

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

**Український державний університет науки і технологій**

Кафедра «Комп’ютерні інформаційні технології»

**Лабораторна робота №6**

**з дисципліни «Бази даних»**

**на тему: «**Обертання даних. Модифікація даних.**»**

Виконав:

Студент гр. ПЗ2011

Проценко Роман

Прийняв:

Самойлов С. П.

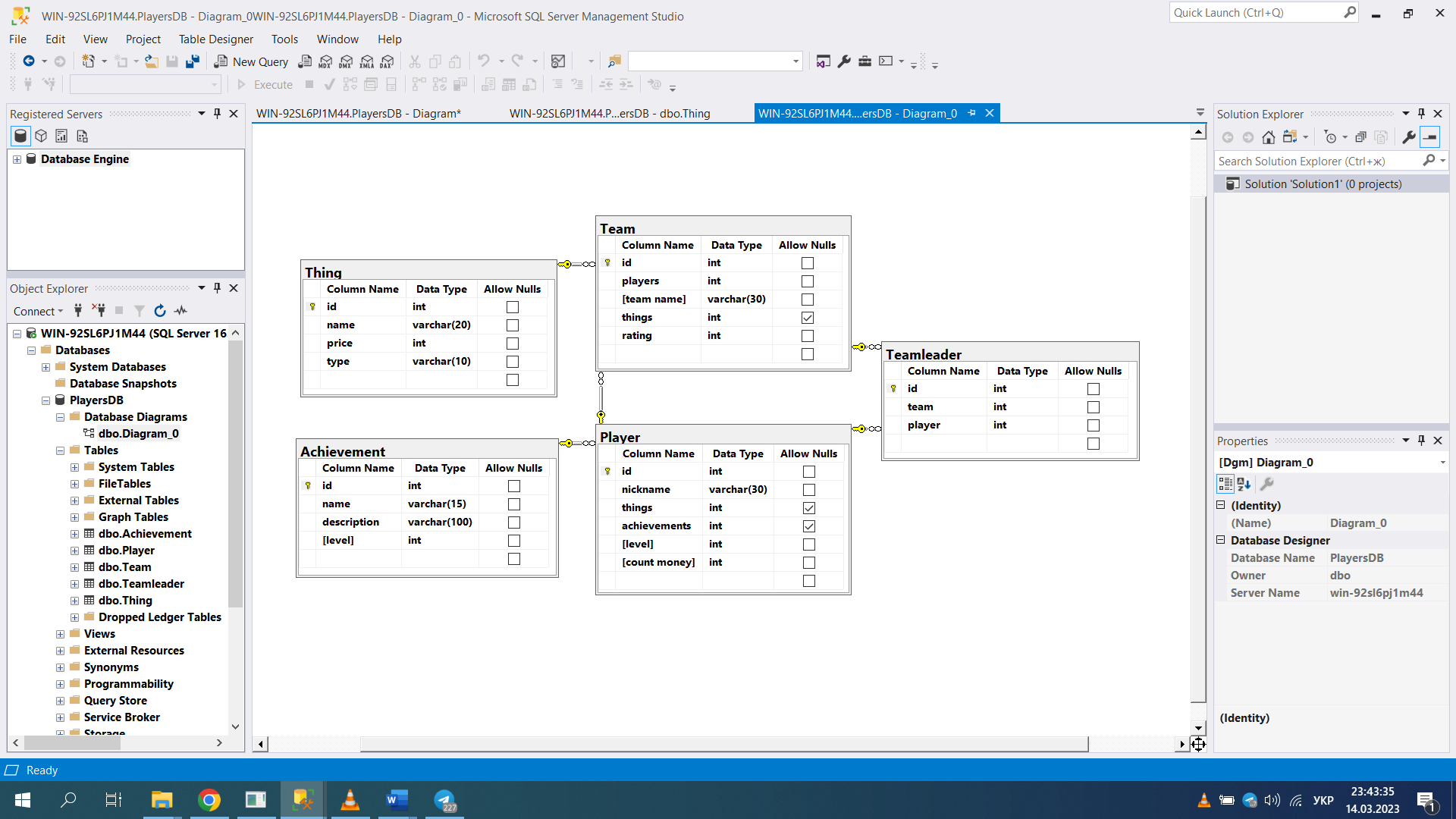
Дніпро, 2023

# Тема. Обертання даних. Модифікація даних.

# Мета. Отримати навички написання запитів SQL, що реалізують обертання даних.

# Ознайомитись з операціями модифікації даних.

# Скріншот діаграми бази даних із середовища SQL Server Management Studio



**Файл “** **AdvancedQuerying.sql”**

use Players

--1.1.1 Ranking Window Functions:

--ROW\_NUMBER:

SELECT

nickname,

COUNT(\*) AS achievements\_count,

ROW\_NUMBER() OVER (ORDER BY COUNT(\*) DESC) AS row\_num

FROM lr2.Player

JOIN dbo.Achievement ON lr2.Player.achievements = dbo.Achievement.id

GROUP BY nickname

ORDER BY achievements\_count DESC;

--1.1.2 Offset Window Functions:

SELECT

nickname,

count\_money,

LAG(count\_money) OVER (ORDER BY count\_money) AS prev\_money

FROM lr2.Player

ORDER BY count\_money;

--1.1.3 Aggregate Window Functions:

SELECT

team\_name,

AVG(rating) OVER (PARTITION BY team\_name) AS avg\_rating,

MAX(rating) OVER (PARTITION BY team\_name) AS max\_rating,

MIN(rating) OVER (PARTITION BY team\_name) AS min\_rating

FROM lr2.Team

ORDER BY team\_name;

--1.2 PIVOT:

SELECT \*

FROM (

SELECT team\_name, rating

FROM lr2.Team

) AS SourceTable

PIVOT (

AVG(rating) FOR team\_name IN ([Fire Dragons], [Thunder Wolves], [Ice Phoenixes], [Rock Avengers])

) AS PivotTable;

--1.3 UNPIVOT:

SELECT team\_name, team\_rating

FROM (

SELECT [Fire Dragons], [Thunder Wolves], [Ice Phoenixes], [Rock Avengers]

FROM (

SELECT team\_name, rating

FROM lr2.Team

) AS SourceTable

PIVOT (

AVG(rating) FOR team\_name IN ([Fire Dragons], [Thunder Wolves], [Ice Phoenixes], [Rock Avengers])

) AS PivotTable

) AS UnpivotSource

UNPIVOT (

team\_rating FOR team\_name IN ([Fire Dragons], [Thunder Wolves], [Ice Phoenixes], [Rock Avengers])

) AS UnpivotTable;

--1.4. GROUPING SETS:

SELECT

team\_name,

achievements,

COUNT(\*) AS achievement\_count

FROM lr2.Team

JOIN lr2.Player ON lr2.Team.id = lr2.Player.team\_id

JOIN dbo.Achievement ON lr2.Player.achievements = dbo.Achievement.id

GROUP BY GROUPING SETS ((team\_name, achievements), ());

--1.5. CUBE:

SELECT

team\_name,

achievements,

COUNT(\*) AS achievement\_count

FROM lr2.Team

JOIN lr2.Player ON lr2.Team.id = lr2.Player.team\_id

JOIN dbo.Achievement ON lr2.Player.achievements = dbo.Achievement.id

GROUP BY CUBE (team\_name, achievements);

--1.6. ROLLUP:

SELECT

team\_name,

achievements,

COUNT(\*) AS achievement\_count

FROM lr2.Team

JOIN lr2.Player ON lr2.Team.id = lr2.Player.team\_id

JOIN dbo.Achievement ON lr2.Player.achievements = dbo.Achievement.id

GROUP BY ROLLUP (team\_name, achievements);

--1.7. GROUPING():

SELECT

team\_name,

achievements,

GROUPING(team\_name) AS team\_name\_grouping,

GROUPING(achievements) AS achievement\_grouping,

COUNT(\*) AS achievement\_count

FROM lr2.Team

JOIN lr2.Player ON lr2.Team.id = lr2.Player.team\_id

JOIN dbo.Achievement ON lr2.Player.achievements = dbo.Achievement.id

GROUP BY team\_name, achievements WITH CUBE;

--1.8. GROUPING\_ID():

SELECT

team\_name,

achievements,

GROUPING\_ID(team\_name, achievements) AS grouping\_id,

COUNT(\*) AS achievement\_count

FROM lr2.Team

JOIN lr2.Player ON lr2.Team.id = lr2.Player.team\_id

JOIN dbo.Achievement ON lr2.Player.achievements = dbo.Achievement.id

GROUP BY team\_name, achievements WITH CUBE;

--1.9.1. INSERT VALUES:

INSERT INTO dbo.Thing (name, price, [type])

VALUES ('New Item', 200, 'Accessory');

--1.9.4. SELECT INTO:

SELECT

nickname,

things,

achievements,

[level],

count\_money,

team\_id

INTO lr2.NewPlayerTable

FROM lr2.Player

WHERE team\_id = 1;

--1.10.1. $identity:

SET IDENTITY\_INSERT dbo.Thing ON;

INSERT INTO dbo.Thing (id, name, price, [type])

VALUES (100, 'Special Item', 300, 'Weapon');

SET IDENTITY\_INSERT dbo.Thing OFF;

--1.10.2. @@IDENTITY:

INSERT INTO lr2.Player (nickname, things, achievements, [level], count\_money, team\_id)

VALUES ('NewPlayer', 1, 2, 4, 1000, 1);

SELECT @@IDENTITY AS 'LastIdentity';

--1.10.3. SCOPE\_IDENTITY():

INSERT INTO lr2.Player (nickname, things, achievements, [level], count\_money, team\_id)

VALUES ('NewPlayer1234', 1, 2, 4, 1000, 1);

SELECT SCOPE\_IDENTITY() AS 'LastIdentity';

--1.10.4. IDENT\_CURRENT():

SELECT IDENT\_CURRENT('dbo.Thing') AS 'CurrentIdentity';

--1.11. CREATE SEQUENCE:

CREATE SEQUENCE lr2.PlayerSequence

START WITH 1

INCREMENT BY 1

MINVALUE 1

MAXVALUE 100

CACHE 10;

--1.12.1. sys.sequences view:

SELECT

name,

object\_id,

schema\_id,

start\_value,

increment,

minimum\_value,

maximum\_value,

current\_value

FROM sys.sequences;

--1.13. DELETE:

DELETE FROM lr2.Player

WHERE [level] < 3;

--1.14. TRUNCATE:

TRUNCATE TABLE dbo.Thing;

--1.15. UPDATE:

UPDATE lr2.Player

SET count\_money = count\_money + 1000

WHERE team\_id = 1;

--1.16. MERGE:

MERGE INTO lr2.Player AS Target

USING dbo.Thing AS Source

ON Target.things = Source.id

WHEN MATCHED THEN

UPDATE SET

Target.achievements = 1,

Target.[level] = 3,

Target.count\_money = 500,

Target.team\_id = 1;

MERGE INTO lr2.Player AS Target

USING dbo.Thing AS Source

ON Target.things = Source.id

WHEN MATCHED AND Target.[level] < 5 THEN

UPDATE SET Target.[level] = Target.[level] + 1;

MERGE INTO lr2.Player AS Target

USING dbo.Thing AS Source

ON Target.things = Source.id

WHEN NOT MATCHED BY SOURCE AND Target.nickname = 'JohnDoe' THEN

DELETE

OUTPUT $action, Inserted.\*, Deleted.\*;

--1.17. INSERT ... OUTPUT:

DECLARE @InsertedRecords TABLE (

id INT,

name VARCHAR(30),

price INT,

[type] VARCHAR(20)

);

INSERT INTO dbo.Thing (name, price, [type])

OUTPUT INSERTED.id, INSERTED.name, INSERTED.price, INSERTED.[type] INTO @InsertedRecords

VALUES ('NewThing2', 200, 'NewType');

-- 1.18. DELETE ... OUTPUT:

SELECT \* FROM @InsertedRecords;

DECLARE @DeletedRecords TABLE (

id INT,

nickname VARCHAR(30),

things INT,

achievements INT,

[level] INT,

count\_money INT,

team\_id INT

);

DELETE FROM lr2.Player

OUTPUT DELETED.id, DELETED.nickname, DELETED.things, DELETED.achievements, DELETED.[level], DELETED.count\_money, DELETED.team\_id INTO @DeletedRecords

WHERE nickname = 'JohnDoe';

-- 1.19. UPDATE ... OUTPUT:

SELECT \* FROM @DeletedRecords;

DECLARE @UpdatedRecords TABLE (

id INT,

nickname VARCHAR(30),

things INT,

achievements INT,

[level] INT,

count\_money INT,

team\_id INT

);

UPDATE lr2.Player

SET [level] = [level] + 1

OUTPUT INSERTED.id, INSERTED.nickname, INSERTED.things, INSERTED.achievements, INSERTED.[level], INSERTED.count\_money, INSERTED.team\_id INTO @UpdatedRecords

WHERE nickname = 'JaneDoe';

--1.20. MERGE ... OUTPUT:

SELECT \* FROM @UpdatedRecords;

DECLARE @MergedRecords TABLE (

ActionTaken NVARCHAR(10),

id INT,

nickname VARCHAR(30),

things INT,

achievements INT,

[level] INT,

count\_money INT,

team\_id INT

);

MERGE INTO lr2.Player AS Target

USING dbo.Thing AS Source

ON Target.things = Source.id

WHEN MATCHED THEN

UPDATE SET Target.achievements = 1

WHEN NOT MATCHED THEN

INSERT (nickname, things, achievements, [level], count\_money, team\_id)

VALUES ('NewPlayer777', Source.id, 1, 3, 500, 1)

OUTPUT $action, INSERTED.\* INTO @MergedRecords;

-- Retrieve the merged records

SELECT \* FROM @MergedRecords;

**Висновок**

У ході лабораторної роботи з обертання та модифікації даних в базі "Players" було успішно використано віконні функції, групування, та різноманітні операції вставки, оновлення, видалення та синхронізації. Оператори OUTPUT дозволили отримати та вивести змінені дані. Отримано практичні навички у написанні ефективних SQL-запитів, що є важливим у розробці оптимізованих операцій з базами даних.