. Відношення R і S мають бути сумісні по об'єднанню .

Перетином двох таблиць R і S є таблиця, що містить усі рядки, присутні в обох початкових таблицях одночасно.

Перетин відношень в SQL має вигляд.

```
SELECT R.a1, R.a2
FROM R, S
WHERE R.a1=S.b1 AND R.a2=S.b2
```

чи

```
SELECT R.a1, R.a2
FROM R
WHERE R.a1 IN
   (    (SELECT S.b1 FROM S
         WHERE S.b1=R.a1) AND R.a2 IN
        (         (SELECT S.b2
                   FROM S
                   WHERE S.b2=R.a2)
```

Різниця (EXCEP) R - S двох відношень R і S складається з кортежів, які є відносно R, але відсутні відносно S. Причому відношення R і S мають бути сумісні по об'єднанню.

Різницею двох таблиць R і S є таблиця, що містить усі рядки, які є присутніми в таблиці R, але відсутні в таблиці S.

Різниця відношень в SQL має вигляд.

```
SELECT R.a1, R.a2
FROM R
WHERE NOT EXISTS
   (    (SELECT S.b1, S.b2
         FROM S
         WHERE S.b1=R.a2 AND S.b2=R.a1)
```

1.3 Обчислення і підведення підсумків в запитах

У загальному випадку для створення обчислюваного (похідного) поля в списку SELECT слід вказати деяке вираження мови SQL. У цих виразах застосовуються арифметичні операції складання, віднімання, множення і ділення, а також вбудовані функції мови SQL. Можна вказати ім'я будь-якого стовпця (поля) таблиці або запиту, але використовувати ім'я стовпця тільки тієї таблиці або запиту, які вказані в списку пропозиції FROM відповідної інструкції. При побудові складних виразів можуть знадобитися дужки.

Стандарти SQL дозволяють явним чином задавати імена стовпців результуючої таблиці, для чого застосовується фраза AS.

За допомогою підсумкових (агрегатних) функцій у рамках SQL-запиту можна отримати ряд узагальнювальних статистичних відомостей про множину відібраних значень вихідного набору.

Користувачеві доступні наступні основні підсумкові функції:

- Count (Вираження) – визначає кількість записів у вихідному наборі SQL-запиту;

- Min/Max (Вираження) – визначають найменше і найбільше з множини значень в деякому полі запиту;
- Avg (Вираження) – ця функція дозволяє розрахувати середнє значення множини значень, що зберігаються в певному полі відібраних запитом записів. Воно є арифметичним середнім значенням, тобто сумою значень, що ділиться на їх кількість.
- Sum (Вираження) – обчислює суму множини значень, що містяться в певному полі відібраних запитом записів.

Найчастіше вираженням виступають імена стовпців. Вираження може обчислюватися і по значеннях декількох таблиць.

Усі ці функції оперують зі значеннями в єдиному стовпці таблиці або з арифметичним вираженням і повертають єдине значення. Функції COUNT, MIN і MAX застосовані як до числових, так і до нечислових полів, тоді як функції SUM і AVG можуть використовуватися тільки у разі числових полів, за винятком COUNT(*). При обчисленні результатів будь-яких функцій спочатку виключаються усі порожні значення, після чого необхідна операція застосовується тільки до конкретних значень стовпця, що залишилися. Варіант COUNT(*) – особливий випадок використання функції COUNT, його призначення полягає в підрахунку усіх рядків в результуючій таблиці, незалежно від того, містяться там порожні, такі, що дублюються або будь-які інші значення.

Якщо до застосування узагальнювальної функції необхідно виключити значення, що дублюються, слід перед ім'ям стовпця у визначенні функції помістити ключове слово DISTINCT. Воно не має сенсу для функцій MIN і MAX, проте його використання може вплинути на результати виконання функцій SUM і AVG, тому необхідно заздалегідь обдумати, чи повинне воно бути присутнім у кожному конкретному випадку. Крім того, ключове слово DISTINCT може бути вказане у будь-якому запиті не більше одного разу.

Дуже важливо відмітити, що підсумкові функції можуть використовуватися тільки в списку пропозиції SELECT і у складі

пропозиції HAVING. У усіх інших випадках це неприпустимо. Якщо список в пропозиції SELECT містить підсумкові функції, а в тексті запиту відсутня фраза GROUP BY, що забезпечує об'єднання даних в групи, то жоден з елементів списку пропозиції SELECT не може включати яких-небудь посилань на поля, за винятком ситуації, коли поля виступають аргументами підсумкових функцій.

Часто в запитах вимагається формувати проміжні підсумки, що зазвичай відображається появою в запиті фрази «для кожного». Для цієї мети в операторові SELECT використовується пропозиція GROUP BY. Запит, в якому є присутнім GROUP BY, називається групуючим запитом, оскільки в нім групуються дані, отримані в результаті виконання операції SELECT, після чого для кожної окремої групи створюється єдиний сумарний рядок. Стандарт SQL вимагає, щоб пропозиція SELECT і фраза GROUP BY були тісно пов'язані між собою. За наявності в операторові SELECT фрази GROUP BY кожен елемент списку в пропозиції SELECT повинен мати єдине значення для усієї групи. Більше того, пропозиція SELECT може включати тільки наступні типи елементів : імена полів, підсумкові функції, константи і вирази, що включають комбінації перелічених вище елементів.

Усі імена полів, приведені в списку пропозиції SELECT, мають бути присутніми і у фразі GROUP BY за винятком випадків, коли ім'я стовпця використовується в підсумковій функції. Зворотне правило не є справедливим – у фразі GROUP BY можуть бути імена стовпців, відсутні в списку пропозиції SELECT.

Якщо спільно з GROUP BY використовується пропозиція WHERE, то воно обробляється першим, а групуванню піддаються тільки ті рядки, які задовольняють умові пошуку.

Стандартом SQL визначено, що при проведенні групування усі відсутні значення розглядаються як рівні. Якщо два рядки таблиці в одному і тому ж груповаому стовпці містять значення NULL і ідентичні

значення в усіх інших непорожніх групованих стовпцях, вони поміщаються в одну і ту ж групу.

За допомогою HAVING відбиваються усі попередньо згруповані за допомогою GROUP BY блоки даних, задовольняючі заданим в HAVING умовам. Це додаткова можливість "профільтрувати" вихідний набір.

Умови в HAVING відрізняються від умов в WHERE:

- HAVING виключає з результуючого набору даних групи з результатами агрегованих значень;
- WHERE виключає з розрахунку агрегатних значень по угрупуванню записи, що не задовольняють умові;
- у умові пошуку WHERE не можна задавати агрегатні функції.

Додатково до стандартних операторів GROUP BY та HAVING SQL Server підтримує ще чотири спеціальні розширення для групування даних: ROLLUP, CUBE, GROUPING SETS та OVER.

Оператор ROLLUP додає рядок, що підсумовує, в результуючий набір. При групуванні за кількома критеріями ROLLUP буде створювати підсумовуючий рядок для кожної з підгруп.

Оператор CUBE схожий на ROLLUP за тим винятком, що CUBE додає рядки, що підсумовують, для кожної комбінації груп.

Оператор GROUPING SETS аналогічно ROLLUP і CUBE додає сумуючий рядок для груп. Але при цьому він не включає самі групи. При цьому його можна поєднувати з ROLLUP або CUBE.

Вираз OVER дозволяє підсумовувати дані, при цьому повертаючи ті рядки, які використовувалися для отримання сумованих даних. Вираз OVER ставиться після агрегатної функції, потім у дужках йде вираз PARTITION BY і стовпець, за яким виконується угруповання.

1.4 Використання підзапитів

Часто неможливо вирішити поставлене завдання шляхом одного запиту. Це особливо актуально, коли при використанні умови пошуку в пропозиції WHERE значення, з яким потрібно порівнювати, заздалегідь не

визначене і має бути вичислене у момент виконання оператора SELECT. У такому разі приходять на допомогу закінчені оператори SELECT, впроваджені в тіло іншого оператора SELECT. Внутрішній підзапит є також оператор SELECT, а кодування його пропозицій підкоряється тим же правилам, що і основного оператора SELECT. Зовнішній оператор SELECT використовує результат виконання внутрішнього оператора для визначення змісту остаточного результату усієї операції. Внутрішні запити можуть бути поміщені безпосередньо після оператора порівняння (=,<=, >=, <>) в пропозиції WHERE і HAVING зовнішнього оператора SELECT - вони отримують назву підзапитів або вкладених запитів. Крім того, внутрішні оператори SELECT можуть застосовуватися в операторах INSERT, UPDATE і DELETE.

Підзапит – це інструмент створення тимчасової таблиці, вміст якої витягається і обробляється зовнішнім оператором. Текст підзапиту має бути поміщений в дужки. До підзапитів застосовуються наступні правила і обмеження :

- фраза ORDER BY не використовується, хоча і може бути присутньою в зовнішньому підзапиті ;
- список в пропозиції SELECT складається з імен окремих стовпців або складених з них виразів - за винятком випадку, коли в підзапиті є присутнім ключове слово EXISTS ;
- за умовчанням імена стовпців в підзапиті відносяться до таблиці, ім'я якої вказане в пропозиції FROM. Проте допускається посилання і на стовпці таблиці, вказаної у фразі FROM зовнішнього запиту, для чого застосовуються кваліфіковані імена стовпців (тобто з вказівкою таблиці);
- якщо підзапит є одним з двох операндів, що беруть участь в операції порівняння, то запит повинен вказуватися в правій частині цієї операції.

Існує таки типи підзапитів :

- скалярний підзапит повертає єдине значення. В принципі, він може використовуватися скрізь, де вимагається вказати єдине значення;
- рядковий підзапит повертає безліч значень, тобто значення одного стовпця таблиці, розміщені у більш ніж одному рядку;
- табличний підзапит повертає безліч значень, тобто значення одного або декількох стовпців таблиці, розміщені у більш ніж одному рядку. Він можливий скрізь, де допускається наявність таблиці.

У багатьох випадках підзапити, що використовуються у порівняннях в пропозиціях WHERE або HAVING, повертають не одно, а декілька значень. Вкладені підзапити генерують непойменоване проміжне відношення, тобто тимчасову таблицю. Таке проміжне відношення може використовуватися тільки в тому місці, де з'являється в підзапиті. До такого відношення неможливо звернутися по імені з якого-небудь іншого місця запиту. Застосовувані до підзапиту операції засновані на тих операціях, які, у свою чергу, застосовуються до множини, а саме:

- { WHERE | HAVING } вираження [NOT] IN (підзапит);
- { WHERE | HAVING } вираження оператор_порівняння { ALL | SOME | ANY } (підзапит);
- {WHERE | HAVING } [NOT] EXISTS (підзапит);

Оператор IN використовується для порівняння деякого значення із списком значень, при цьому перевіряється, чи входить значення в наданий список або порівнюване значення не є елементом представленого списку. Те ж саме стосується операторів ALL, SOME, ANY. Оператори IN, ALL, SOME, ANY зазвичай використовують для роботи з результатами рядкових підзапитів.

Оператор [NOT] EXISTS використовується для перевірки наявності змісту (тобто рядків) у таблиці, яка утворюється в результаті виконання підзапиту. Цей оператор зазвичай використовують для роботи з результатами табличних підзапитів.

Звичайно, що наведений вище опис оператора SELECT є стислим та спрощеним. Більш докладні відомості про особливості реалізації оператора SELECT у мові Transact-SQL у разі потреби можна переглянути, наприклад, на веб-ресурсі:

https://learn.microsoft.com/en-us/sql/t-sql/queries/select-transact-sql?view=sql-server-ver16

1.5 Створювані користувачем функції та вбудовані функції

При реалізації за допомогою мови SQL складних алгоритмів, які можуть знадобитися більше одного разу, відразу постає питання про збереження розробленого коду для подальшого застосування. Цю задачу можна було б реалізувати за допомогою збережених процедур, однак їх архітектура не дозволяє використовувати процедури безпосередньо у виразах, тому вони вимагають проміжного присвоєння повернутого значення змінної, яка потім і вказується в вираженні. Природно, подібний метод застосування програмного коду не дуже зручний. Багато розробників вже давно хотіли мати можливість виклику розроблених алгоритмів безпосередньо в виразах.

Функції користувача являють собою самостійні об'єкти бази даних, такі, наприклад, як збережені процедури або тригери. Функція користувача розташовується в певній базі даних і доступна тільки в її контексті.

У Microsoft SQL Server є наступні класи функцій користувача:

- Scalar (scalar-valued user-defined function) – скалярні функції повертають звичайне скалярне значення; кожна функція може включати безліч команд, що об'єднуються в один блок за допомогою конструкції BEGIN ... END;
- Inline (inline table-valued function) – табличні функції містять всього одну команду SELECT і повертають користувачеві набір даних у вигляді значення типу даних TABLE;
- Multi-statement (multi-statement table-valued function) – багатооператорні табличні функції також повертають користувачеві

значення типу даних TABLE, що містить набір даних, проте в тілі функції знаходиться безліч команд SQL (SELECT, INSERT, UPDATE і т.д.). Саме з їх допомогою і формується набір даних, який має бути повернуто після виконання функції.

Користувальницькі функції схожі з збереженими процедурами, але, на відміну від них, можуть застосовуватися в запитах так само, як і системні вбудовані функції. Користувальницькі функції, що повертають таблиці, можуть стати альтернативою представленням. Представлення обмежені одним виразом SELECT, а користувальницькі функції здатні включати додаткові вирази, що дозволяє створювати більш складні і потужні конструкції.

Створення та зміна функції скалярного типу (scalar-valued user-defined function) виконується за допомогою команди:

```
<Визначення_скаляр_функції> :: =
{CREATE | ALTER} FUNCTION [власник.]
   імя_функції
([{@ імя_параметра скаляр_тип_даних
   [= default]} [, ... n]])
RETURNS скаляр_тип_даних
[WITH {ENCRYPTION | SCHEMABINDING}
   [, .. . n]]
[AS]
BEGIN
<тіло_функції>
RETURN скаляр_вираз
END
```

Розглянемо призначення параметрів команди.

Функція може містити один або декілька вхідних параметрів або не містити жодного. Кожен параметр повинен мати унікальне в межах

створюваної функції ім'я і починатися з символу "@". Після імені вказується тип даних параметра. Додатково можна вказати значення, яке буде автоматично присвоюватися параметру (DEFAULT), якщо користувач явно не вказав значення відповідного параметра при виклику функції .

За допомогою конструкції RETURNS скаляр_тип_даних вказується, який тип даних буде мати значення, яке повертається функцією.

Додаткові параметри, з якими повинна бути створена функція, можуть бути зазначені за допомогою ключового слова WITH. Завдяки ключовим словом ENCRYPTION код команди, використовуваний для створення функції, буде зашифровано, і ніхто не зможе переглянути його. Ця можливість дозволяє приховати логіку роботи функції. Крім того, в тілі функції може виконуватися звернення до різних об'єктів бази даних, а тому зміна або видалення відповідних об'єктів може призвести до порушення роботи функції. Щоб уникнути цього, потрібно заборонити внесення змін, вказавши при створенні цієї функції ключове слово SCHEMABINDING.

Між ключовими словами BEGIN ... END вказується набір команд, вони й будуть тілом функції.

Коли в ході виконання коду функції зустрічається ключове слово RETURN, виконання функції завершується і як результат її обчислення повертається значення, вказане безпосередньо після слова RETURN. Відзначимо, що в тілі функції дозволяється використання безлічі команд RETURN, які можуть повертати різні значення. В якості значення, що повертається допускаються як звичайні константи, так і складні вирази. Єдина умова – тип даних значення, що повертається, повинен збігатися з типом даних, зазначених після ключового слова RETURNS.

Створення та зміна табличної функції (inline table-valued function) виконується за допомогою команди:

<Визначен_табл_функції> :: =
{CREATE | ALTER} FUNCTION [власник.]

```
   ім'я_функції
([{@ ім'я_параметра скаляр_тип_даних
   [= default]} [, ... n]])
RETURNS TABLE
[WITH {ENCRYPTION | SCHEMABINDING}
   [, .. . n]]
[AS]
RETURN [(] SELECT_оператор [)]
```

Основна частина параметрів, використовуваних при створенні табличних функцій, аналогічна параметрам скалярної функції. Проте створення табличних функцій має свою специфіку.

Після ключового слова RETURNS завжди повинно вказуватися ключове слово TABLE. Таким чином, функція даного типу повинна строго повертати значення типу даних TABLE. Структура значення, що повертається типом TABLE, не вказується явно при описі власне типу даних. Замість цього сервер буде автоматично використовувати для значення, що повертається TABLE, структуру, що повертається запитом SELECT, який є єдиною командою функції.

Особливість функції даного типу полягає в тому, що структура значення TABLE створюється автоматично в ході виконання запиту, а не вказується явно при визначенні типу після ключового слова RETURNS.

Значення типу TABLE, що повертається функцією, може бути використано безпосередньо у запиті, тобто в розділі FROM.

Створення та зміна функцій типу Multi-statement виконується за допомогою наступної команди:

```
<Визначення_мульти_функції> :: =
{CREATE | ALTER} FUNCTION [власник.]
   ім'я_функції
([{@ ім'я_параметра скаляр_тип_даних
```

```
   [= default]} [, ... n]])
RETURNS @ імя_параметра TABLE
   <визначення_таблиці>
[WITH {ENCRYPTION | SCHEMABINDING }
   [, ... n]]
[AS]
BEGIN
<тіло_функції>
RETURN
END
```

Використання більшої частини параметрів розглядалося при описі попередніх функцій.

Відзначимо, що функції даного типу, як і табличні, повертають значення типу TABLE. Однак, на відміну від табличних функцій, при створенні функцій Multi-statement необхідно явно задати структуру значення, що повертається. Вона зазначається безпосередньо після ключового слова TABLE і, таким чином, є частиною визначення типу даних, що повертається. Синтаксис конструкції <визначен_таблиці> повністю відповідає однойменним структурам, використовуваним при створенні звичайних таблиць за допомогою команди CREATE TABLE.

Набір даних, що повертається, повинен формуватися за допомогою команд INSERT, виконуваних в тілі функції. Крім того, в тілі функції допускається використання різних конструкцій мови SQL, які можуть контролювати значення, що розміщуються у вихідному наборі рядків. При роботі з командою INSERT потрібно явно вказати ім'я того об'єкта, куди необхідно вставити рядки. Тому в функціях типу Multi-statement, на відміну від табличних, необхідно присвоїти якесь ім'я об'єкту з типом даних TABLE – воно і вказується як значення, що повертається.

Завершення роботи функції відбувається у двох випадках: якщо виникають помилки виконання і якщо з'являється ключове слово RETURN.

На відміну від функцій скалярного типу, при використанні команди RETURN не потрібно вказувати значення, що повертається. Сервер автоматично поверне набір даних типу TABLE, ім'я та структура якого була вказана після ключового слова RETURNS. У тілі функції може бути вказано більше однієї команди RETURN.

Необхідно відзначити, що робота функції завершується тільки при наявності команди RETURN. Це твердження вірне і в тому випадку, коли мова йде про досягнення кінця тіла функції – самою останньою командою повинна бути команда RETURN.

Вбудовані функції, що наявні в розпорядженні користувачів при роботі з SQL, можна умовно розділити на наступні групи:

- математичні функції;
- рядкові функції;
- функції для роботи з датою і часом;
- функції конфігурування;
- функції системи безпеки;
- функції управління метаданими;
- статистичні функції.

Стислий перелік математичних функцій представлений в таблиці 1.1.

Таблиця 1.1 – Стислий перелік математичних функцій

Функція	Призначення функції
ABS	обчислює абсолютне значення числа
ACOS	обчислює арккосинус
ASIN	обчислює арксинус
ATAN	обчислює арктангенс
ATN2	обчислює арктангенс з урахуванням квадратів
CEILING	виконує округлення вгору
COS	обчислює косинус кута
COT	повертає котангенс кута
DEGREES	перетворить значення кута з радіан в градуси

Функція	Призначення функції
EXP	повертає експоненту
FLOOR	виконує округлення вниз
LOG	обчислює натуральний логарифм
LOG10	обчислює десятковий логарифм
PI	повертає значення "пі"
POWER	зводить число до ступеня
RADIANS	перетворить значення кута з градуса в радіани
RAND	повертає випадкове число
ROUND	виконує округлення із заданою точністю
SIGN	визначає знак числа
SIN	обчислює синус кута
SQUARE	виконує зведення числа в квадрат
SQRT	витягує квадратний корінь
TAN	повертає тангенс кута

Стислий перелік строкових функцій представлений в таблиці 1.2.

Таблиця 1.2 – Стислий перелік строкових функцій

Функція	Призначення функції
ASCII	повертає код ASCII лівого символу рядка
CHAR	за кодом ASCII повертає символ
CHARINDEX	визначає порядковий номер символу, з якого починається входження підрядка в рядок
DIFFERENCE	повертає показник збігу рядків
LEFT	повертає вказану кількість символів з початку рядка
LEN	повертає довжину рядка
LOWER	переводить всі символи рядка в нижній регістр
LTRIM	видаляє пробіли на початку рядка
NCHAR	повертає за кодом символ Unicode
PATINDEX	виконує пошук підрядка в рядку за вказаною шаблоном
REPLACE	замінює входження підрядка на вказане значення
QUOTENAME	конвертує рядок в формат Unicode
REPLICATE	виконує тиражування рядки певне число разів

Функція	Призначення функції
REVERSE	повертає рядок, символи якої записані у зворотному порядку
RIGHT	повертає вказану кількість символів з кінця рядка
RTRIM	видаляє пробіли в кінці рядка
SOUNDEX	повертає код звучання рядка
SPACE	повертає вказану кількість пробілів
STR	виконує конвертування значення числового типу в символьний формат
STUFF	видаляє вказане число символів, замінюючи новим підрядком
SUBSTRING	повертає для рядка підрядок зазначеної довжини з заданого символу
UNICODE	повертає Unicode-код лівого символу рядка
UPPER	переводить всі символи рядка в верхній регістр

Стислий перелік основних функцій для роботи з датою і часом представлений в таблиці 1.3.

Таблиця 1.3 – Стислий перелік основних функцій для роботи з датою і часом

Функція	Призначення функції
DATEADD	додає до дати зазначена значення днів, місяців, годин і т.д.
DATEDIFF	повертає різницю між зазначеними частинами двох дат
DATENAME	виділяє з дати вказану частину і повертає її в символьному форматі
DATEPART	виділяє з дати вказану частину і повертає її в числовому форматі
DAY	повертає число з вказаної дати
GETDATE	повертає поточний системний час
ISDATE	перевіряє правильність виразу на відповідність одному з можливих форматів вводу дати
MONTH	повертає значення місяця з вказаної дати
YEAR	повертає значення року з вказаної дати

З іншими типами вбудованих функцій у разі потреби можна ознайомитися, наприклад, на веб-ресурсі:

https://learn.microsoft.com

1.6 Збережені процедури

Збережені процедури є групами пов'язаних між собою операторів SQL, застосування яких робить роботу програміста легшою і гнучкішою, оскільки виконати процедуру, що зберігається, часто виявляється набагато простіше, ніж послідовність окремих операторів SQL. Збережені процедури є набором команд, що складається з одного або декількох операторів SQL або функцій і зберігається у базі даних у вигляді, що відкомпілювався. Виконання у базі даних процедур замість окремих операторів SQL, що зберігаються, дає користувачеві наступні переваги:

- необхідні оператори вже містяться у базі даних;
- усі вони пройшли етап синтаксичного аналізу і знаходяться у виконуваному форматі; перед виконанням збереженої процедури SQL Server генерує для неї план виконання, виконує її оптимізацію і компіляцію;
- збережені процедури підтримують модульне програмування, оскільки дозволяють розбивати великі завдання на самостійні, дрібніші і зручніші в управлінні частини;
- збережені процедури можуть викликати інші збережені процедури і функції;
- збережені процедури можуть бути викликані з прикладних програм інших типів;
- як правило, збережені процедури виконуються швидше, ніж послідовність окремих операторів;
- збережені процедури простіше використовувати: вони можуть складатися з десятків і сотень команд, але для їх запуску досить вказати усього лише ім'я потрібної збереженої процедури. Це дозволяє зменшити

розмір запиту, що посилається від клієнта на сервер, отже, і навантаження на мережу.

Зберігання процедур в тому ж місці, де вони виконуються, забезпечує зменшення об'єму даних, що передаються по мережі, і підвищує загальну продуктивність системи. Застосування процедур, що зберігаються, спрощує супровід програмних комплексів і внесення змін в них. Зазвичай усі обмеження цілісності у вигляді правил і алгоритмів обробки даних реалізуються на сервері баз даних і доступні кінцевому застосуванню у вигляді набору процедур, що зберігаються, які і представляють інтерфейс обробки даних. Для забезпечення цілісності даних, а також в цілях безпеки, застосунок зазвичай не дістає прямого доступу до даних – уся робота з ними ведеться шляхом виклику тих або інших процедур, що зберігаються.

Подібний підхід спрощує модифікацію алгоритмів обробки даних, які негайно стають доступними для усіх користувачів мережі, і забезпечує можливість розширення системи без внесення змін безпосередньо до застосунків – досить змінити процедуру, що зберігається, на сервері баз даних. Розробникові не треба перекомпілювати застосунок, створювати його копії, а також інструктувати користувачів про необхідність роботи з новою версією. Користувачі взагалі можуть не підозрювати про те, що в систему внесені зміни.

Збережені процедури, існують незалежно від таблиць або яких-небудь інших об'єктів баз даних. Вони викликаються клієнтською програмою, іншою процедурою, що зберігається, або тригером. Розробник може управляти правами доступу до збереженої процедури, дозволяючи або забороняючи її виконання. Змінювати код збереженої процедури, дозволяється тільки її власникові або членові фіксованої ролі бази даних. При необхідності можна передати права володіння нею від одного користувача до іншого.

При роботі з SQL Server користувачі можуть створювати власні процедури, що реалізовують ті або інші дії. Збережені процедури є

повноцінними об'єктами бази даних, а тому кожна з них зберігається в конкретній базі даних. Безпосередній виклик збереженої процедури можливий тільки якщо він здійснюється в контексті тієї бази даних, де знаходиться процедура.

У SQL Server є декілька типів процедур, що зберігаються.

- Системні збережені процедури, призначені для виконання різних адміністративних дій. Практично усі дії з адміністрування сервера виконуються з їх допомогою. Можна сказати, що системні Збережені процедури, є інтерфейсом, що забезпечує роботу з системними таблицями, яка, кінець кінцем, зводиться до зміни, додавання, видалення і вибірки даних з системних таблиць як призначених для користувача, так і системних баз даних. Системні збережені процедури мають префікс sp_, зберігаються в системній базі даних і можуть бути викликані в контексті будь-якої іншої бази даних.
- Призначені для користувача збережені процедури, реалізують ті або інші дії. Такі збережені процедури – повноцінний об'єкт бази даних. Внаслідок цього кожна процедура, що зберігається, розташовується в конкретній базі даних, де і виконується.
- Тимчасові збережені процедури, існують лише деякий час, після чого автоматично знищуються сервером. Вони діляться на локальні і глобальні. Локальні тимчасові збережені процедури, можуть бути викликані тільки з того з'єднання, в якому створені. При створенні такої процедури їй необхідно дати ім'я, що починається з одного символу #. Як і усі тимчасові об'єкти, процедури цього типу, що зберігаються, автоматично віддаляються при відключенні користувача, перезапуску або зупинці сервера. Глобальні тимчасові збережені процедури, доступні для будь-яких з'єднань сервера, на якому є така ж процедура. Для її визначення досить дати їй ім'я, що починається з символів ##. Видаляються ці процедури при перезапуску або зупинці сервера, а також при закритті з'єднання, в контексті якого вони були створені.

Створення збереженої процедури, передбачає вирішення наступних завдань:

- визначення типу створюваної збереженої процедури: тимчасова або призначена для користувача. Окрім цього, можна створити свою власну системну процедуру, що зберігається, призначивши їй ім'я з префіксом sp_ і помістивши її в системну базу даних. Така процедура буде доступна в контексті будь-якої бази даних локального сервера;
- планування прав доступу. При створенні збереженої процедури, слід враховувати, що вона матиме ті ж права доступу до об'єктів бази даних, що і користувач, що створив її;
- визначення параметрів збереженої процедури. Подібно до процедур, що входять до складу більшості мов програмування, Збережені процедури, можуть бути наділені вхідними і вихідними параметрами ;
- розробка коду збереженої процедури. Код процедури може містити послідовність будь-яких команд SQL, включаючи виклик інших процедур, що зберігаються.

Створення нової і зміна наявної збереженої процедури, здійснюється за допомогою наступної команди:

```
<визначення_процедури>::=
{CREATE | ALTER } [PROCEDURE] ім'я_процедури
      [;номер]
[{@ім'я_параметра тип_даних } [VARYING ]
            [=default][OUTPUT] ][,..n]
[WITH { RECOMPILE | ENCRYPTION | RECOMPILE
            ENCRYPTION }]
[FOR REPLICATION]
AS
            sql_оператор [..n]
```

Розглянемо параметри цієї команди.

Використовуючи префікси sp_, #, ##, створювану процедуру можна визначити в якості системної або тимчасової. Як видно з синтаксису команди, не допускається вказувати ім'я власника, якому належатиме створювана процедура, а також ім'я бази даних, де вона має бути розміщена. Так, щоб розмістити утворюючу збережену процедуру в конкретній базі даних, необхідно виконати команду CREATE PROCEDURE в контексті цієї бази даних. При зверненні з тіла збереженої процедури, до об'єктів тієї ж бази даних можна використовувати укорочені імена, тобто без вказівки імені бази даних. Коли ж вимагається звернутися до об'єктів, розташованих в інших базах даних, вказівка імені бази даних обов'язково.

Номер в імені – це ідентифікаційний номер збереженої процедури, що однозначно визначає її в групі процедур. Для зручності управління процедурами логічно однотипні збережені процедури можна групувати, привласнюючи їм однакові імена, але різні ідентифікаційні номери.

Для передачі вхідних і вихідних даних в створюваній процедурі, що зберігається, можуть використовуватися параметри, імена яких, як і імена локальних змінних, повинні починатися з символу @. В одній процедурі, що зберігається, можна задати безліч параметрів, розділених комами. У тілі процедури не повинні застосовуватися локальні змінні, чиї імена співпадають з іменами параметрів цієї процедури.

Для визначення типу даних, який матиме відповідний параметр збереженої процедури, годяться будь-які типи даних SQL, включаючи визначені користувачем. Проте тип даних CURSOR може бути використаний тільки як вихідний параметр збереженої процедури, тобто з вказівкою ключового слова OUTPUT.

Наявність ключового слова OUTPUT означає, що відповідний параметр призначений для повернення даних з збереженої процедури. Проте це зовсім не означає, що параметр не підходить для передачі значень в процедуру, що зберігається. Вказівка ключового слова OUTPUT наказує серверу при виході з збереженої процедури, присвоїти поточне

значення параметра локальної змінної, яка була вказана при виклику процедури в якості значення параметра. Відмітимо, що при вказівці ключового слова OUTPUT значення відповідного параметра при виклику процедури може бути задане тільки за допомогою локальної змінної. Не дозволяється використання будь-яких виразів або констант, допустиме для звичайних параметрів.

Ключове слово VARYING застосовується спільно з параметром OUTPUT, що має тип CURSOR. Воно визначає, що вихідним параметром буде результуюча множина.

Ключове слово DEFAULT є значенням, яке прийматиме відповідний параметр за умовчанням. Таким чином, при виклику процедури можна не вказувати явно значення відповідного параметра.

Оскільки сервер кешує план виконання запиту і компілює код, при наступному виклику процедури використовуватимуться вже готові значення. Проте в деяких випадках все ж вимагається виконувати перекомпіляцію коду процедури. Вказівка ключового слова RECOMPILE наказує системі створювати план виконання збереженої процедури, при кожному її виклику.

Параметр FOR REPLICATION затребуваний при реплікації даних і включенні створюваної збереженої процедури, в якості статті в публікацію.

Ключове слово ENCRYPTION наказує серверу виконати шифрування коду збереженої процедури, що може забезпечити захист від використання авторських алгоритмів, що реалізовують роботу збереженої процедури.

Ключове слово AS розміщується на початку власне тіла збереженої процедури, тобто набору команд SQL, за допомогою яких і реалізовуватиметься те або інша дія. У тілі процедури можуть застосовуватися практично усі команди SQL, оголошуватися транзакції, встановлюватися блокування і викликатися інші Збережені процедури.

Вихід зі збереженої процедури можна здійснити за допомогою команди RETURN.

Видалення збереженої процедури, здійснюється командою:

```
DROP PROCEDURE {ім'я_процедури} [,..n]
```

Для виконання збереженої процедури, використовується команда:

```
[[ EXEC [ UTE] ім'я_процедури [;номер]
[[@ім'я_параметра=]{значення | @ім'я_змінної}
        [OUTPUT ]|[DEFAULT ]][,..n]
```

Якщо виклик збереженої процедури, не є єдиною командою в пакеті, то присутність команди EXECUTE обов'язкова. Більше того, цю команду потрібно для виклику процедури з тіла іншої процедури або тригера.

Використання ключового слова OUTPUT при виклику процедури дозволяється тільки для параметрів, які були оголошені при створенні процедури з ключовим словом OUTPUT.

Коли ж при виклику процедури для параметра вказується ключове слово DEFAULT, то буде використано значення за умовчанням. Природно, вказане слово DEFAULT дозволяється тільки для тих параметрів, для яких визначено значення за умовчанням.

З синтаксису команди EXECUTE видно, що імена параметрів можуть бути опущені при виклику процедури. Проте в цьому випадку користувач повинен вказувати значення для параметрів в тому ж порядку, в якому вони перераховувалися при створенні процедури. Присвоїти параметру значення за умовчанням, просто пропустивши його при перерахуванні не можна. Якщо ж вимагається опустити параметри, для яких визначено значення за умовчанням, досить явної вказівки імен параметрів при виклику збереженої процедури. Більше того, у такий спосіб можна перераховувати параметри і їх значення в довільному порядку.

Також слід зазначити, що при виклику процедури вказуються або імена параметрів зі значеннями, або тільки значення без імені параметра. Їх комбінування не допускається.

2 ВИКОНАННЯ РОБОТИ

При виконанні лабораторної роботи передбачається використання застосунку SQL Server Management Studio, який входить до складу інтегрованої платформи Microsoft SQL Server. При цьому передбачається використання СУБД Microsoft SQL Server версії 2012 або вищої. У зв'язку із цим елементи інтерфейсу можуть мати відмінності порівняно із тими, що наведено далі у вигляді рисунків (особливо у випадку використання версії, локалізованої для використання певної національної мови). Ці відмінності не є принциповими та не впливають на виконання роботи та отримані результати.

Для початку виконання роботи треба підключити базу даних (SupplySalesDB), яку було створено при виконанні попередньої лабораторної роботи (тобто, лабораторної роботи за темою «Створення оперативної бази даних (OLTP) на прикладі використання СУБД Microsoft SQL Server»).

Увага! Всі запити, що розглядаються далі, повинні бути результативними (тобто в результаті виконання запиту повинні бути виведені один або кілька записів). Відсутність результату запиту є ознакою помилок під час побудови запиту, невідповідності запиту наявним даним тощо. Такий запит потребує аналізу та перевірки. Також передбачається, що результати запитів ґрунтуються на тих даних, які були введені до бази даних при виконанні попередньої лабораторної роботи.

2.1 Побудова та використання створюваних користувачем функцій

Для отримання доступу до переліку функцій, потрібно у списку об'єктів бази даних відкрити пункт Programmability і в ньому відкрити пункт Functions (рисунок 2.1). Потім треба відкрити пункт, що відповідає певному типу функцій.

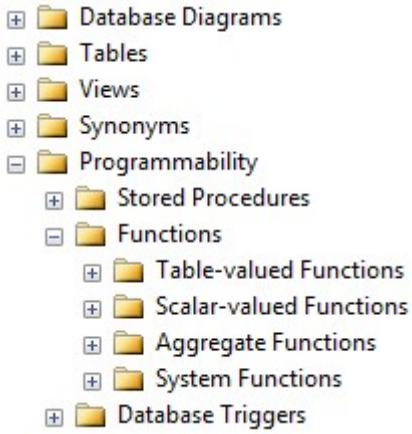

Рисунок 2.1

Розглянемо послідовність дій при створенні та використанні багатооператорної табличної функції (multi-statement table-valued function).

Призначення функції: формування довідника товарів із зазначенням повної назви товарів у вигляді текстової стрічки, до якої входять усі складові назви групи товарів та назва товару.

Для створення функції треба.

1. Клацнути правою кнопкою миші по пункту Table-valued Functions і в меню вибрати пункт New Multi-statement Table-valued Function…. В результаті буде створено запит, що містить «заготівку» функції.

2. Замість «заготівки» ввести текст запиту, наведений на рисунку 2.2. При цьому зберігати коментарі, які залишилися від «заготівки» (вони позначені зеленим кольором), не обов'язково.

3. Створити функцію, натиснувши кнопку Execute. У разі, якщо в тексті запиту немає помилок, буде створено функцію. Переглянути список табличних функцій та пересвідчитися, що функцію було успішно створено.

4. Перевірити працездатність функції. Для цього створити запит, який наведено на рисунку 2.3.

5. Виконати запит. Результат виконання наведений на рисунку 2.4. Слід звернути увагу на те, що перша складова назви групи товарів відокремлюється від загальної назви комою.

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE FUNCTION [dbo].[FPGN] ()
--FullProductGroupName
Returns @retFPGN TABLE
(
   FullName VARCHAR (MAX) NOT NULL,
   id int NOT NULL,
   ProductID int primary key NOT NULL,
   UnitID int NOT NULL
)
AS
BEGIN
WITH FullProductGroupName ( data , id , pid , level , FullName )
AS (
     SELECT t.ProductGroupName, t.ProductGroupID, t.PrProductGroupID, 0, TRIM(CAST ( t.ProductGroupName AS VARCHAR (MAX) ))
     FROM ProductGroups t
     WHERE t.ProductGroupID = 1
     UNION ALL
     SELECT t.ProductGroupName, t.ProductGroupID, t.PrProductGroupID, level + 1 ,
               TRIM(CAST ( tr.FullName + ' ' + TRIM(t.ProductGroupName) + IIF(t.PrProductGroupID=1,', ',' ') AS VARCHAR(MAX) ))
     FROM ProductGroups t INNER JOIN FullProductGroupName tr ON ( t.PrProductGroupID = tr.id )
   )
    INSERT @retFPGN
    SELECT tr.FullName+' '+RTRIM(ProductName),tr.id,ProductID,MeasurementUnitID FROM FullProductGroupName tr
    INNER JOIN Products ON tr.id=ProductGroupID
    RETURN
END
```

Рисунок 2.2

```
USE SupplySalesDB

SELECT * FROM FPGN()
```

Рисунок 2.3

	FullName	id	ProductID	UnitID
1	Бакалія, Цукор	2	1	1
2	Бакалія, Сіль Екстра	6	2	1
3	Бакалія, Кислота оцтова 9% ЩебПак	7	3	2
4	Бакалія, Крохмаль картопляний	8	4	1
5	Бакалія, Лист лавровий	2	5	1
6	Бакалія, Перець чорний мелений Мрія	11	6	3
7	Консерви, рибні "Кілька у томаті"	12	7	4
8	Бакалія, Кава в зернах Арабіка	14	8	1
9	Кондитерські вироби, Цукерки шоколадні "Білочка"	16	9	1
10	Консерви, рибні "Бички у томаті"	12	10	4
11	Кондитерські вироби, Цукерки льодяники "Барбарис"	17	11	1
12	Алкогольні напої, Коньяк "Десна"	19	12	4
13	Алкогольні напої, Горілка "Nemiroff"	20	13	4
14	Алкогольні напої, Вино солодке "Кагор"	22	14	4

Рисунок 2.4

Розглянемо послідовність дій при створенні та використанні табличної функції (inline table-valued function).

Призначення функції: розрахунок доходу по кожному виду товарів.

Увага! У даному випадку дохід розраховується дуже спрощено, як різниця між вартістю продажу та вартістю закупки.

Для створення функції треба клацнути правою кнопкою миші по пункту Table-valued Functions і в меню вибрати пункт New Inline Table-valued Function…. Послідовність дій при створенні та використанні такої функції аналогічна розглянутим вище діям при створенні попередньої функції.

Текст запиту для створення функції наведений на рисунку 2.5. Текст запиту для перевірки роботи функції наведений на рисунку 2.6. Результат роботи функції наведений на рисунку 2.7.

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

CREATE FUNCTION [dbo].[yield_product_calc]
(
)
RETURNS TABLE
AS
RETURN
(
SELECT FullName AS ProductName,SUM(SoldAmount*(SoldPrice-PurchasedPrice)) AS yield
       FROM Sold INNER JOIN Purchased ON Sold.ProductID = Purchased.ProductID AND Sold.PurchasingID = Purchased.PurchasingID
                 INNER JOIN FPGN() f ON Sold.ProductID = f.ProductID
GROUP BY FullName
)
```

Рисунок 2.5

```
USE SupplySalesDB
SELECT * FROM yield_product_calc()
```

Рисунок 2.6

	ProductName	yield
1	Алкогольні напої, Вино солодке "Кагор"	134390.00
2	Алкогольні напої, Горілка "Nemiroff"	50400.00
3	Алкогольні напої, Коньяк "Десна"	14428.00
4	Бакалія, Кава в зернах Арабіка	21971.00
5	Бакалія, Кислота оцтова 9% ЩебПак	6348.00
6	Бакалія, Крохмаль картопляний	11762.50
7	Бакалія, Лист лавровий	9085.50
8	Бакалія, Перець чорний мелений Мрія	6230.00
9	Бакалія, Сіль Екстра	3334.00
10	Бакалія, Цукор	8340.00
11	Кондитерські вироби, Цукерки льодяники "Барбарис"	18882.50
12	Кондитерські вироби, Цукерки шоколадні "Білочка"	45375.00
13	Консерви, рибні "Бички у томаті"	18350.00
14	Консерви, рибні "Кілька у томаті"	3062.50

Рисунок 2.7

Розглянемо послідовність дій при створенні та використанні табличної функції (inline table-valued function).

Призначення функції: розрахунок доходу по кожному виду товарів з урахуванням дати продажу.

Увага! У даному випадку дохід розраховується дуже спрощено, як різниця між вартістю продажу та вартістю закупки.

Для створення функції треба клацнути правою кнопкою миші по пункту Table-valued Functions і в меню вибрати пункт New Inline Table-valued Function…. Послідовність дій при створенні та використанні такої функції аналогічна розглянутим вище.

Текст запиту для створення функції наведений на рисунку 2.8. Текст запиту для перевірки роботи функції наведений на рисунку 2.9. Фрагмент результату роботи функції наведений на рисунку 2.10. Загалом функція повертає 87 рядків.

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

CREATE FUNCTION [dbo].[yield_product_date_calc]
(
)
RETURNS TABLE
AS
RETURN
(
SELECT FullName AS ProductName,SaleDate,SUM(SoldAmount) AS Amount,SUM(SoldAmount*(SoldPrice-PurchasedPrice)) AS yield
       FROM Sold INNER JOIN Purchased ON Sold.ProductID = Purchased.ProductID AND Sold.PurchasingID = Purchased.PurchasingID
                 INNER JOIN FPGN() f ON Sold.ProductID = f.ProductID
                 INNER JOIN Sales ON Sold.SaleID=Sales.SaleID
GROUP BY FullName,SaleDate
)
```

Рисунок 2.8

```
USE SupplySalesDB
SELECT * FROM yield_product_date_calc()
```

Рисунок 2.9

	ProductName	SaleDate	Amount	yield
1	Бакалія, Кава в зернах Арабіка	2002-09-05 00:00:00.000	100	390.00
2	Бакалія, Кислота оцтова 9% ЩебПак	2002-09-05 00:00:00.000	100	170.00
3	Бакалія, Крохмаль картопляний	2002-09-05 00:00:00.000	250	462.50
4	Бакалія, Лист лавровий	2002-09-05 00:00:00.000	100	990.00
5	Бакалія, Сіль Екстра	2002-09-05 00:00:00.000	20	14.00
6	Консерви, рибні "Бички у томаті"	2002-09-05 00:00:00.000	500	750.00
7	Бакалія, Кислота оцтова 9% ЩебПак	2002-09-15 00:00:00.000	50	70.00
8	Бакалія, Сіль Екстра	2002-09-15 00:00:00.000	30	22.50
9	Бакалія, Кава в зернах Арабіка	2002-09-16 00:00:00.000	80	312.00
10	Бакалія, Цукор	2002-09-16 00:00:00.000	350	537.50
11	Бакалія, Лист лавровий	2002-09-17 00:00:00.000	250	662.50
12	Бакалія, Сіль Екстра	2002-09-17 00:00:00.000	20	24.00
13	Бакалія, Цукор	2002-09-17 00:00:00.000	200	260.00
14	Бакалія, Цукор	2002-09-25 00:00:00.000	600	900.00
15	Бакалія, Кава в зернах Арабіка	2002-10-25 00:00:00.000	180	387.00
16	Бакалія, Кислота оцтова 9% ЩебПак	2002-10-25 00:00:00.000	200	400.00
17	Бакалія, Крохмаль картопляний	2002-10-25 00:00:00.000	550	1100.00
18	Бакалія, Лист лавровий	2002-10-25 00:00:00.000	380	1623.00
19	Бакалія, Сіль Екстра	2002-10-25 00:00:00.000	180	163.50
20	Бакалія, Цукор	2002-10-25 00:00:00.000	500	750.00
21	Алкогольні напої, Вино солодке "Кагор"	2002-11-25 00:00:00.000	400	944.00
22	Алкогольні напої, Коньяк "Десна"	2002-11-25 00:00:00.000	950	3003.00
23	Бакалія, Кава в зернах Арабіка	2002-11-25 00:00:00.000	120	282.00
24	Бакалія, Кислота оцтова 9% ЩебПак	2002-11-25 00:00:00.000	250	300.00

Рисунок 2.10

Розглянемо послідовність дій при створенні та використанні табличної функції (inline table-valued function).

Призначення функції: розрахунок доходу по кожному факту продажу товарів.

Увага! У даному випадку дохід розраховується дуже спрощено, як різниця між вартістю продажу та вартістю закупки.

Для створення функції треба клацнути правою кнопкою миші по пункту Table-valued Functions і в меню вибрати пункт New Inline Table-valued Function…. Послідовність дій при створенні та використанні такої функції аналогічна розглянутим вище.

Текст запиту для створення функції наведений на рисунку 2.11. Текст запиту для перевірки роботи функції наведений на рисунку 2.12. Фрагмент результату роботи функції наведений на рисунку 2.13. Загалом функція повертає 55 рядків.

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

CREATE FUNCTION yield_sales_calc
(
)
RETURNS TABLE
AS
RETURN
(
SELECT Sales.SaleID,Sales.SaleDate,CustomerName,SUM(SoldAmount*(SoldPrice-PurchasedPrice)) AS yield
       FROM Sold INNER JOIN Sales ON Sold.SaleID = Sales.SaleID
                 INNER JOIN Purchased ON Sold.ProductID = Purchased.ProductID AND Sold.PurchasingID = Purchased.PurchasingID
                 INNER JOIN Customers ON Customers.CustomerID = Sales.CustomerID
GROUP BY Sales.SaleID,Sales.SaleDate,CustomerName
)
GO
```

Рисунок 2.11

```
use SupplySalesDB
select * from yield_sales_calc()
```

Рисунок 2.12

	SaleID	SaleDate	CustomerName	yield
1	5	2002-09-25 00:00:00.000	ВАТ "Перемога"	900.00
2	8	2002-10-25 00:00:00.000	ВАТ "Перемога"	172.00
3	15	2002-11-25 00:00:00.000	ВАТ "Перемога"	597.00
4	20	2002-12-25 00:00:00.000	ВАТ "Перемога"	1300.00
5	25	2003-01-25 00:00:00.000	ВАТ "Перемога"	11885.00
6	30	2003-02-23 00:00:00.000	ВАТ "Перемога"	6885.00
7	35	2003-03-25 00:00:00.000	ВАТ "Перемога"	7223.00
8	40	2003-04-25 00:00:00.000	ВАТ "Перемога"	29587.00
9	45	2003-05-25 00:00:00.000	ВАТ "Перемога"	8000.00
10	50	2003-06-25 00:00:00.000	ВАТ "Перемога"	8610.00
11	55	2003-07-25 00:00:00.000	ВАТ "Перемога"	13120.00
12	4	2002-09-17 00:00:00.000	ЗАТ "Вулкан"	946.50
13	6	2002-10-25 00:00:00.000	ЗАТ "Вулкан"	272.50
14	11	2002-11-25 00:00:00.000	ЗАТ "Вулкан"	260.00
15	19	2002-12-25 00:00:00.000	ЗАТ "Вулкан"	900.00
16	24	2003-01-25 00:00:00.000	ЗАТ "Вулкан"	260.00
17	29	2003-02-22 00:00:00.000	ЗАТ "Вулкан"	11610.50
18	34	2003-03-25 00:00:00.000	ЗАТ "Вулкан"	4489.50

Рисунок 2.13

Розглянемо послідовність дій при створенні та використанні табличної функції (inline table-valued function).

Призначення функції: розрахунок залишків по кожному виду товарів, тобто запасів товарів як різниці між кількістю одиниць закуплених товарів та кількістю одиниць проданих товарів.

Для створення функції треба клацнути правою кнопкою миші по пункту Table-valued Functions і в меню вибрати пункт New Inline Table-valued Function…. Послідовність дій при створенні та використанні такої функції аналогічна розглянутим вище.

Текст запиту для створення функції наведений на рисунку 2.14. Текст запиту для перевірки роботи функції наведений на рисунку 2.15. Результат роботи функції наведений на рисунку 2.16.

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

CREATE FUNCTION [dbo].[product_inventory]()
RETURNS TABLE
AS
RETURN
(
SELECT FullName AS ProductName,Purchased.ProductID,
       SUM(PurchasedAmount) AS PurchasedAmount,
       (SELECT SUM(SoldAmount) FROM sold WHERE Purchased.ProductID = Sold.ProductID) AS SoldAmount,
       SUM(PurchasedAmount)-(SELECT SUM(SoldAmount) FROM Sold WHERE Purchased.ProductID = Sold.ProductID) AS inventory
       FROM Purchased  INNER JOIN FPGN() f ON Purchased.ProductID = f.ProductID
                                GROUP BY FullName,Purchased.ProductID
)
```

Рисунок 2.14

```
USE SupplySalesDB
SELECT * FROM product_inventory()
```

Рисунок 2.15

	ProductName	ProductID	PurchasedAmount	SoldAmount	inventory
1	Бакалія, Цукор	1	5150	5150	0
2	Бакалія, Сіль Екстра	2	2705	2650	55
3	Бакалія, Кислота оцтова 9% ЩебПак	3	4760	4760	0
4	Бакалія, Крохмаль картопляний	4	3800	3800	0
5	Бакалія, Лист лавровий	5	2430	2430	0
6	Бакалія, Перець чорний мелений Мрія	6	1780	1780	0
7	Консерви, рибні "Кілька у томаті"	7	1250	1250	0
8	Бакалія, Кава в зернах Арабіка	8	5080	5080	0
9	Кондитерські вироби, Цукерки шоколадні "Білочка"	9	5500	5500	0
10	Консерви, рибні "Бички у томаті"	10	3700	3700	0
11	Кондитерські вироби, Цукерки льодяники "Барбарис"	11	4150	4150	0
12	Алкогольні напої, Коньяк "Десна"	12	2450	2450	0
13	Алкогольні напої, Горілка "Nemiroff"	13	6300	6300	0
14	Алкогольні напої, Вино солодке "Кагор"	14	17980	17980	0

Рисунок 2.16

Розглянемо послідовність дій при створенні та використанні табличної функції (inline table-valued function).

Призначення функції: розрахунок по кожному виду товарів фактичного об’єму продажів за вказаний рік у кількісному та вартісному значеннях та співставлення фактичного об’єму продажів із запланованим. Вказаний рік повинен передаватися до функції як параметр.

Увага! У даному випадку запланований об'єм продажів розраховується дуже спрощено. Робиться припущення, що планові показники розраховуються на основі об'єму продажів за рік, який передує вказаному, збільшеному на 10%.

Для створення функції треба клацнути правою кнопкою миші по пункту Table-valued Functions і в меню вибрати пункт New Inline Table-valued Function…. Послідовність дій при створенні та використанні такої функції аналогічна розглянутим вище.

Текст запиту для створення функції наведений на рисунку 2.17. Текст запиту для перевірки роботи функції наведений на рисунку 2.18. Результат роботи функції наведений на рисунку 2.19.

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

CREATE FUNCTION [dbo].[product_sales_plan] (@thisYear integer)

RETURNS TABLE
AS
RETURN
(
SELECT FullName AS ProductName,
       f.ProductID,
       ISNULL(SUM(SoldAmount),0) AS thisYearSoldAmount,
       ISNULL(SUM(SoldAmount*SoldPrice),0) AS thisYearSoldCost,
       ISNULL((SELECT SUM(SoldAmount) FROM Sold INNER JOIN Sales ON Sold.SaleID=Sales.SaleID
                        WHERE Sold.ProductID=f.ProductID AND YEAR(Sales.SaleDate)=@thisYear-1),0)*1.1 AS PlanSoldAmount,
       ISNULL((SELECT SUM(SoldAmount*SoldPrice) FROM Sold INNER JOIN Sales ON Sold.SaleID=Sales.SaleID
                        WHERE Sold.ProductID=f.ProductID AND YEAR(Sales.SaleDate)=@thisYear-1),0)*1.1 AS PlanSoldCost
        FROM FPGN() f LEFT JOIN Sold on Sold.ProductID = f.ProductID
                      INNER JOIN Sales ON Sold.SaleID=Sales.SaleID
        WHERE YEAR(Sales.SaleDate)=@thisYear
        GROUP BY FullName,f.ProductID
)
```

Рисунок 2.17

```
USE SupplySalesDB
SELECT * FROM product_sales_plan(2003)
```

Рисунок 2.18

	ProductName	ProductID	thisYearSoldAmount	thisYearSoldCost	PlanSoldAmount	PlanSoldCost
1	Бакалія, Цукор	1	2000	14492.50	3465.0	14949.000
2	Бакалія, Сіль Екстра	2	2300	7590.00	385.0	460.900
3	Бакалія, Кислота оцтова 9% ЩебПак	3	3160	16162.00	1760.0	9306.000
4	Бакалія, Крохмаль картопляний	4	3000	32850.00	880.0	8041.000
5	Бакалія, Лист лавровий	5	1500	28350.00	1023.0	19503.000
6	Бакалія, Перець чорний мелений Мрія	6	1780	15664.00	0.0	0.000
7	Консерви, рибні "Кілька у томаті"	7	1250	22500.00	0.0	0.000
8	Бакалія, Кава в зернах Арабіка	8	4000	115600.00	1188.0	30363.300
9	Кондитерські вироби, Цукерки шоколадні "Білочка"	9	5500	99000.00	0.0	0.000
10	Консерви, рибні "Бички у томаті"	10	3200	89600.00	550.0	12650.000
11	Кондитерські вироби, Цукерки льодяники "Барбарис"	11	4150	116200.00	0.0	0.000
12	Алкогольні напої, Коньяк "Десна"	12	1250	60375.00	1320.0	29056.500
13	Алкогольні напої, Горілка "Nemiroff"	13	6300	459900.00	0.0	0.000
14	Алкогольні напої, Вино солодке "Кагор"	14	17000	671900.00	1078.0	27528.600

Рисунок 2.19

Розглянемо послідовність дій при створенні та використанні табличної функції (inline table-valued function).

Призначення функції: сформувати узагальнені дані про закупівлі товарів у постачальників, а саме: назва групи товарів та конкретного товару (бажано, окремо), назва постачальника, загальна вартість закупки, питома вага (наприклад, у відсотках) цього товару у загальній вартості закупок у цього постачальника.

Текст запиту для створення функції наведений на рисунку 2.20. Текст запиту для перевірки роботи функції наведений на рисунку 2.21. Результат роботи функції у вигляді початкового та завершального фрагменту набору даних наведений на рисунку 2.22. Дані щодо питомої ваги наведено у вигляді відсотку та дробу. Також слід звернути увагу на використання спеціального розширення для групування даних – оператора OVER.

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

CREATE FUNCTION PurchasedProductsSpecificWeght
(
)
RETURNS TABLE
AS
RETURN
(
SELECT MainProductGroup, ProductName, SupplierName, PurchasedCost,
       SUM(PurchasedCost) OVER (PARTITION BY SupplierName) AS ThisSupplierTotalPurchasedCost,
       CAST(ROUND(PurchasedCost/SUM(PurchasedCost) OVER (PARTITION BY SupplierName),2)*100 AS int) AS ProcentOfPurchased,
       CAST(PurchasedCost/SUM(PurchasedCost) OVER (PARTITION BY SupplierName) AS decimal(8,5)) AS PartOfPurchased
       FROM
       (
       SELECT SUBSTRING(FullName,1,CHARINDEX(',',FullName)-1) AS MainProductGroup,
       SUBSTRING(FullName,CHARINDEX(',',FullName)+2,LEN(FullName)-CHARINDEX(',',FullName)) AS ProductName,
       SupplierName,
       SUM(PurchasedAmount*PurchasedPrice) AS PurchasedCost
              FROM Purchased INNER JOIN Purchasing ON Purchasing.PurchasingID = Purchased.PurchasingID
                     INNER JOIN Suppliers ON Purchasing.SupplierID=Suppliers.SupplierID
                     INNER JOIN FPGN() f ON Purchased.ProductID = f.ProductID
       GROUP BY SUBSTRING(FullName,1,CHARINDEX(',',FullName)-1),
                SUBSTRING(FullName,CHARINDEX(',',FullName)+2,LEN(FullName)-CHARINDEX(',',FullName)),
                SupplierName
       ) pp
)
GO
```

Рисунок 2.20

```
USE SupplySalesDB
SELECT * FROM PurchasedProductsSpecificWeght()
```

Рисунок 2.21

	MainProductGroup	ProductName	SupplierName	PurchasedCost	ThisSupplierTotalPurchasedCost	ProcentOfPurchased	PartOfPurchased
1	Бакалія	Крохмаль картопляний	ВАТ "Граніт"	1480.00	5257.50	28	0.28150
2	Бакалія	Лист лавровий	ВАТ "Граніт"	3140.00	5257.50	60	0.59724
3	Бакалія	Цукор	ВАТ "Граніт"	637.50	5257.50	12	0.12126
4	Алкогольні напої	Вино солодке "Кагор"	ЗАТ "Варіант"	9280.00	297930.00	3	0.03115
5	Алкогольні напої	Горілка "Nemiroff"	ЗАТ "Варіант"	260000.00	297930.00	87	0.87269
6	Алкогольні напої	Коньяк "Десна"	ЗАТ "Варіант"	15000.00	297930.00	5	0.05035
7	Бакалія	Кава в зернах Арабіка	ЗАТ "Варіант"	13650.00	297930.00	5	0.04582
8	Бакалія	Кава в зернах Арабіка	ПП Афанасьєв	3180.00	28718.00	11	0.11073
9	Алкогольні напої	Коньяк "Десна"	ПП Афанасьєв	7912.00	28718.00	28	0.27551
10	Алкогольні напої	Вино солодке "Кагор"	ПП Афанасьєв	12656.00	28718.00	44	0.44070
11	Бакалія	Цукор	ПП Афанасьєв	2750.00	28718.00	10	0.09576
…							
33	Консерви	рибні "Бички у томаті"	ТОВ "Маяк"	72000.00	715074.00	10	0.10069
34	Бакалія	Перець чорний меле...	ТОВ "Маяк"	9434.00	715074.00	1	0.01319
35	Консерви	рибні "Кілька у томаті"	ТОВ "Маяк"	19437.50	715074.00	3	0.02718
36	Консерви	рибні "Бички у томаті"	ТОВ "Схід"	10750.00	282588.50	4	0.03804
37	Бакалія	Крохмаль картопляний	ТОВ "Схід"	3537.50	282588.50	1	0.01252
38	Бакалія	Лист лавровий	ТОВ "Схід"	6857.50	282588.50	2	0.02427
39	Бакалія	Сіль Екстра	ТОВ "Схід"	21.50	282588.50	0	0.00008
40	Бакалія	Цукор	ТОВ "Схід"	780.00	282588.50	0	0.00276
41	Алкогольні напої	Вино солодке "Кагор"	ТОВ "Схід"	254400.00	282588.50	90	0.90025
42	Бакалія	Кава в зернах Арабіка	ТОВ "Схід"	4872.00	282588.50	2	0.01724
43	Бакалія	Кислота оцтова 9% Щ...	ТОВ "Схід"	1370.00	282588.50	0	0.00485

Рисунок 2.22

Наведені вище приклади функцій є досить простими. Більш детальну інформацію щодо побудови та використання створюваних користувачем функцій засобами мови Transact-SQL, можна, наприклад, отримати за посиланням:

https://learn.microsoft.com/en-us/sql/t-sql/statements/create-function-transact-sql?view=sql-server-ver16

2.2 Побудова та виконання запитів SELECT-SQL

Розглянемо послідовність дій при створенні та виконанні першого запиту.

Мета запиту: вивести на екран список контрагентів (тобто постачальників або покупців), які знаходяться у населеному пункті «Полтава» (код населеного пункту – 5). Дані, які повинні бути виведені: код, назва, адреса контрагента, назва населеного пункту, назва регіону, статус контрагента (тобто, постачальник або покупець).

Для створення та виконання запиту треба.

1. Натиснути кнопку New Query на панелі інструментів.
2. Ввести текст запиту, наведений рисунку 2.23.

```
USE SupplySalesDB

SELECT SupplierID AS ID,SupplierName AS Name,SupplierAddress AS Address,
       CityName,RegionName,'постачальник' AS Status
     FROM Suppliers INNER JOIN Cities ON  Cities.CityID=Suppliers.CityID
                    INNER JOIN Regions ON Cities.RegionID=Regions.RegionID
     WHERE Suppliers.CityID=5
UNION
SELECT CustomerID AS ID,CustomerName AS Name,CustomerAddress AS Address,
       CityName,RegionName,'покупець' AS Status
     FROM Customers INNER JOIN Cities ON  Cities.CityID=Customers.CityID
                    INNER JOIN Regions ON Cities.RegionID=Regions.RegionID
     WHERE Customers.CityID=5
```

Рисунок 2.23

3. Натиснути кнопку Execute. У разі, якщо в тексті запиту немає помилок, буде виведено результат запиту. Цей результат має вигляд, наведений на рисунку 2.24.

	ID	Name	Address	CityName	RegionName	Status
1	1	ТОВ "Схід"	вул. Лісова, 25	Полтава	Полтавська область	постачальник
2	2	ПП Сидоренко	вул. Перемоги, 64	Полтава	Полтавська область	постачальник
3	5	ВАТ "Перемога"	вул. Франко, 28	Полтава	Полтавська область	покупець
4	5	ПП Афанасьєв	вул. Різдвяна, 14	Полтава	Полтавська область	постачальник

Рисунок 2.24

4. Текст запиту можна зберегти як файл (наприклад, SupplySalesDB_SQLQuery1.sql). У разі, якщо цей запит потрібно буде виконати повторно або змінити, можна відкрити файл запиту. Для цього у головному меню потрібно вибрати пункт File, а потім у вертикальному меню вибрати пункт Open, підпункт File та вибрати відповідний файл.

Створення та виконання інших запитів виконується аналогічно. Тому далі для кожного запиту буде наведено лише його порядковий номер, умову запиту, його текст, результат та рекомендоване ім'я файлу для збереження.

Запит 2.

Мета запиту: сформувати номенклатуру товарів (тобто, список найменувань товарів), які закуповувалися у постачальників, що знаходяться у населеному пункті «Полтава» (код населеного пункту – 5), та продавалися покупцям, що знаходяться у населеному пункті «Полтава».

Текст запиту та результат виконання запиту наведені на рисунках 2.25 та 2.26 відповідно. Запит можна зберегти у файлі з ім'ям SupplySalesDB_SQLQuery2.sql.

```
USE SupplySalesDB

SELECT DISTINCT FullName AS ProductName
       FROM Purchased INNER JOIN Purchasing ON Purchasing.PurchasingID = Purchased.PurchasingID
                      INNER JOIN FPGN() f ON Purchased.ProductID = f.ProductID
                      INNER JOIN Suppliers ON Purchasing.SupplierID=Suppliers.SupplierID
            WHERE Suppliers.CityID=5
INTERSECT
SELECT DISTINCT FullName AS ProductName
       FROM  Sold INNER JOIN Sales ON  Sold.SaleID = Sales.SaleID
                  INNER JOIN FPGN() f ON  Sold.ProductID = f.ProductID
                  INNER JOIN Customers ON Sales.CustomerID=Customers.CustomerID
            WHERE Customers.CityID=5
```

Рисунок 2.25

	ProductName
1	Алкогольні напої, Вино солодке "Кагор"
2	Алкогольні напої, Коньяк "Десна"
3	Бакалія, Кава в зернах Арабіка
4	Бакалія, Кислота оцтова 9% ЩебПак
5	Бакалія, Крохмаль картопляний
6	Бакалія, Лист лавровий
7	Бакалія, Сіль Екстра
8	Бакалія, Цукор
9	Консерви, рибні "Бички у томаті"

Рисунок 2.26

Запит 3.

Мета запиту: сформувати номенклатуру товарів (тобто, список найменувань товарів), які продавалися покупцю ЗАТ "Вулкан" (код покупця – 4), і при цьому не закуповувалися у постачальника ПП Афанасьєв (код постачальника – 5).

Текст запиту та результат виконання запиту наведені на рисунках 2.27 та 2.28 відповідно. Запит можна зберегти у файлі з ім'ям SupplySalesDB_SQLQuery3.sql.

```
USE SupplySalesDB

SELECT DISTINCT FullName AS ProductName
       FROM  Sold INNER JOIN Sales ON  Sold.SaleID = Sales.SaleID
                  INNER JOIN FPGN() f ON  Sold.ProductID = f.ProductID
            WHERE Sales.CustomerID=4
EXCEPT
SELECT DISTINCT FullName AS ProductName
       FROM Purchased INNER JOIN Purchasing ON Purchasing.PurchasingID = Purchased.PurchasingID
                      INNER JOIN FPGN() f ON Purchased.ProductID = f.ProductID
            WHERE Purchasing.SupplierID=5
```

Рисунок 2.27

	ProductName
1	Алкогольні напої, Горілка "Nemiroff"
2	Бакалія, Кислота оцтова 9% ЩебПак
3	Кондитерські вироби, Цукерки льодяники "Барбарис"
4	Кондитерські вироби, Цукерки шоколадні "Білочка"
5	Консерви, рибні "Бички у томаті"

Рисунок 2.28

Запит 4.

Мета запиту: сформувати узагальнені дані про закупівлі товарів у постачальників, а саме: назва групи товарів та конкретного товару (бажано, окремо), назва постачальника, загальна кількість закуплених одиниць товару, загальна вартість закупки. Також потрібні підсумкові дані, як проміжні, так і загальні.

Текст попередньої версії запиту та фрагмент результату виконання запиту (загалом запит повертає 43 рядки) наведені на рисунках 2.29 та 2.30 відповідно. Запит можна зберегти у файлі з ім'ям SupplySalesDB_SQLQuery41.sql.

```
USE SupplySalesDB;

SELECT FullName AS ProductName,SupplierName,
       SUM(PurchasedAmount) AS SumPurchasedAmount,
       SUM(PurchasedAmount*PurchasedPrice) AS PurchasedCost
       FROM Purchased INNER JOIN Purchasing ON Purchasing.PurchasingID = Purchased.PurchasingID
                      INNER JOIN Suppliers ON Purchasing.SupplierID=Suppliers.SupplierID
                      INNER JOIN FPGN() f ON Purchased.ProductID = f.ProductID
GROUP BY FullName,SupplierName
```

Рисунок 2.29

	ProductName	SupplierName	SumPurchasedAmount	PurchasedCost
1	Бакалія, Крохмаль картопляний	ВАТ "Граніт"	200	1480.00
2	Бакалія, Лист лавровий	ВАТ "Граніт"	200	3140.00
3	Бакалія, Цукор	ВАТ "Граніт"	250	637.50
4	Алкогольні напої, Вино солодке "Кагор"	ЗАТ "Варіант"	580	9280.00
5	Алкогольні напої, Горілка "Nemiroff"	ЗАТ "Варіант"	4000	260000.00
6	Алкогольні напої, Коньяк "Десна"	ЗАТ "Варіант"	1000	15000.00
7	Бакалія, Кава в зернах Арабіка	ЗАТ "Варіант"	600	13650.00
8	Алкогольні напої, Вино солодке "Кагор"	ПП Афанасьєв	400	12656.00
9	Алкогольні напої, Коньяк "Десна"	ПП Афанасьєв	200	7912.00
10	Бакалія, Кава в зернах Арабіка	ПП Афанасьєв	150	3180.00
11	Бакалія, Крохмаль картопляний	ПП Афанасьєв	100	730.00

Рисунок 2.30

Нескладно помітити, що наведеним вище вимогам результат запиту не відповідає, а саме, немає відокремленої базової назви групи товарів та немає підсумкових даних. Отже, цей запит треба доробити.

Текст наступної версії запиту та фрагменти результату виконання запиту (загалом запит повертає 62 рядки) наведені на рисунках 2.31 та 2.32 відповідно. Запит можна зберегти у файлі з ім'ям SupplySalesDB_SQLQuery42.sql.

```
USE SupplySalesDB

SELECT SUBSTRING(FullName,1,CHARINDEX(',',FullName)-1) AS MainProductGroup,
       SUBSTRING(FullName,CHARINDEX(',',FullName)+2,LEN(FullName)-CHARINDEX(',',FullName)) AS ProductName,
       SupplierName,
       SUM(PurchasedAmount) AS SumPurchasedAmount,SUM(PurchasedAmount*PurchasedPrice) AS PurchasedCost
       FROM Purchased INNER JOIN Purchasing ON Purchasing.PurchasingID = Purchased.PurchasingID
                      INNER JOIN Suppliers ON Purchasing.SupplierID=Suppliers.SupplierID
                      INNER JOIN FPGN() f ON Purchased.ProductID = f.ProductID
GROUP BY ROLLUP(SUBSTRING(FullName,1,CHARINDEX(',',FullName)-1),
         SUBSTRING(FullName,CHARINDEX(',',FullName)+2,LEN(FullName)-CHARINDEX(',',FullName)),
         SupplierName)
```

Рисунок 2.31

	MainProductGroup	ProductName	SupplierName	SumPurchasedAmount	PurchasedCost
1	Алкогольні напої	Вино солодке "Кагор"	ЗАТ "Варіант"	580	9280.00
2	Алкогольні напої	Вино солодке "Кагор"	ПП Афанасьєв	400	12656.00
3	Алкогольні напої	Вино солодке "Кагор"	ПП Сидоренко	6000	190800.00
4	Алкогольні напої	Вино солодке "Кагор"	ТОВ "Маяк"	3000	95400.00
5	Алкогольні напої	Вино солодке "Кагор"	ТОВ "Схід"	8000	254400.00
6	Алкогольні напої	Вино солодке "Кагор"	NULL	17980	562536.00
7	Алкогольні напої	Горілка "Nemiroff"	ЗАТ "Варіант"	4000	260000.00
8	Алкогольні напої	Горілка "Nemiroff"	ТОВ "Маяк"	2300	149500.00
9	Алкогольні напої	Горілка "Nemiroff"	NULL	6300	409500.00
10	Алкогольні напої	Коньяк "Десна"	ЗАТ "Варіант"	1000	15000.00
11	Алкогольні напої	Коньяк "Десна"	ПП Афанасьєв	200	7912.00
…					
52	Кондитерські вироби	Цукерки льодяники "Барбарис"	NULL	4150	97317.50
53	Кондитерські вироби	Цукерки шоколадні "Білочка"	ТОВ "Маяк"	5500	53625.00
54	Кондитерські вироби	Цукерки шоколадні "Білочка"	NULL	5500	53625.00
55	Кондитерські вироби	NULL	NULL	9650	150942.50
56	Консерви	рибні "Бички у томаті"	ТОВ "Маяк"	3200	72000.00
57	Консерви	рибні "Бички у томаті"	ТОВ "Схід"	500	10750.00
58	Консерви	рибні "Бички у томаті"	NULL	3700	82750.00
59	Консерви	рибні "Кілька у томаті"	ТОВ "Маяк"	1250	19437.50
60	Консерви	рибні "Кілька у томаті"	NULL	1250	19437.50
61	Консерви	NULL	NULL	4950	102187.50
62	NULL	NULL	NULL	67035	1536464.00

Рисунок 2.32

Нескладно помітити, що наведеним вище вимогам результат запиту відповідає, але лише формально, тому що для підсумкових даних використовується значення NULL, яке може бути незрозумілим користувачу. Отже, цей запит треба доробити.

Текст наступної версії запиту та фрагменти результату виконання запиту (загалом запит повертає 62 рядки) наведені на рисунках 2.33 та 2.34 відповідно. Запит можна зберегти у файлі з ім'ям SupplySalesDB_SQLQuery43.sql.

```
USE SupplySalesDB

SELECT ISNULL(SUBSTRING(FullName,1,CHARINDEX(',',FullName)-1),'Усі групи товарів') AS MainProductGroup,
       ISNULL(SUBSTRING(FullName,CHARINDEX(',',FullName)+2,LEN(FullName)-CHARINDEX(',',FullName)),'Усі товари') AS ProductName,
       ISNULL(SupplierName,'Усі постачальники') AS SupplierName,
       SUM(PurchasedAmount) AS SumPurchasedAmount,SUM(PurchasedAmount*PurchasedPrice) AS PurchasedCost
       FROM Purchased INNER JOIN Purchasing ON Purchasing.PurchasingID = Purchased.PurchasingID
                      INNER JOIN Suppliers ON Purchasing.SupplierID=Suppliers.SupplierID
                      INNER JOIN FPGN() f ON Purchased.ProductID = f.ProductID
GROUP BY ROLLUP(SUBSTRING(FullName,1,CHARINDEX(',',FullName)-1),
         SUBSTRING(FullName,CHARINDEX(',',FullName)+2,LEN(FullName)-CHARINDEX(',',FullName)),
         SupplierName)
```

Рисунок 2.33

	MainProductGroup	ProductName	SupplierName	SumPurchasedAmount	PurchasedCost
1	Алкогольні напої	Вино солодке "Кагор"	ЗАТ "Варіант"	580	9280.00
2	Алкогольні напої	Вино солодке "Кагор"	ПП Афанасьєв	400	12656.00
3	Алкогольні напої	Вино солодке "Кагор"	ПП Сидоренко	6000	190800.00
4	Алкогольні напої	Вино солодке "Кагор"	ТОВ "Маяк"	3000	95400.00
5	Алкогольні напої	Вино солодке "Кагор"	ТОВ "Схід"	8000	254400.00
6	Алкогольні напої	Вино солодке "Кагор"	Усі постачальники	17980	562536.00
7	Алкогольні напої	Горілка "Nemiroff"	ЗАТ "Варіант"	4000	260000.00
8	Алкогольні напої	Горілка "Nemiroff"	ТОВ "Маяк"	2300	149500.00
9	Алкогольні напої	Горілка "Nemiroff"	Усі постачальники	6300	409500.00
10	Алкогольні напої	Коньяк "Десна"	ЗАТ "Варіант"	1000	15000.00
11	Алкогольні напої	Коньяк "Десна"	ПП Афанасьєв	200	7912.00
…					
52	Кондитерські вироби	Цукерки льодяники "Барбарис"	Усі постачальники	4150	97317.50
53	Кондитерські вироби	Цукерки шоколадні "Білочка"	ТОВ "Маяк"	5500	53625.00
54	Кондитерські вироби	Цукерки шоколадні "Білочка"	Усі постачальники	5500	53625.00
55	Кондитерські вироби	Усі товари	Усі постачальники	9650	150942.50
56	Консерви	рибні "Бички у томаті"	ТОВ "Маяк"	3200	72000.00
57	Консерви	рибні "Бички у томаті"	ТОВ "Схід"	500	10750.00
58	Консерви	рибні "Бички у томаті"	Усі постачальники	3700	82750.00
59	Консерви	рибні "Кілька у томаті"	ТОВ "Маяк"	1250	19437.50
60	Консерви	рибні "Кілька у томаті"	Усі постачальники	1250	19437.50
61	Консерви	Усі товари	Усі постачальники	4950	102187.50
62	Усі групи товарів	Усі товари	Усі постачальники	67035	1536464.00

Рисунок 2.34

Таким чином, умови запиту виконано. Це стало можливим завдяки використанню спеціального розширення для групування даних – оператора ROLLUP та відповідних змін у тексті запиту.

Але у певних ситуаціях може виникнути потреба у застосуванні іншого спеціального розширення для групування даних – оператора CUBE.

Текст наступної версії запиту та фрагменти результату виконання запиту (загалом запит повертає 141 рядок) наведені на рисунках 2.35 та 2.36 відповідно. Запит зберігати у файлі не обов’язково, бо він буде модифікований при створенні наступного запиту.

```
USE SupplySalesDB

SELECT ISNULL(SUBSTRING(FullName,1,CHARINDEX(',',FullName)-1),'Усі групи товарів') AS MainProductGroup,
       ISNULL(SUBSTRING(FullName,CHARINDEX(',',FullName)+2,LEN(FullName)-CHARINDEX(',',FullName)),'Усі товари') AS ProductName,
       ISNULL(SupplierName,'Усі постачальники') AS SupplierName,
       SUM(PurchasedAmount) AS SumPurchasedAmount,SUM(PurchasedAmount*PurchasedPrice) AS PurchasedCost
       FROM Purchased INNER JOIN Purchasing ON Purchasing.PurchasingID = Purchased.PurchasingID
                      INNER JOIN Suppliers ON Purchasing.SupplierID=Suppliers.SupplierID
                      INNER JOIN FPGN() f ON Purchased.ProductID = f.ProductID
GROUP BY CUBE(SUBSTRING(FullName,1,CHARINDEX(',',FullName)-1),
         SUBSTRING(FullName,CHARINDEX(',',FullName)+2,LEN(FullName)-CHARINDEX(',',FullName)),
         SupplierName)
```

Рисунок 2.35

	MainProductGroup	ProductName	SupplierName	SumPurchasedAmount	PurchasedCost
1	Бакалія	Крохмаль картопляний	ВАТ "Граніт"	200	1480.00
2	Усі групи товарів	Крохмаль картопляний	ВАТ "Граніт"	200	1480.00
3	Бакалія	Лист лавровий	ВАТ "Граніт"	200	3140.00
4	Усі групи товарів	Лист лавровий	ВАТ "Граніт"	200	3140.00
5	Бакалія	Цукор	ВАТ "Граніт"	250	637.50
6	Усі групи товарів	Цукор	ВАТ "Граніт"	250	637.50
7	Усі групи товарів	Усі товари	ВАТ "Граніт"	650	5257.50
8	Алкогольні напої	Вино солодке "Кагор"	ЗАТ "Варіант"	580	9280.00
9	Усі групи товарів	Вино солодке "Кагор"	ЗАТ "Варіант"	580	9280.00
10	Алкогольні напої	Горілка "Nemiroff"	ЗАТ "Варіант"	4000	260000.00
11	Усі групи товарів	Горілка "Nemiroff"	ЗАТ "Варіант"	4000	260000.00
…					
131	Усі групи товарів	рибні "Бички у томаті"	Усі постачальники	3700	82750.00
132	Консерви	рибні "Кілька у томаті"	Усі постачальники	1250	19437.50
133	Усі групи товарів	рибні "Кілька у томаті"	Усі постачальники	1250	19437.50
134	Бакалія	Сіль Екстра	Усі постачальники	2705	4861.50
135	Усі групи товарів	Сіль Екстра	Усі постачальники	2705	4861.50
136	Кондитерські вироби	Цукерки льодяники "Барбарис"	Усі постачальники	4150	97317.50
137	Усі групи товарів	Цукерки льодяники "Барбарис"	Усі постачальники	4150	97317.50
138	Кондитерські вироби	Цукерки шоколадні "Білочка"	Усі постачальники	5500	53625.00
139	Усі групи товарів	Цукерки шоколадні "Білочка"	Усі постачальники	5500	53625.00
140	Бакалія	Цукор	Усі постачальники	5150	19742.50
141	Усі групи товарів	Цукор	Усі постачальники	5150	19742.50

Рисунок 2.36

Нескладно помітити, що результат запиту є не дуже вдалим та містить надлишкові дані. Це стосується окремого групування за групою товарів. Це є недоцільним, бо кожен товар входить лише до однієї групи товарів.

Текст наступної версії запиту та фрагменти результату виконання запиту (загалом запит повертає 65 рядків) наведені на рисунках 2.37 та 2.38 відповідно. Нескладно помітити, що зміна результату була обумовлена об’єднанням умов групування – групи товарів та товару – за допомогою додаткових дужок. Запит можна зберегти у файлі з ім'ям SupplySalesDB_SQLQuery44.sql.

```
USE SupplySalesDB

SELECT ISNULL(SUBSTRING(FullName,1,CHARINDEX(',',FullName)-1),'Усі групи товарів') AS MainProductGroup,
       ISNULL(SUBSTRING(FullName,CHARINDEX(',',FullName)+2,LEN(FullName)-CHARINDEX(',',FullName)),'Усі товари') AS ProductName,
       ISNULL(SupplierName,'Усі постачальники') AS SupplierName,
       SUM(PurchasedAmount) AS SumPurchasedAmount,SUM(PurchasedAmount*PurchasedPrice) AS PurchasedCost
       FROM Purchased INNER JOIN Purchasing ON Purchasing.PurchasingID = Purchased.PurchasingID
                      INNER JOIN Suppliers ON Purchasing.SupplierID=Suppliers.SupplierID
                      INNER JOIN FPGN() f ON Purchased.ProductID = f.ProductID
GROUP BY CUBE((SUBSTRING(FullName,1,CHARINDEX(',',FullName)-1),
         SUBSTRING(FullName,CHARINDEX(',',FullName)+2,LEN(FullName)-CHARINDEX(',',FullName))),
         SupplierName)
```

Рисунок 2.37

	MainProductGroup	ProductName	SupplierName	SumPurchasedAmount	PurchasedCost
1	Бакалія	Крохмаль картопляний	ВАТ "Граніт"	200	1480.00
2	Бакалія	Лист лавровий	ВАТ "Граніт"	200	3140.00
3	Бакалія	Цукор	ВАТ "Граніт"	250	637.50
4	Усі групи товарів	Усі товари	ВАТ "Граніт"	650	5257.50
5	Алкогольні напої	Вино солодке "Кагор"	ЗАТ "Варіант"	580	9280.00
6	Алкогольні напої	Горілка "Nemiroff"	ЗАТ "Варіант"	4000	260000.00
7	Алкогольні напої	Коньяк "Десна"	ЗАТ "Варіант"	1000	15000.00
8	Бакалія	Кава в зернах Арабіка	ЗАТ "Варіант"	600	13650.00
9	Усі групи товарів	Усі товари	ЗАТ "Варіант"	6180	297930.00
10	Алкогольні напої	Вино солодке "Кагор"	ПП Афанасьєв	400	12656.00
11	Алкогольні напої	Коньяк "Десна"	ПП Афанасьєв	200	7912.00
...					
55	Бакалія	Кислота оцтова 9% ЩебПак	Усі постачальники	4760	18274.00
56	Алкогольні напої	Коньяк "Десна"	Усі постачальники	2450	72362.00
57	Бакалія	Крохмаль картопляний	Усі постачальники	3800	28397.50
58	Бакалія	Лист лавровий	Усі постачальники	2430	36994.50
59	Бакалія	Перець чорний мелений Мрія	Усі постачальники	1780	9434.00
60	Консерви	рибні "Бички у томаті"	Усі постачальники	3700	82750.00
61	Консерви	рибні "Кілька у томаті"	Усі постачальники	1250	19437.50
62	Бакалія	Сіль Екстра	Усі постачальники	2705	4861.50
63	Кондитерські вироби	Цукерки льодяники "Барбарис"	Усі постачальники	4150	97317.50
64	Кондитерські вироби	Цукерки шоколадні "Білочка"	Усі постачальники	5500	53625.00
65	Бакалія	Цукор	Усі постачальники	5150	19742.50

Рисунок 2.38

Таким чином, вихідні умови запиту також виконано. Це стало можливим завдяки використанню спеціального розширення для групування даних – оператора CUBE та відповідних змін у тексті запиту. Також рекомендується порівняти між собою результати запитів, де для групування було використано ROLLUP та CUBE.

У певних ситуаціях користувачу при аналізі результатів наведених вище запитів можуть бути потрібні лише підсумкові дані. Для цього може бути застосований інший засіб спеціального розширення для групування даних – оператор GROUPING SETS.

Текст наступної версії запиту та результат виконання запиту наведені на рисунках 2.39 та 2.40 відповідно. Запит зберігати у файлі не обов’язково, бо він буде модифікований при створенні наступного запиту.

```
USE SupplySalesDB

SELECT ISNULL(SUBSTRING(FullName,1,CHARINDEX(',',FullName)-1),'Усі групи товарів') AS MainProductGroup,
       ISNULL(SUBSTRING(FullName,CHARINDEX(',',FullName)+2,LEN(FullName)-CHARINDEX(',',FullName)),'Усі товари') AS ProductName,
       ISNULL(SupplierName,'Усі постачальники') AS SupplierName,
       SUM(PurchasedAmount) AS SumPurchasedAmount,SUM(PurchasedAmount*PurchasedPrice) AS PurchasedCost
       FROM Purchased INNER JOIN Purchasing ON Purchasing.PurchasingID = Purchased.PurchasingID
                      INNER JOIN Suppliers ON Purchasing.SupplierID=Suppliers.SupplierID
                      INNER JOIN FPGN() f ON Purchased.ProductID = f.ProductID
GROUP BY GROUPING SETS(SUBSTRING(FullName,1,CHARINDEX(',',FullName)-1),
         SUBSTRING(FullName,CHARINDEX(',',FullName)+2,LEN(FullName)-CHARINDEX(',',FullName)),
         SupplierName)
```

Рисунок 2.39

Нескладно помітити, що результат запиту є не дуже вдалим та містить надлишкові дані. Це стосується окремого групування за групою товарів. Це є недоцільним, бо кожен товар входить лише до однієї групи товарів.

	MainProductGroup	ProductName	SupplierName	SumPurchasedAmount	PurchasedCost
1	Усі групи товарів	Усі товари	ВАТ "Граніт"	650	5257.50
2	Усі групи товарів	Усі товари	ЗАТ "Варіант"	6180	297930.00
3	Усі групи товарів	Усі товари	ПП Афанасьєв	2150	28718.00
4	Усі групи товарів	Усі товари	ПП Сидоренко	8150	201202.00
5	Усі групи товарів	Усі товари	ТОВ "Горизонт"	1460	5694.00
6	Усі групи товарів	Усі товари	ТОВ "Маяк"	38030	715074.00
7	Усі групи товарів	Усі товари	ТОВ "Схід"	10415	282588.50
8	Усі групи товарів	Вино солодке "Кагор"	Усі постачальники	17980	562536.00
9	Усі групи товарів	Горілка "Nemiroff"	Усі постачальники	6300	409500.00
10	Усі групи товарів	Кава в зернах Арабіка	Усі постачальники	5080	121232.00
11	Усі групи товарів	Кислота оцтова 9% ...	Усі постачальники	4760	18274.00
12	Усі групи товарів	Коньяк "Десна"	Усі постачальники	2450	72362.00
13	Усі групи товарів	Крохмаль картоплян...	Усі постачальники	3800	28397.50
14	Усі групи товарів	Лист лавровий	Усі постачальники	2430	36994.50
15	Усі групи товарів	Перець чорний меле...	Усі постачальники	1780	9434.00
16	Усі групи товарів	рибні "Бички у томаті"	Усі постачальники	3700	82750.00
17	Усі групи товарів	рибні "Кілька у томаті"	Усі постачальники	1250	19437.50
18	Усі групи товарів	Сіль Екстра	Усі постачальники	2705	4861.50
19	Усі групи товарів	Цукерки льодяники ...	Усі постачальники	4150	97317.50
20	Усі групи товарів	Цукерки шоколадні "...	Усі постачальники	5500	53625.00
21	Усі групи товарів	Цукор	Усі постачальники	5150	19742.50
22	Алкогольні напої	Усі товари	Усі постачальники	26730	1044398.00
23	Бакалія	Усі товари	Усі постачальники	25705	238936.00
24	Кондитерські в...	Усі товари	Усі постачальники	9650	150942.50
25	Консерви	Усі товари	Усі постачальники	4950	102187.50

Рисунок 2.40

Текст наступної версії запиту та результат виконання запиту наведені на рисунках 2.41 та 2.42 відповідно. Нескладно помітити, що зміна результату була обумовлена об’єднанням умов групування – групи товарів та товару – за допомогою додаткових дужок. Запит можна зберегти у файлі з ім'ям SupplySalesDB_SQLQuery45.sql.

```
USE SupplySalesDB

SELECT ISNULL(SUBSTRING(FullName,1,CHARINDEX(',',FullName)-1),'Усі групи товарів') AS MainProductGroup,
       ISNULL(SUBSTRING(FullName,CHARINDEX(',',FullName)+2,LEN(FullName)-CHARINDEX(',',FullName)),'Усі товари') AS ProductName,
       ISNULL(SupplierName,'Усі постачальники') AS SupplierName,
       SUM(PurchasedAmount) AS SumPurchasedAmount,SUM(PurchasedAmount*PurchasedPrice) AS PurchasedCost
       FROM Purchased INNER JOIN Purchasing ON Purchasing.PurchasingID = Purchased.PurchasingID
                      INNER JOIN Suppliers ON Purchasing.SupplierID=Suppliers.SupplierID
                      INNER JOIN FPGN() f ON Purchased.ProductID = f.ProductID
GROUP BY GROUPING SETS( (SUBSTRING(FullName,1,CHARINDEX(',',FullName)-1),
         SUBSTRING(FullName,CHARINDEX(',',FullName)+2,LEN(FullName)-CHARINDEX(',',FullName)) ),
         SupplierName)
```

Рисунок 2.41

	MainProductGroup	ProductName	SupplierName	SumPurchasedAmount	PurchasedCost
1	Усі групи товарів	Усі товари	ВАТ "Граніт"	650	5257.50
2	Усі групи товарів	Усі товари	ЗАТ "Варіант"	6180	297930.00
3	Усі групи товарів	Усі товари	ПП Афанасьєв	2150	28718.00
4	Усі групи товарів	Усі товари	ПП Сидоренко	8150	201202.00
5	Усі групи товарів	Усі товари	ТОВ "Горизонт"	1460	5694.00
6	Усі групи товарів	Усі товари	ТОВ "Маяк"	38030	715074.00
7	Усі групи товарів	Усі товари	ТОВ "Схід"	10415	282588.50
8	Алкогольні напої	Вино солодке "Кагор"	Усі постачальники	17980	562536.00
9	Алкогольні напої	Горілка "Nemiroff"	Усі постачальники	6300	409500.00
10	Бакалія	Кава в зернах Арабіка	Усі постачальники	5080	121232.00
11	Бакалія	Кислота оцтова 9% ЩебПак	Усі постачальники	4760	18274.00
12	Алкогольні напої	Коньяк "Десна"	Усі постачальники	2450	72362.00
13	Бакалія	Крохмаль картопляний	Усі постачальники	3800	28397.50
14	Бакалія	Лист лавровий	Усі постачальники	2430	36994.50
15	Бакалія	Перець чорний мелений Мрія	Усі постачальники	1780	9434.00
16	Консерви	рибні "Бички у томаті"	Усі постачальники	3700	82750.00
17	Консерви	рибні "Кілька у томаті"	Усі постачальники	1250	19437.50
18	Бакалія	Сіль Екстра	Усі постачальники	2705	4861.50
19	Кондитерські вироби	Цукерки льодяники "Барбарис"	Усі постачальники	4150	97317.50
20	Кондитерські вироби	Цукерки шоколадні "Білочка"	Усі постачальники	5500	53625.00
21	Бакалія	Цукор	Усі постачальники	5150	19742.50

Рисунок 2.42

Запит 5.

Мета запиту: сформувати дані про кількісну динаміку закупок товарів протягом 2003 року. Дані мають бути агреговані по-місячно та подані у вигляді таблиці, рядками якої є назви товарів, а стовпцями –

номери місяців 2003 року. На перетині рядка та стовпця має відображатися кількість товару, закупленого в цьому місяці.

Примітка. Цей запит ілюструє особливості створення та використання перехресного запиту засобами Transact-SQL.

Текст запиту та результат виконання запиту наведені на рисунках 2.43 та 2.44 відповідно. Запит можна зберегти у файлі з ім'ям SupplySalesDB_SQLQuery51.sql.

```
USE SupplySalesDB;

SELECT ProductName, [1],[2],[3],[4],[5],[6],[7],[8],[9],[10],[11],[12]
FROM
(
SELECT FullName AS ProductName,MONTH(PurchasingDate) AS mis, PurchasedAmount
FROM Purchasing INNER JOIN Purchased ON Purchased.PurchasingID=Purchasing.PurchasingID
                INNER JOIN FPGN() f ON Purchased.ProductID=f.ProductID
WHERE YEAR(PurchasingDate)=2003
) p
PIVOT
(SUM(PurchasedAmount) FOR mis IN ([1],[2],[3],[4],[5],[6],[7],[8],[9],[10],[11],[12])) AS pvt
ORDER BY ProductName
```

Рисунок 2.43

	ProductName	1	2	3	4	5	6	7	8	9	10	11	12
1	Алкогольні напої, Вино солодке "Кагор"	3000	NULL	6000	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
2	Алкогольні напої, Горілка "Nemiroff"	2300	4000	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
3	Алкогольні напої, Коньяк "Десна"	1250	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
4	Бакалія, Кава в зернах Арабіка	4000	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
5	Бакалія, Кислота оцтова 9% ЩебПак	2700	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
6	Бакалія, Крохмаль картопляний	3000	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
7	Бакалія, Лист лавровий	1500	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
8	Бакалія, Перець чорний мелений Мрія	1780	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
9	Бакалія, Сіль Екстра	2400	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
10	Бакалія, Цукор	2000	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
11	Кондитерські вироби, Цукерки льодяники "Барбарис"	4150	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
12	Кондитерські вироби, Цукерки шоколадні "Білочка"	5500	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
13	Консерви, рибні "Бички у томаті"	3200	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
14	Консерви, рибні "Кілька у томаті"	1250	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Рисунок 2.44

Наведений на рисунку 2.44 результат запиту може бути незручним для сприйняття (наприклад, через наявність значень NULL). Цей недолік можна усунути, наприклад, шляхом заміни значень NULL на 0. Текст зміненого запиту та результат виконання запиту наведені на рисунках 2.45 та 2.46 відповідно. Запит можна зберегти у файлі з ім'ям SupplySalesDB_SQLQuery52.sql

```
USE SupplySalesDB;

SELECT ProductName, ISNULL([1],0) AS [1], ISNULL([2],0) AS [2],   ISNULL([3],0) AS [3],   ISNULL([4],0) AS [4],
                    ISNULL([5],0) AS [5], ISNULL([6],0) AS [6],   ISNULL([7],0) AS [7],   ISNULL([8],0) AS [8],
                    ISNULL([9],0) AS [9], ISNULL([10],0) AS [10], ISNULL([11],0) AS [11], ISNULL([12],0) AS [12]
FROM
(
SELECT FullName AS ProductName,MONTH(PurchasingDate) AS mis, PurchasedAmount
FROM Purchasing INNER JOIN Purchased ON Purchased.PurchasingID=Purchasing.PurchasingID
                INNER JOIN FPGN() f ON Purchased.ProductID=f.ProductID
WHERE YEAR(PurchasingDate)=2003
) p
PIVOT
(SUM(PurchasedAmount) FOR mis IN ([1],[2],[3],[4],[5],[6],[7],[8],[9],[10],[11],[12])) AS pvt
ORDER BY ProductName
```

Рисунок 2.45

	ProductName	1	2	3	4	5	6	7	8	9	10	11	12
1	Алкогольні напої, Вино солодке "Кагор"	3000	0	6000	0	0	0	0	0	0	0	0	0
2	Алкогольні напої, Горілка "Nemiroff"	2300	4000	0	0	0	0	0	0	0	0	0	0
3	Алкогольні напої, Коньяк "Десна"	1250	0	0	0	0	0	0	0	0	0	0	0
4	Бакалія, Кава в зернах Арабіка	4000	0	0	0	0	0	0	0	0	0	0	0
5	Бакалія, Кислота оцтова 9% ЩебПак	2700	0	0	0	0	0	0	0	0	0	0	0
6	Бакалія, Крохмаль картопляний	3000	0	0	0	0	0	0	0	0	0	0	0
7	Бакалія, Лист лавровий	1500	0	0	0	0	0	0	0	0	0	0	0
8	Бакалія, Перець чорний мелений Мрія	1780	0	0	0	0	0	0	0	0	0	0	0
9	Бакалія, Сіль Екстра	2400	0	0	0	0	0	0	0	0	0	0	0
10	Бакалія, Цукор	2000	0	0	0	0	0	0	0	0	0	0	0
11	Кондитерські вироби, Цукерки льодяники "Барбарис"	4150	0	0	0	0	0	0	0	0	0	0	0
12	Кондитерські вироби, Цукерки шоколадні "Білочка"	5500	0	0	0	0	0	0	0	0	0	0	0
13	Консерви, рибні "Бички у томаті"	3200	0	0	0	0	0	0	0	0	0	0	0
14	Консерви, рибні "Кілька у томаті"	1250	0	0	0	0	0	0	0	0	0	0	0

Рисунок 2.46

Запит 6.

Мета запиту: сформувати дані про загальну кількість одиниць товарів, закуплених у постачальників. Дані треба сформувати у вигляді перехресної таблиці (pivot table), тобто дані мають бути подані у вигляді таблиці, рядками якої є назви товарів, а стовпцями – назви постачальників. На перетині рядка та стовпця має відображатися кількість товару, закупленого у відповідного постачальника. Якщо дані відсутні, повинна бути вказана нульова кількість.

Особливістю цього запиту, на відміну від попереднього, є те, що заздалегідь визначити структуру результуючої таблиці неможливо, оскільки дані про закупки товарів у постачальників можуть постійно

змінюватися. Таким чином, виникає потреба у створенні динамічної перехресної таблиці (dynamic pivot table).

При створенні цього запиту були використані матеріали інтернет-ресурсу https://stackoverflow.com/questions/10404348/sql-server-dynamic-pivot-query.

Текст запиту та результат виконання запиту наведені на рисунках 2.47 та 2.48 відповідно. Запит можна зберегти у файлі з ім'ям SupplySalesDB_SQLQuery6.sql.

```
USE SupplySalesDB;

SELECT FullName AS ProductName, SupplierName, PurchasedAmount
      INTO #temp
      FROM FPGN() f INNER JOIN Purchased ON Purchased.ProductID=f.ProductID
                    INNER JOIN Purchasing ON Purchased.PurchasingID=Purchasing.PurchasingID
                    INNER JOIN Suppliers ON Suppliers.SupplierID=Purchasing.SupplierID

DECLARE @DynamicPivotQuery AS NVARCHAR(MAX), @PivotColumnNames AS NVARCHAR(MAX), @PivotSelectColumnNames AS NVARCHAR(MAX)

--Get distinct values of the PIVOT Column
SELECT @PivotColumnNames= ISNULL(@PivotColumnNames + ',','') + QUOTENAME(SupplierName)
       FROM (SELECT DISTINCT SupplierName FROM #temp) AS cat

--Get distinct values of the PIVOT Column with isnull
SELECT @PivotSelectColumnNames = ISNULL(@PivotSelectColumnNames + ',','')
                               + 'ISNULL(' + QUOTENAME(SupplierName) + ', 0) AS '
                               + QUOTENAME(SupplierName)
       FROM (SELECT DISTINCT SupplierName FROM #temp) AS cat

--Prepare the PIVOT query using the dynamic
SET @DynamicPivotQuery =
N'SELECT ProductName, ' + @PivotSelectColumnNames + '
FROM #temp
pivot(SUM(PurchasedAmount) for SupplierName in (' + @PivotColumnNames + ')) as pvt';

--Execute the Dynamic Pivot Query
EXEC sp_executesql @DynamicPivotQuery
DROP TABLE #temp
```

Рисунок 2.47

	ProductName	ВАТ "Граніт"	ЗАТ "Варіант"	ПП Афанасьєв	ПП Сидоренко	ТОВ "Горизонт"	ТОВ "Маяк"	ТОВ "Схід"
1	Алкогольні напої, Вино солодке "Кагор"	0	580	400	6000	0	3000	8000
2	Алкогольні напої, Горілка "Nemiroff"	0	4000	0	0	0	2300	0
3	Алкогольні напої, Коньяк "Десна"	0	1000	200	0	0	1250	0
4	Бакалія, Кава в зернах Арабіка	0	600	150	120	0	4000	210
5	Бакалія, Кислота оцтова 9% ШебПак	0	0	0	250	1460	2700	350
6	Бакалія, Крохмаль картопляний	200	0	100	0	0	3000	500
7	Бакалія, Лист лавровий	200	0	100	180	0	1500	450
8	Бакалія, Перець чорний мелений Мрія	0	0	0	0	0	1780	0
9	Бакалія, Сіль Екстра	0	0	100	100	0	2400	105
10	Бакалія, Цукор	250	0	1100	1500	0	2000	300
11	Кондитерські вироби, Цукерки льодяники "Барбарис"	0	0	0	0	0	4150	0
12	Кондитерські вироби, Цукерки шоколадні "Білочка"	0	0	0	0	0	5500	0
13	Консерви, рибні "Бички у томаті"	0	0	0	0	0	3200	500
14	Консерви, рибні "Кілька у томаті"	0	0	0	0	0	1250	0

Рисунок 2.48

2.3 Побудова та використання збережених процедур

Для отримання доступу до переліку збережених процедур потрібно у списку об'єктів бази даних відкрити пункт Programmability і в ньому відкрити пункт Stored Procedures. В результаті з'явиться список використовуваних процедур (якщо такі процедури були створені раніше). Пункт System Stored Procedures відкривати не потрібно.

Розглянемо послідовність дій при створенні та використанні простої збереженої процедури, яка використовує параметри.

Призначення збереженої процедури: розрахунок загальної вартості продукції, яку було продано у певні населені пункти (тобто, споживачам, які знаходяться у цих населених пунктах) за певний період часу, який визначається датою початку та датою закінчення (при цьому дати початку та закінчення періоду часу повинні передаватися до збереженої процедури як параметри)..

Для створення збереженої процедури треба.

1. Клацнути правою кнопкою миші по пункту Stored Procedures і в меню вибрати пункт New Stored Procedure…. В результаті буде створено запит, що містить «заготівку» збереженої процедури.

2. Замість «заготівки» ввести текст запиту, наведений на рисунку 2.49. При цьому зберігати коментарі, які залишилися від «заготівки» (вони позначені зеленим кольором), не обов'язково.

3. Створити збережену процедуру, натиснувши кнопку Execute. У разі, якщо запит виконано успішно, у вікні Messages з'явиться повідомлення Command(s) successfully completed. У цьому випадку вікно запиту з текстом збереженої процедури можна закрити, причому запит зберігати у вигляді файлу не потрібно. Збережена процедура повинна з'явитися в списку збережених процедур. Якщо процедура відсутня у списку, потрібно клацнути правою кнопкою миші по пункту Stored Procedures і в меню вибрати пункт Refresh.

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

CREATE PROCEDURE sp_cities_sales
                @vardate1 date, @vardate2 date
AS
BEGIN

    SET NOCOUNT ON;
    SELECT Cities.CityName,SUM(SoldAmount*SoldPrice) AS TotalSoldCost
          FROM Sold INNER JOIN Sales ON Sold.SaleID=Sales.SaleID
                    INNER JOIN Customers ON Sales.CustomerID=Customers.CustomerID
                    INNER JOIN Cities ON Cities.CityID=Customers.CityID
          WHERE Sales.SaleDate BETWEEN @vardate1 AND @vardate2
          GROUP BY Cities.CityName
          ORDER BY TotalSoldCost DESC
END
GO
```

Рисунок 2.49

4. Перевірити працездатність збереженої процедури. Для цього створити запит, який наведено на рисунку 2.50. Цей запит можна потім закрити без збереження. Іншим способом виконання збереженої процедури є її вибір правою кнопкою миші у списку збережених процедур і потім вибір в меню пункту Execute Stored Procedure….

5. Виконати запит. Результат виконання наведений на рисунку 2.51.

```
USE SupplySalesDB
exec sp_cities_sales '20030101','20031231'
```

Рисунок 2.50

	CityName	TotalSoldCost
1	Харків	719366.00
2	Полтава	485868.00
3	Чернигів	285069.50
4	Київ	259880.00

Рисунок 2.51

6. При необхідності внесення змін до тексту збереженої процедури, її можна відкрити в режимі редагування, для чого потрібно клацнути правою

кнопкою миші на ім'я збереженої процедури в списку збережених процедур і в меню вибрати пункт Modify. В результаті на екран буде виведений запит, що містить текст збереженої процедури. Якщо в текст збереженої процедури вносилися зміни і ці зміни потрібно зберегти, запит, що містить текст збереженої процедури, потрібно виконати. Після успішного виконання запит у вигляді файлу не потрібно зберігати. Для перевірки зміни змісту збереженої процедури, її потрібно знову відкрити в режимі редагування.

Розглянемо послідовність дій при створенні та використанні наступної збереженої процедури, яка також використовує параметри.

Призначення збереженої процедури: розрахунок ступеню затребуваності товарів та визначення вказаної кількості найбільш затребуваних товарів. При цьому ступень затребуваності товарів визначається кількістю разів їх придбання. Вказана кількість найбільш затребуваних товарів повинна передаватися до збереженої процедури як параметр.

Послідовність дій при створенні та використанні такої збереженої процедури аналогічна розглянутим вище діям при створенні збереженої процедури. Текст запиту для створення збереженої процедури наведений на рисунку 2.52. Текст запиту для виконання збереженої процедури та результат її виконання наведені на рисунках 2.53 та 2.54 відповідно.

Слід звернути увагу на особливості наведеного результату. Незважаючи на те, що вказане значення параметру передбачало формування списку з 10 найбільш затребуваних товарів, до результуючої таблиці включено 11 рядків. Це обумовлено тим, що десятий та одинадцятий товари у списку мають однакове значення кількості разів їх придбання. Тому обидва ці товари і потрапили до результуючого списку.

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

CREATE PROCEDURE sp_ProductSalesFrequency
                @varTOP int
AS
BEGIN

    SET NOCOUNT ON;

    SELECT FullName AS ProductName,Sold.ProductID,COUNT(Sold.ProductID) AS ProductSalesFrequency
           FROM Sold INNER JOIN FPGN() f ON Sold.ProductID = f.ProductID
           GROUP BY FullName,Sold.ProductID
           HAVING COUNT(Sold.ProductID) >=
                  ANY(SELECT TOP(@varTOP) COUNT(Sold.ProductID) FROM Sold GROUP BY Sold.ProductID ORDER BY COUNT(Sold.ProductID) DESC)
           ORDER BY COUNT(Sold.ProductID) DESC
END
GO
```

Рисунок 2.52

```
USE SupplySalesDB
exec sp_ProductSalesFrequency 10
```

Рисунок 2.53

	ProductName	ProductID	ProductSalesFrequency
1	Бакалія, Кислота оцтова 9% ЩебПак	3	19
2	Алкогольні напої, Вино солодке "Кагор"	14	19
3	Бакалія, Кава в зернах Арабіка	8	17
4	Бакалія, Сіль Екстра	2	15
5	Бакалія, Крохмаль картопляний	4	13
6	Бакалія, Лист лавровий	5	12
7	Алкогольні напої, Горілка "Nemiroff"	13	11
8	Бакалія, Цукор	1	10
9	Кондитерські вироби, Цукерки шоколадні "Білочка"	9	10
10	Консерви, рибні "Бички у томаті"	10	8
11	Алкогольні напої, Коньяк "Десна"	12	8

Рисунок 2.54

2.4 Збереження результатів роботи

Рекомендується зберегти тексти запитів для створення створюваних користувачем функцій та збережених процедур, а також файли, у яких були збережені створені запити.

3 ВИМОГИ ДО ЗВІТУ

Відповіднім чином оформлений та роздрукований звіт з лабораторної роботи є документом, що підтверджує виконання студентом лабораторної роботи.

У звіті з лабораторної роботи:

1) стисло описати основні етапи виконання завдання;

2) для кожного з реалізованих запитів навести умову запиту, текст запиту та результат виконання запиту;

3) зробити висновки за результатами виконання лабораторної роботи.

Звіт з лабораторної роботи роздруковується на аркушах формату А4, він повинен мати відповідній титульний аркуш. Роздрукованій звіт здається студентом викладачу у файлі.

Звіт має буті оформлень за такими вимогами:

– параметри сторінки: лівий відступ – 3 см; правий – 1,5 см; верхній та нижній відступи по 2 см;

– шрифт Times New Roman, 14;

– налаштування абзацу: вирівнювання – за шириною, відступи зліва та справа – 0 см, відступ першого рядка - 1,25 см, інтервал перед та після абзацу – 0 пт, міжрядковій інтервал – одинарний; на вкладці «Положення на сторінці» відключити функцію «Заборона висячих рядків».

Усі скріншоти розміщені у звіті, є рисунками, отже повинні мати підписи та відповідну нумерацію.

Ознакою того, що студент не тільки виконав лабораторну роботу, але й здав її (тобто підтвердив наявність відповідних знань та навичок), є наявність на титульному аркуші підпису викладача та дати здачі.

4 ПИТАННЯ ДЛЯ САМОПЕРЕВІРКИ

1. Команда SELECT-SQL. Загальна характеристика, призначення та використання.

2. Команда SELECT-SQL. Реляційні операції, які реалізуються за допомогою команди SELECT-SQL.

3. Команда SELECT-SQL. Стовпці виведення результату запиту. Використання виразів в шпальтах виведення.

4. Команда SELECT-SQL. Пропозиція FROM. Призначення і використання.

5. Команда SELECT-SQL. Пропозиція WHERE. Призначення і використання.

6. Команда SELECT-SQL. Пропозиція WHERE. Умови фільтрації і їх використання.

7. Команда SELECT-SQL. Пропозиція WHERE. Формування запиту на базі декількох таблиць. JOIN-умови і їх використання.

8. Команда SELECT-SQL. З'єднання таблиць в багатотабличних запитах. Види з'єднань – INNER JOIN, LEFT JOIN, RIGHT JOIN і їх особливості.

9. Команда SELECT-SQL. Перехресні запити і їх особливості.

10. Команда SELECT-SQL. Аргумент DISTINCT. Призначення і використання.

11. Команда SELECT-SQL. Аргумент TOP. Призначення і використання.

12. Команда SELECT-SQL. Булеві оператори AND, OR, NOT та їх використання.

13. Команда SELECT-SQL. Спеціальні оператори IN, BETWEEN. Призначення і використання.

14. Команда SELECT-SQL. Агрегуючі функції. Призначення і використання.

15. Команда SELECT-SQL. Пропозиція GROUP BY. Призначення і використання.

16. Команда SELECT-SQL. Визначить особливості спеціальних розширень для групування даних: ROLLUP, CUBE, GROUPING SETS та OVER

17. Команда SELECT-SQL. Пропозиція ORDER BY. Призначення і використання.

18. Команда SELECT-SQL. Пропозиція HAVING. Призначення і використання.

19. Команда SELECT-SQL. Підзапити. Призначення і використання.

20. Команда SELECT-SQL. Підзапити. Види підзапитів і їх особливості. Якими засобами можна працювати із результатами виконання підзапитів?

21. Команда SELECT-SQL. Операції об'єднання, перетину, різниці. Призначення і використання.

22. Команда SELECT-SQL. Функція CAST() та її призначення. Чим обумовлено застосування цієї функції у тих прикладах, що розглядалися?

23. Команда SELECT-SQL. Функція ISNULL() та її призначення.

24. Створювані користувачем функції. Загальна характеристика, призначення та використання.

25. Вбудовані функції. Загальна характеристика, призначення та використання.

26. Класифікація функцій користувача.

27. Скалярні функції. Загальна характеристика, призначення та використання.

28. Табличні функції. Загальна характеристика, призначення та використання.

29. Багатооператорні табличні функції. Загальна характеристика, призначення та використання.

30. Команди SQL для роботи з функціями користувача.

31. Вбудовані математичні функції. Загальна характеристика, призначення та використання.

32. Вбудовані строкові функції. Загальна характеристика, призначення та використання.

33. Вбудовані функції для роботи з датою і часом. Загальна характеристика, призначення та використання.

34. Збережені процедури. Загальна характеристика, призначення та використання.

35. Класифікація збережених процедур.

36. Основні завдання при створенні збереженої процедури.

37. Команди SQL для роботи зі збереженими процедурами.

38. Параметри збережених процедур.

СПИСОК ЛІТЕРАТУРИ

1. Програмування баз даних. Практикум [Електронний ресурс] : навч. посіб. для студ. спеціальності 152 «Метрологія та інформаційно-вимірювальна техніка» / М. В. Добролюбова, М. В. Філіппова, О. М. Маркіна. – Київ : КПІ ім. Ігоря Сікорського, 2021. – 164 с.

2. Anthony DeBarros. Practical SQL. 2nd Edition. – No Starch Press, Inc., 2022. – 440 c.

3. Jeffrey A. Hoffer, V. Ramesh, Heikki Topi. Modern Database Management. Thirteenth Edition. – Pearson Education Limited, 2020. – 591 p.

4. Dušan Petković. Microsoft SQL Server 2019. A Beginner’s Guide. Seventh Edition. – McGraw-Hill Education, 2020. – 865 p.

5. Mukesh Negi. Fundamentals of Database Management System: Learn essential concepts of database systems. – BPB Publications, 2019. – 175 p.

6. Edward Sciore. Database Design and Implementation: Second Edition. – Springer Nature, 2020. – 468 p.

7. Gavin Powell. Database Modeling Step by Step. – CRC Press, 2020. – 268 p.

8. Sanjiv Purba. Handbook of Data Management: 1999 Edition. – CRC Press, 2019. – 1101 p.

9. C. J. Date. Database Design and Relational Theory: Normal Forms and All That Jazz. – Apress, 2019. – 451 p.

10. Jonathan Eckstein, Bonnie R. Schultz. Introductory Relational Database Design for Business, with Microsoft Access. – John Wiley & Sons, 2018. – 328 p.

11. Alan Beaulieu. Learning SQL: Generate, Manipulate, and Retrieve Data. – O'Reilly Media, Inc., 2020. – 384 p.

12. M. Tamer Özsu, Patrick Valduriez. Principles of Distributed Database Systems. – Springer Nature, 2019. – 674 p.

Навчальне видання

МЕТОДИЧНІ ВКАЗІВКИ
до виконання лабораторної роботи за темою «Аналітична обробка оперативних даних засобами мови SQL у середовищі СУБД Microsoft SQL Server»
для студентів спеціальності
122 «Комп’ютерні науки»

Укладачі:
ОРЛОВСЬКИЙ Дмитро Леонідович
КОПП Андрій Михайлович

Відповідальний за випуск Годлевський М.Д.
Роботу до видання рекомендував Гамаюн І.П.

План 2024 р., поз. 162

Підп. до друку 15.02.2024. Гарнітура Times New Roman.
Ум. друк. арк. 3,4.

Видавничий центр НТУ «ХПІ»,
вул. Кирпичова, 2, м. Харків, 61002
Свідоцтво про державну реєстрацію ДК № 3478 від 21.08.2017 р.

Самостійне електронне видання