# Лабораторна робота №6

# З'єднання таблиць

# Неявне з'єднання таблиць

Для зведення даних з різних таблиць ми можемо використовувати стандартну команду SELECT. У нас є таблиці, які пов'язані між собою зв'язками (УВАГА! Якщо ви виконували всі роботи підряд, то у вас ці таблиці вже є):

|  |
| --- |
| USE productsdb;  CREATE TABLE Products  (  Id INT IDENTITY PRIMARY KEY,  ProductName NVARCHAR (30) NOT NULL,  Manufacturer NVARCHAR (20) NOT NULL,  ProductCount INT DEFAULT 0,  Price MONEY NOT NULL  );  CREATE TABLE Customers  (  Id INT IDENTITY PRIMARY KEY,  FirstName NVARCHAR (30) NOT NULL  );  CREATE TABLE Orders  (  Id INT IDENTITY PRIMARY KEY,  ProductId INT NOT NULL REFERENCES Products (Id),  CustomerId INT NOT NULL REFERENCES Customers (Id),  CreatedAt DATE NOT NULL,  ProductCount INT DEFAULT 1,  Price MONEY NOT NULL  ); |

Тут таблиці Products і Customers пов'язані з таблицею Orders зв'язком один до багатьох. Таблиця Orders у вигляді зовнішніх ключів ProductId і CustomerId містить посилання на стовпці Id з відповідно таблиць Products і Customers. Також вона зберігає кількість купленого товару (ProductCount) і і за якою ціною він був куплений (Price). І крім того, таблиці також зберігає у вигляді стовпчика CreatedAt дату покупки.

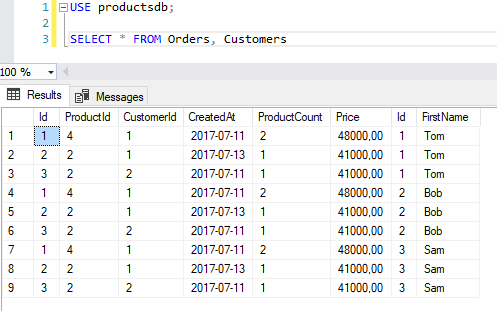
Ці таблиці містять такі дані (**УВАГА! Якщо ви виконували всі роботи підряд, то у вас ці таблиці вже є**):

|  |
| --- |
| INSERT INTO Products  VALUES ( 'IPhone 11', 'Apple', 2, 21999),  ( 'IPhone 12 Pro', 'Apple', 2, 56000),  ( 'IPhone 12', 'Apple', 5, 28999),  ( 'Galaxy S21 Ultra', 'Samsung', 2, 39999),  ( 'Galaxy Z Fold2', 'Samsung', 1, 60000),  ( 'Xiaomi Mi 11', 'Xiaomi', 2, 26999),  ( 'OnePlus 8', 'OnePlus', 6, 17775)  INSERT INTO Customers VALUES ('Tom'), ('Bob'), ('Sam')  INSERT INTO Orders  VALUES  (4,2,'2021-02-11',2,39999),  (2,2,'2021-02-11',2,56000),  (4,1,'2021-02-13',1,39999),  (7,1,'2021-02-14',5,17775) |

Тепер з'єднаємо дві таблиці Orders і Customers:

|  |  |
| --- | --- |
| 1 | SELECT \* FROM Orders, Customers |

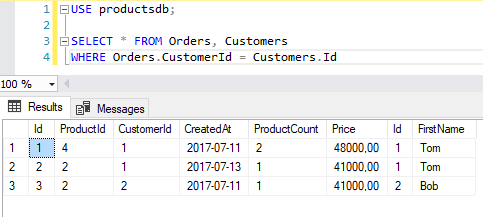
При такій вибірці для кожен рядок з таблиці Orders буде поєднуватися з кожним рядком з таблиці Customers. Тобто, вийде перехресне з'єднання. Наприклад, в Orders три рядки, а в Customers той же три рядки, значить ми отримаємо 3 \* 3 = 9 рядків:



Тобто в даному випадку ми отримуємо пряме (декартово) твір двох груп. Але навряд чи це той результат, який хотілося б бачити. Тим більше кожне замовлення з Orders пов'язаний з конкретним покупцем з Customers, а не з усіма можливими покупцями.

Щоб вирішити задачу, необхідно використовувати вираз WHERE і фільтрувати рядки за умови, що поле CustomerId з Orders відповідає полю Id з Customers:

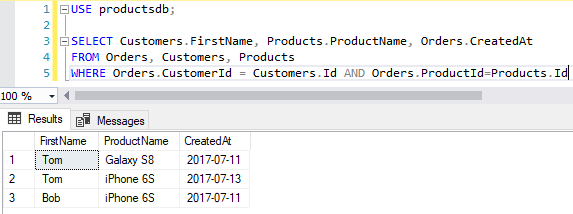
|  |
| --- |
| SELECT \* FROM Orders, Customers  WHERE Orders.CustomerId = Customers.Id |



Тепер об'єднаємо дані по трьом таблиць Orders, Customers і Proucts. Тобто отримаємо всі замовлення і додамо інформацію по клієнту і пов'язаного товару:

|  |
| --- |
| SELECT Customers.FirstName, Products.ProductName, Orders.CreatedAt  FROM Orders, Customers, Products  WHERE Orders.CustomerId = Customers.Id AND Orders.ProductId = Products.Id |

Оскільки треба з'єднати три таблиці, то застосовуються як мінімум дві умови. Ключовий таблицею залишається Orders, з якої витягуються всі замовлення, а потім до неї приєднується дані по клієнту за умовоюOrders.CustomerId = Customers.Id і дані по товару за умовою Orders.ProductId = Products.Id



Оскільки в даному випадку назви таблиць сильно збільшують код, то ми його можемо скоротити за рахунок використання псевдонімів таблиць:

|  |
| --- |
| SELECT C.FirstName, P.ProductName, O.CreatedAt  FROM Orders AS O, Customers AS C, Products AS P  WHERE O.CustomerId = C.Id AND O.ProductId = P.Id |

Якщо необхідно при використанні псевдоніма вибрати всі стовпці з певної таблиці, то можна використовувати зірочку:

|  |
| --- |
| SELECT C.FirstName, P.ProductName, O. \*  FROM Orders AS O, Customers AS C, Products AS P  WHERE O.CustomerId = C.Id AND O.ProductId = P.Id |

# INNER JOIN

Для явного з'єднання даних з двох таблиць застосовується оператор JOIN. Загальний формальний синтаксис застосування оператора INNER JOIN:

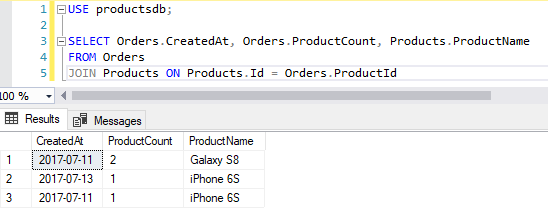
|  |
| --- |
| SELECT стовпці  FROM Таблиця 1  [INNER] JOIN Таблиця 2  ON условіе1  [[INNER] JOIN Табліца3  ON условіе2] |

Після оператора JOIN йде назва другої таблиці, з якої треба додати дані до вибірки. Перед JOIN може використовуватися необов'язкове ключове слово INNER. Його наявність або відсутність ні на що не впливає. Потім після ключового слова ON вказується умова з'єднання. Ця умова встановлює, як дві таблиці будуть порівнюватися. У більшості випадків для з'єднання застосовується первинний ключ головної таблиці і зовнішній ключ залежною таблиці.

Використовуючи JOIN, виберемо всі замовлення і додамо до них інформацію про товари:

|  |
| --- |
| SELECT Orders.CreatedAt, Orders.ProductCount, Products.ProductName  FROM Orders  JOIN Products ON Products.Id = Orders.ProductId |

Оскільки таблиці можуть містити стовпці з однаковими назвами, то при вказівці стовпців для вибірки вказується їх повне ім'я разом з ім'ям таблиці, наприклад, "Orders.ProductCount".

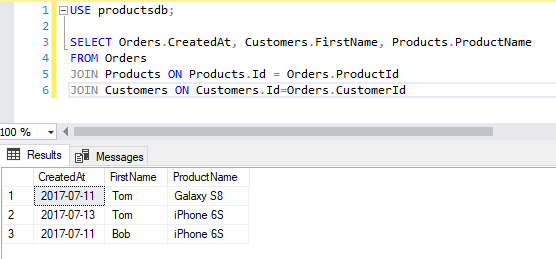


Також використовуючи псевдоніми, ми можемо скоротити код:

|  |
| --- |
| SELECT O.CreatedAt, O.ProductCount, P.ProductName  FROM Orders AS O  JOIN Products AS P  ON P.Id = O.ProductId |

Подібним чином ми можемо приєднувати і інші таблиці. Наприклад, додамо до замовлення інформацію про покупця з таблиці Customers:

|  |
| --- |
| SELECT Orders.CreatedAt, Customers.FirstName, Products.ProductName  FROM Orders  JOIN Products ON Products.Id = Orders.ProductId  JOIN Customers ON Customers.Id = Orders.CustomerId |



Завдяки поєднанню таблиць ми можемо використовувати їх стовпці для фільтрації вибірки або її сортування:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | SELECT Orders.CreatedAt, Customers.FirstName, Products.ProductName  FROM Orders  JOIN Products ON Products.Id = Orders.ProductId  JOIN Customers ON Customers.Id = Orders.CustomerId  WHERE Products.Price <45000  ORDER BY Customers.FirstName |

Умови після ключового слова ON можуть бути більш складними за складом:

|  |  |
| --- | --- |
| 1  2  3  4  5 | SELECT Orders.CreatedAt, Customers.FirstName, Products.ProductName  FROM Orders  JOIN Products ON Products.Id = Orders.ProductId AND Products.Manufacturer = 'Apple'  JOIN Customers ON Customers.Id = Orders.CustomerId  ORDER BY Customers.FirstName |

В даному випадку вибираємо всі замовлення на товари, виробником яких є Apple.

При використанні оператора JOIN слід враховувати, що процес з'єднання таблиць може бути ресурсоємним, тому слід з'єднувати тільки ті таблиці, дані з яких дійсно необхідні. Чим більше таблиць з'єднується, тим більше знижується продуктивність.

# OUTER JOIN

MS SQL Server також підтримує зовнішнє з'єднання або outer join. На відміну від inner join зовнішнє з'єднання повертає всі рядки однієї або двох таблиць, які беруть участь в з'єднанні.

Outer Join має наступний формальний синтаксис:

|  |  |
| --- | --- |
| 1  2  3  4 | SELECT стовпці  FROM Таблиця 1  {LEFT | RIGHT | FULL} [OUTER] JOIN таблица2 ON условіе1  [{LEFT | RIGHT | FULL} [OUTER] JOIN Табліца3 ON условіе2] ... |

Перед оператором JOIN вказується одне з ключових слів LEFT, RIGHT або FULL, які визначають тип з'єднання:

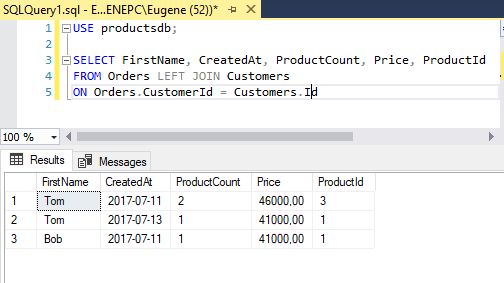
* **LEFT**: Вибірка буде містити всі рядки з першої або лівої таблиці
* **RIGHT**: Вибірка буде містити всі рядки з другої або правої таблиці
* **FULL**: Вибірка буде містити всі рядки з обох таблиць

Також перед оператором JOIN може вказуватися ключове слово OUTER, але його застосування необов'язково. Далі після JOIN вказується приєднується таблиця, а потім йде умова з'єднання.

Наприклад, з'єднаємо таблиці Orders і Customers:

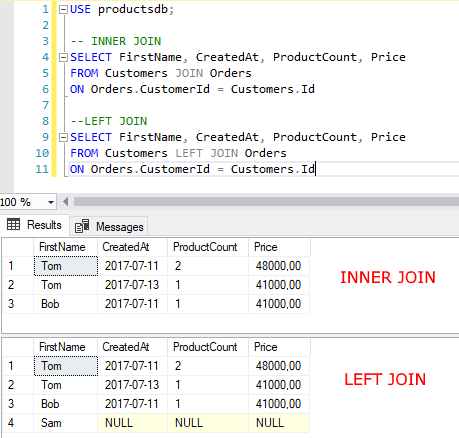
|  |  |
| --- | --- |
| 1  2  3 | SELECT FirstName, CreatedAt, ProductCount, Price, ProductId  FROM Orders LEFT JOIN Customers  ON Orders.CustomerId = Customers.Id |

Таблиця Orders є першою або лівої таблицею, а таблиця Customers - правої таблицею. Тому, так як тут використовується вибірка по лівій таблиці, то спочатку будуть вибиратися всі рядки з Orders, а потім до них за умовоюOrders.CustomerId = Customers.Idдодаватимуться пов'язані рядки з Customers.



За вищенаведеним результату може здатися, що лівосторонній з'єднання аналогічно INNER Join, але це не так. Inner Join об'єднує рядки з дух таблиць при відповідно умові. Якщо одна з таблиць містить рядки, які не відповідають цій умові, то дані рядка не включаються у вихідну вибірку. Left Join вибирає всі рядки першої таблиці і потім приєднує до них рядки правої таблиці. Наприклад, візьмемо таблицю Customers і додамо до покупців інформацію про їх замовленнях:

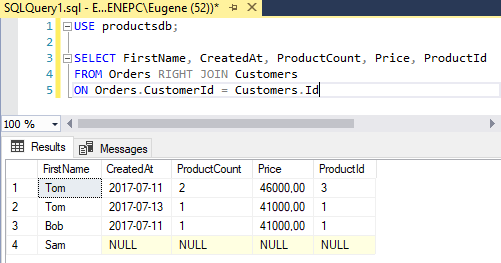
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | - INNER JOIN  SELECT FirstName, CreatedAt, ProductCount, Price  FROM Customers JOIN Orders  ON Orders.CustomerId = Customers.Id  --LEFT JOIN  SELECT FirstName, CreatedAt, ProductCount, Price  FROM Customers LEFT JOIN Orders  ON Orders.CustomerId = Customers.Id |



Змінимо в прикладі вище тип з'єднання на правосторонній:

|  |  |
| --- | --- |
| 1  2  3 | SELECT FirstName, CreatedAt, ProductCount, Price, ProductId  FROM Orders RIGHT JOIN Customers  ON Orders.CustomerId = Customers.Id |

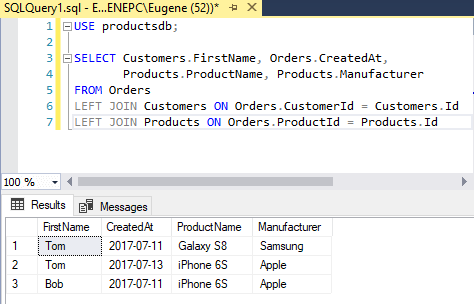
Тепер будуть вибиратися всі рядки з Customers, а до них вже буде приєднуватися пов'язані з умовою рядки з таблиці Orders:



Оскільки один з покупців з таблиці Customers не має пов'язаних замовлень з Orders, то відповідні стовпці, які беруться з Orders, матимуть значення NULL.

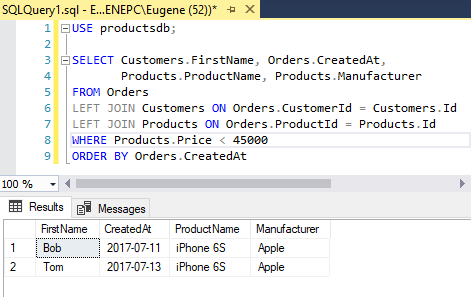
Використовуємо лівосторонній з'єднання для додавання до замовлень інформації про користувачів і товари:

|  |
| --- |
| SELECT Customers.FirstName, Orders.CreatedAt,  Products.ProductName, Products.Manufacturer  FROM Orders  LEFT JOIN Customers ON Orders.CustomerId = Customers.Id  LEFT JOIN Products ON Orders.ProductId = Products.Id |



І також можна застосовувати більш комплексні умови з фільтрацією і сортуванням. Наприклад, виберемо всі замовлення з інформацією про клієнтів і товари по тих товарах, у яких ціна менше 45000, і відсортуємо по даті замовлення:

|  |
| --- |
| SELECT Customers.FirstName, Orders.CreatedAt,  Products.ProductName, Products.Manufacturer  FROM Orders  LEFT JOIN Customers ON Orders.CustomerId = Customers.Id  LEFT JOIN Products ON Orders.ProductId = Products.Id  WHERE Products.Price <45000  ORDER BY Orders.CreatedAt |



Або виберемо всіх користувачів з Customers, у яких немає замовлень в таблиці Orders:

|  |  |
| --- | --- |
| 1  2  3 | SELECT FirstName FROM Customers  LEFT JOIN Orders ON Customers.Id = Orders.CustomerId  WHERE Orders.CustomerId IS NULL |

Також можна комбінувати Inner Join і Outer Join:

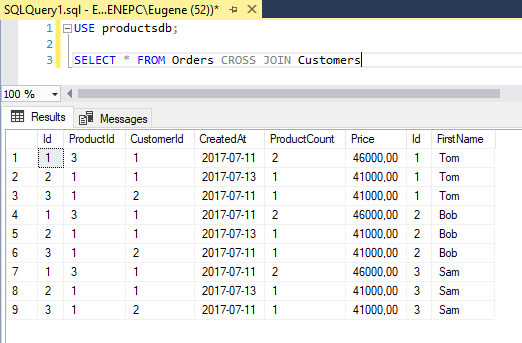
|  |  |
| --- | --- |
| 1  2  3  4  5  6 | SELECT Customers.FirstName, Orders.CreatedAt,  Products.ProductName, Products.Manufacturer  FROM Orders  JOIN Products ON Orders.ProductId = Products.Id AND Products.Price <45000  LEFT JOIN Customers ON Orders.CustomerId = Customers.Id  ORDER BY Orders.CreatedAt |

Спочатку за умовою до таблиці Orders через Inner Join приєднується пов'язана інформація з Products, потім через Outer Join додається інформація з таблиці Customers.

**Cross Join**

Cross Join або перехресне з'єднання створює набір рядків, де кожен рядок з однієї таблиці з'єднується з кожним рядком з другої таблиці. Наприклад, з'єднаємо таблицю замовлень Orders і таблицю покупців Customers:

|  |  |
| --- | --- |
| 1 | SELECT \* FROM Orders CROSS JOIN Customers |



Якщо в таблиці Orders 3 рядки, а в таблиці Customers той же три рядки, то в результаті перехресного з'єднання створюється 3 \* 3 = 9 рядків незалежно, чи пов'язані ці рядки чи ні.

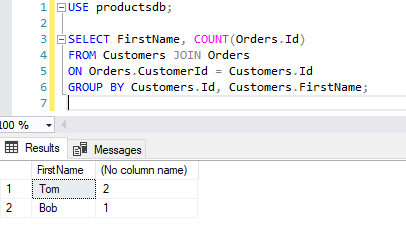
При неявному перехресному з'єднанні можна опустити оператор CROSS JOIN і просто перерахувати всі одержувані таблиці:

|  |  |
| --- | --- |
| 1 | SELECT \* FROM Orders, Customers |

# Груповання в з'єднаннях

У виразах INNER / OUTER JOIN також можна використовувати груповання. Наприклад, виведемо для кожного користувача кількість замовлень, які він зробив:

|  |  |
| --- | --- |
| 1  2  3  4 | SELECT FirstName, COUNT (Orders.Id)  FROM Customers JOIN Orders  ON Orders.CustomerId = Customers.Id  GROUP BY Customers.Id, Customers.FirstName; |

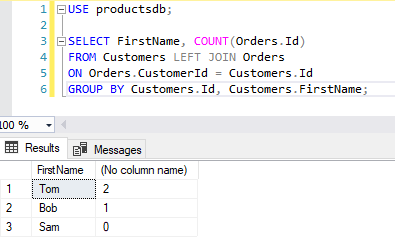


Критерієм груповання виступають Id і ім'я покупця. Вираз SELECT вибирає ім'я покупця і кількість замовлень, використовуючи стовпець Id з таблиці Orders.

Так як це INNER JOIN, то в групах будуть тільки ті покупці, у яких є замовлення.

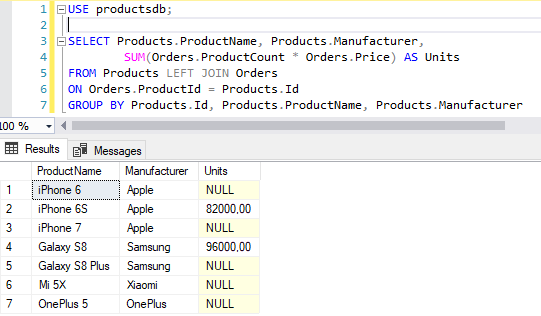
Якщо необхідно вивести навіть тих покупців, у яких немає замовлень, то застосовується OUTER JOIN:

|  |  |
| --- | --- |
| 1  2  3  4 | SELECT FirstName, COUNT (Orders.Id)  FROM Customers LEFT JOIN Orders  ON Orders.CustomerId = Customers.Id  GROUP BY Customers.Id, Customers.FirstName; |



Або виведемо товари із загальною сумою зроблених замовлень:

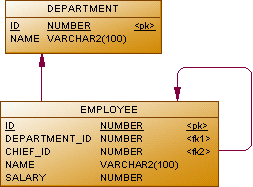
|  |
| --- |
| SELECT Products.ProductName, Products.Manufacturer,  SUM (Orders.ProductCount \* Orders.Price) AS Units  FROM Products LEFT JOIN Orders  ON Orders.ProductId = Products.Id  GROUP BY Products.Id, Products.ProductName, Products.Manufacturer |



# Завдання

1. Реалізувати всі запити з роботи
2. Написати запит, щоб побачити, що купляв кожен з замовників (id замовника, ім’я замовника, назва товару).
3. Написати запит, щоб показати, товари яких компаній купляли користувачі.
4. Написати запит, щоб побачити суму замовлень по кожному замовнику (id замовника, ім’я замовника, сума).
5. Вивести сумарну кількість продажів товарів фірми Apple (кількість і сума) за місяцами.
6. Вивести суми, які витратили користувачі у 2021 році (id замовника, ім’я замовника, сума).
7. Показати, всіх користувачів з товарами, які вони купляли, і кількістю цих товарів.
8. Відобразити товари, які продавались на суму більше 100000 грн. в місяць.
9. Вивести сумарну кількість продажів товарів фірми ‘Apple’ (кількість і сума) за місяцами.
10. Знайти покупця, який витратив більше всього грошей у поточному місяці.
11. \*Написати запит, щоб побачити, який замовник/ки придбав більше всього товарів.
12. \*Хто з користувачів у 2021-му році замовив товарів Apple на найбільшу суму?
13. \*Які товари не продавав оператор з id=1 у 2021-му році.

# Додаткове завдання

Розгорнути схему, яка містить наступні таблиці:  


скласти SQL-запити, для вирішення наступних завдань:

**1.**Вивести список співробітників, які отримують заробітну плату більшу ніж у безпосереднього керівника

**2.**Вивести список співробітників, які отримують максимальну заробітну плату в своєму відділі

**3.**Вивести список ID відділів, кількість співробітників в яких не перевищує 3 осіб

**4.**Вивести список співробітників, які не мають призначеного керівника, який працює в тому-ж відділі

**5.** Знайти список ID відділів з максимальною сумарною зарплатою співробітників

**6\*.**Вивести підсумкову таблицю з усіма департаментами з урахуванням їх підпорядкування при відомому рівні підпорядкованості (рекомендується внести додаткове поле рівень вкладеності) і невідомому рівні вкладеності\* (розраховувати його на льоту - працює не для всіх структур, вимагає циклічного повторення).