

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

**Український державний університет науки і технологій**

Кафедра «Комп’ютерні інформаційні технології»

**Лабораторна робота №5**

**з дисципліни «Бази даних»**

**на тему: «**Підзапити. Табличні вирази.**»**

Виконав:

Студент гр. ПЗ2011

Проценко Роман

Прийняв:

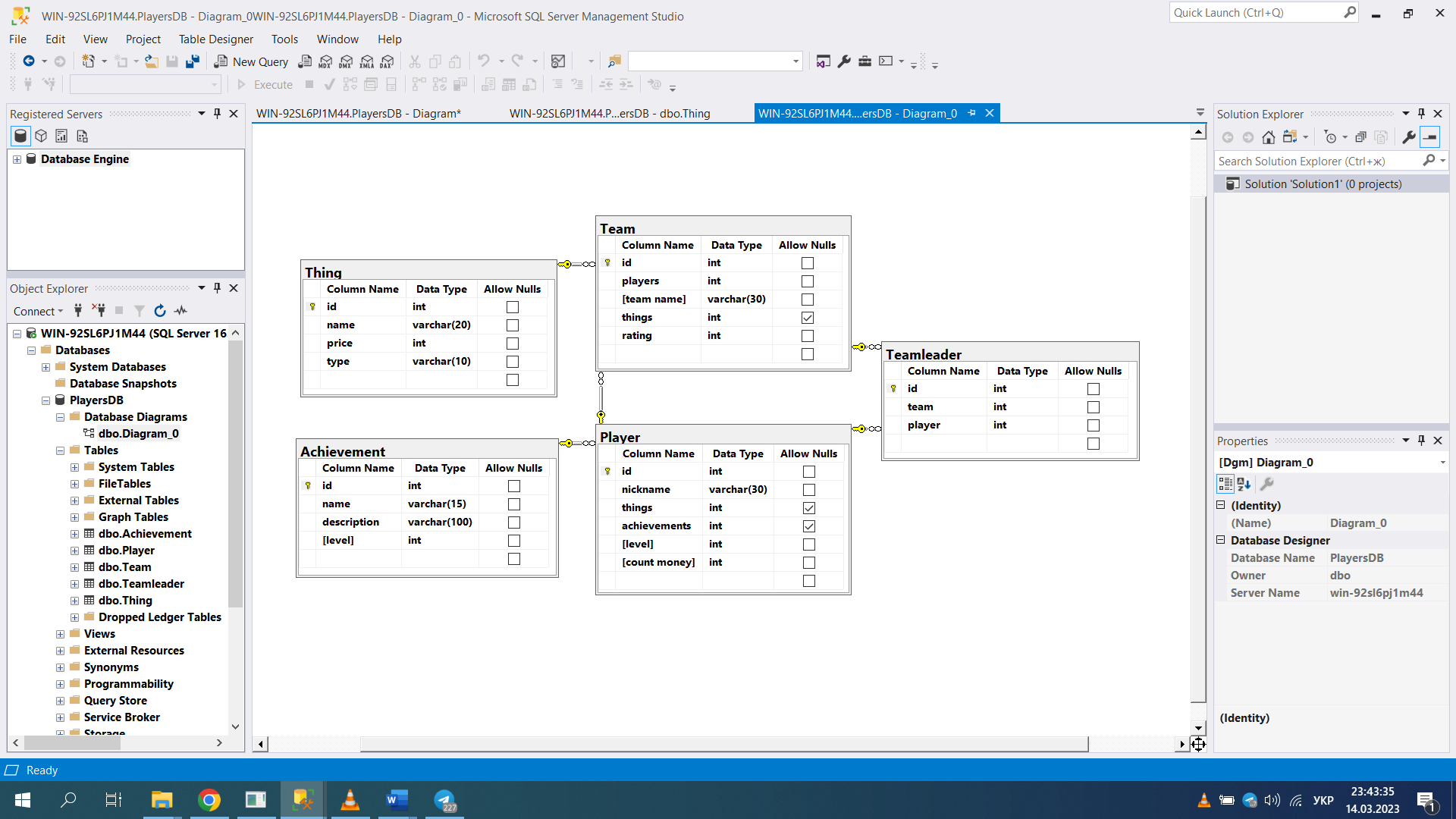
Самойлов С. П.

Дніпро, 2023

# Тема. Підзапити. Табличні вирази

# Мета. Отримати навички написання запитів із використанням підзапитів мови SQL. Познайомитися з табличними виразами, їх різновидами, вивчити сферу застосування табличних виразів.

# Скріншот діаграми бази даних із середовища SQL Server Management Studio



**Файл “Subqueries.sql”**

use Players

go

-- Приклад 1.1.1: Self-contained subquery (single-valued)

SELECT \*

FROM lr2.Player

WHERE [level] = (SELECT MAX([level]) FROM lr2.Player);

-- Приклад 1.1.2: Self-contained subquery (multivalued) з оператором IN

SELECT \*

FROM lr2.Player

WHERE [level] IN (10,11,12,13,14,15);

-- Приклад 1.1.2: Self-contained subquery (multivalued) з оператором SOME

SELECT \*

FROM lr2.Player

WHERE count\_money > SOME (SELECT AVG(count\_money) FROM lr2.Player);

SELECT \*

FROM lr2.Player p

WHERE count\_money > SOME (SELECT AVG(count\_money) FROM lr2.Player WHERE team\_id = p.team\_id);

-- Приклад 1.2.2: Корельований підзапит (multivalued) з оператором IN

SELECT \*

FROM lr2.Team t

WHERE t.id IN (SELECT id FROM lr2.Player p WHERE p.count\_money > 1000 AND p.team\_id = t.id);

-- Приклад 1.2.2: Корельований підзапит (multivalued) з оператором SOME

SELECT \*

FROM lr2.Player p

WHERE [level] > SOME (SELECT [level] FROM lr2.Player WHERE team\_id = p.team\_id);

-- Приклад 1.2.2: Корельований підзапит (multivalued) з оператором ALL

SELECT \*

FROM lr2.Team t

WHERE t.id = ALL (SELECT team\_id FROM lr2.Player p WHERE p.[level] > 5 AND p.team\_id = t.id);

-- Приклад 1.3.1: Використання EXISTS для знаходження записів, для яких існують підзапити

SELECT \*

FROM lr2.Player p

WHERE EXISTS (

SELECT 1

FROM lr2.Team t

WHERE t.id = p.team\_id AND t.team\_name = 'Fire Dragons'

);

-- Приклад 1.3.2: Використання EXISTS для знаходження записів, які не мають підзапитів

SELECT \*

FROM lr2.Team t

WHERE NOT EXISTS (

SELECT 1

FROM lr2.Player p

WHERE p.team\_id = t.id

);

-- Приклад 1.3.3: Використання EXISTS для перевірки наявності певного запису в підзапиті

SELECT \*

FROM lr2.Team t

WHERE EXISTS (

SELECT 1

FROM lr2.Player p

WHERE p.team\_id = t.id AND p.count\_money > 10000

);

-- Приклад 1.4.1: Nested Derived Tables: Вкладена похідна таблиця для обчислення середнього рівня гравців команд

SELECT \*, avg\_level

FROM (

SELECT t.team\_name, AVG(p.[level]) AS avg\_level

FROM lr2.Team t

INNER JOIN lr2.Player p ON t.id = p.team\_id

GROUP BY t.team\_name

) AS TeamLevel

WHERE avg\_level > 5;

-- Приклад 1.4.2: Використання похідної таблиці для обчислення рейтингу та кількості гравців у команді

SELECT TeamRating.team\_name, TeamRating.avg\_level, TeamPlayerCount.player\_count

FROM (

SELECT t.team\_name, AVG(p.[level]) AS avg\_level

FROM lr2.Team t

INNER JOIN lr2.Player p ON t.id = p.team\_id

GROUP BY t.team\_name

) AS TeamRating

INNER JOIN (

SELECT t.team\_name, COUNT(p.id) AS player\_count

FROM lr2.Team t

INNER JOIN lr2.Player p ON t.id = p.team\_id

GROUP BY t.team\_name

) AS TeamPlayerCount ON TeamRating.team\_name = TeamPlayerCount.team\_name;

-- Приклад 1.5.1: Використання двох CTE для обчислення середнього рейтингу та кількості гравців в командах

WITH TeamRating AS (

SELECT t.team\_name, AVG(p.[level]) AS avg\_level

FROM lr2.Team t

INNER JOIN lr2.Player p ON t.id = p.team\_id

GROUP BY t.team\_name

),

TeamPlayerCount AS (

SELECT t.team\_name, COUNT(p.id) AS player\_count

FROM lr2.Team t

INNER JOIN lr2.Player p ON t.id = p.team\_id

GROUP BY t.team\_name

)

SELECT TeamRating.team\_name, TeamRating.avg\_level, TeamPlayerCount.player\_count

FROM TeamRating

INNER JOIN TeamPlayerCount ON TeamRating.team\_name = TeamPlayerCount.team\_name;

-- Приклад 1.5.2: Використання того самого CTE двічі для отримання даних про гравців та команди

WITH TeamPlayerInfo AS (

SELECT p.nickname, p.[level], t.team\_name

FROM lr2.Player p

INNER JOIN lr2.Team t ON p.team\_id = t.id

)

SELECT tpi1.nickname AS player1, tpi2.nickname AS player2

FROM TeamPlayerInfo tpi1

JOIN TeamPlayerInfo tpi2 ON tpi1.team\_name = tpi2.team\_name

WHERE tpi1.nickname < tpi2.nickname;

-- Приклад 1.5.3: Рекурсивний CTE для побудови ієрархії команд та їх керівників

WITH RecursiveTeamHierarchy AS (

SELECT id, team\_name, NULL AS leader\_id, 0 AS [level]

FROM lr2.Team

WHERE id NOT IN (SELECT DISTINCT team FROM lr2.Teamleader)

UNION ALL

SELECT t.id, t.team\_name, tl.player AS leader\_id, rth.[level] + 1

FROM lr2.Team t

JOIN lr2.Teamleader tl ON t.id = tl.team

JOIN RecursiveTeamHierarchy rth ON tl.player = rth.id

)

SELECT id, team\_name, leader\_id, [level]

FROM RecursiveTeamHierarchy

ORDER BY [level], id;

-- Приклад 1.6: Створення перегляду з опціями ENCRYPTION, SCHEMABINDING та CHECK OPTION для гравців у команді

CREATE VIEW lr2.TeamPlayersView

WITH SCHEMABINDING

AS

SELECT

p.nickname AS PlayerName,

t.team\_name AS TeamName

FROM lr2.Player p

INNER JOIN lr2.Team t ON p.team\_id = t.id

WHERE p.level > 5

-- Приклад 1.7: Створення inline функції звітності, яка повертає таблицю зі списком гравців у команді

CREATE FUNCTION dbo.GetPlayersInTeam (@teamId INT)

RETURNS TABLE

AS

RETURN (

SELECT p.nickname, p.[level]

FROM lr2.Player p

WHERE p.team\_id = @teamId

);

-- Використання функції

SELECT \* FROM dbo.GetPlayersInTeam(1);

-- Приклад 1.8.1: Використання CROSS APPLY для отримання даних гравців та їх команд

SELECT p.nickname, t.team\_name

FROM lr2.Player p

CROSS APPLY (

SELECT team\_name

FROM lr2.Team t

WHERE t.id = p.team\_id

) AS t;

-- Приклад 1.8.2: Використання OUTER APPLY для отримання даних гравців та їх команд, включаючи гравців без команд

SELECT p.nickname, t.team\_name

FROM lr2.Player p

OUTER APPLY (

SELECT team\_name

FROM lr2.Team t

WHERE t.id = p.team\_id

) AS t;

**Висновок**

Підзапити та табличні вирази є потужними інструментами у мові SQL, які дозволяють виконувати більш складні та контекстні запити до бази даних.

Підзапити (Subqueries) використовуються для отримання даних із інших запитів або підтаблиць, що використовуються в головному запиті. Зазвичай вони використовуються у виразах WHERE, HAVING або SELECT, і можуть бути корельованими (залежать від даних головного запиту) та самостійними (не залежать від головного запиту).

Табличні вирази (Table Expressions) включають такі концепції, як Common Table Expressions (CTEs), похідні таблиці, тимчасові таблиці та вирази VALUES. Вони дозволяють створювати тимчасові таблиці або підзапити для подальшого використання в запитах. CTEs надають зручну можливість іменування та організації підзапитів для полегшення розуміння запитів.

Ці інструменти застосовуються для виконання різних завдань, включаючи фільтрацію даних, об'єднання таблиць, підрахунки, сортування та інші операції. Вони корисні при створенні звітів, вибірці даних, побудові ієрархій, агрегації та інших аналітичних завдань.

Знання та розуміння підзапитів та табличних виразів дозволяє розробникам більш ефективно використовувати SQL для вирішення різних завдань у роботі з базами даних. Ці інструменти допомагають покращити читабельність запитів та зменшити повторюваний код, що робить роботу з даними більш продуктивною.