

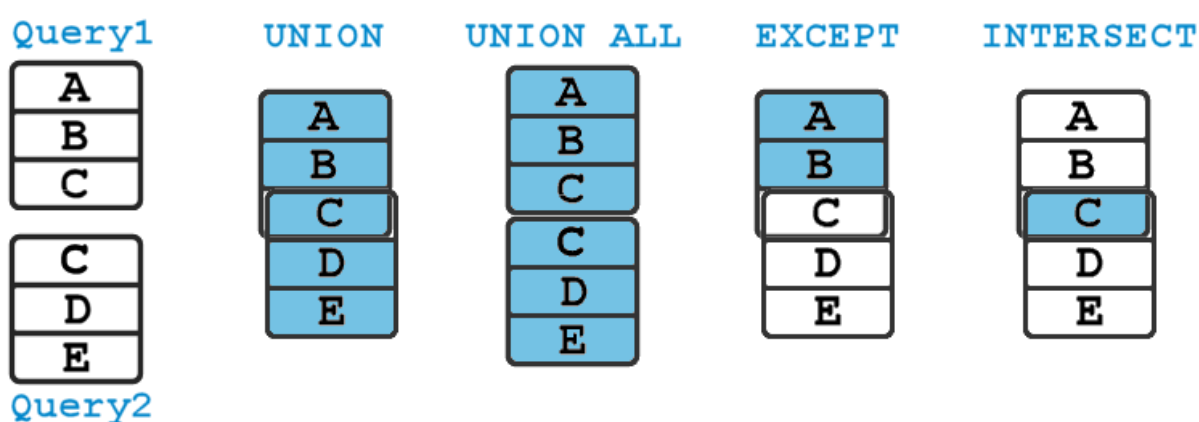
Tematy zajęć:

- 35. Łączenie wyników zapytań-UNION, INTERSECT, EXCEPT.
- 36. Podzapytania.

Omówienie materiału na podstawie bazy Northwind

35. Łączenie wyników zapytań-UNION, INTERSECT, EXCEPT:

Pisząc zapytania *SELECT* czasami zachodzi potrzeba złączenia pobranych wyników. W *SQL* mamy do dyspozycji operatory takie jak: suma (*UNION*), różnica (*EXCEPT*) oraz część wspólną (*INTERSECT*). Przyjrzyjmy się więc jak je używać...



Suma

Wyniki danego zapytania możemy „dodać” do siebie za pomocą operatora *UNION*. Działa on na zasadzie dodawania zbiorów. Wyświetlane są więc wszystkie wiersze.

Przykład użycia:

```
SELECT kol1 FROM tab1
```

```
UNION
```

```
SELECT kol1 FROM tab2;
```

Operator *UNION* powoduje usunięcie z wyniku wszystkich powtórzonych wierszy, można to ominąć stosując w miejsce *UNION* operator *UNION ALL*:

```
SELECT kol1 FROM tab1
```

```
UNION ALL
```

```
SELECT kol1 FROM tab2;
```

Powyższy operator możemy oczywiście użyć kilka razy. Dopuszczalny jest między innymi taki zapis:

```
SELECT kol1 FROM tab1
```

UNION

SELECT kol1 FROM tab2;

UNION

SELECT kol1 FROM tab3

UNION

SELECT kol1 FROM tab4;

Różnica

Operator **EXCEPT** zwraca różnicę pobieranych danych. Działa on na zasadzie różnicy zbiorów czyli wyświetla tylko te wiersze które były w pierwszej tabeli, a nie były w drugiej.

SELECT kol1 FROM tab1

EXCEPT

SELECT kol2 FROM tab2;

Część wspólna

Część wspólną możemy uzyskać za pomocą operatora **INTERSECT**. Zwraca on wiersze które zostały zwrócone przez oba zapytania:

SELECT kol1 FROM tab1

INTERSECT

SELECT kol2 FROM tab2;

Warunki operacji na zbiorach:

Jest kilka zasad, które muszą być spełnione. Warunkiem podstawowym, któregośkolwiek ze sposobów operowania na zbiorach w sposób pionowy, jest podobna struktura tabel wejściowych. **Liczba kolumn w każdym zbiorze (kwerendzie), musi być identyczna oraz typy danych poszczególnych kolumn, muszą do siebie pasować.** Nazwy kolumn, nie mają znaczenia. W zbiorze wynikowym, atrybuty będą nazwane tak jak w pierwszej z kwerend.

Możemy wykonywać wiele operacji na zbiorach, np. złączenie trzech wyników kwerend w jeden zbiór :

-- kwerenda pierwsza (zbiór elementów)

SELECT 'Pierwszy' as Opis, getdate() as Dt, 132 as liczba

UNION -- operator łączenia zbiorów

-- kwerenda druga (zbiór elementów)

SELECT 'Drugi' as ZupelnieInnyOpis, '2020-01-01' as DataZlecenia, 0.2

UNION -- łączenie zbiorów, połączy wynik pierwszego UNION z kwerendą trzecią

```
-- kwerenda trzecia (zbiór elementów)
```

```
SELECT 'Trzeci' as Opisik, '2020-11-21' as dt, 0
```

	Opis	Dt	liczba
1	Drugi	2020-01-01 00:00:00.000	0.2
2	Pierwszy	2021-01-07 20:57:19.990	132.0
3	Trzeci	2020-11-21 00:00:00.000	0.0

Jak widzimy w tym przykładzie, wszystkie trzy „kwerendy” zwracają po jednym elemencie opisanym 3 atrybutami. Niektóre z tych atrybutów są innego typu (np. 132 z pierwszego zbioru to typ integer, wartość 0.2 w drugim to decimal), ale są to typy kompatybilne (możliwa jest ich bezstratna niejawna konwersja). Jeśli spróbowałibyśmy, połączyć zbiory o różnych typach w tych samych kolumnach, np. :

```
-- wartości w kolumnie 3 są typu integer
select 'Pierwszy' as Opis, getdate() as Data, 132 as liczba
UNION
-- z kolei tutaj w 3 kolumnie mamy tekst
```

```
select 'Drugi', '2020-01-01', 'sto dwa'
```

otrzymamy komunikat o błędzie :

```
Msg 245, Level 16, State 1, Line 14
Conversion failed when converting the varchar value 'sto dwa' to data type int.
```

Podobnie jeśli liczba kolumn nie będzie równa :

```
select 'Pierwszy' as Opis, getdate() as Data, 132 as liczba
union
select 'Drugi', '20120-01-01'
```

Tym razem dostaniemy info o różnej liczbie kolumn w zbiorach, które mają być łączone :

```
Msg 205, Level 16, State 1, Line 19
All queries combined using a UNION, INTERSECT or EXCEPT operator must have an equal number of
expressions in their target lists.
```

Trzeba pamiętać o tej podstawowej zasadzie i zadbać o to, aby była ona zawsze spełniona. Zazwyczaj nie jest to trudne, bo jeśli masz dwie kwerendy, które chcesz połączyć i zawierają one różne liczby kolumn, wystarczy sztucznie „uzupełnić” brakującą liczbę, np. wartościami NULL :

```
select kol1, kol2, kol3
From tabela1
union
select kol1, kol2, NULL

from tabela2
```

Podobnie jeśli typy danych nie są kompatybilne, zawsze można je zmienić (np. stosując funkcję CAST lub CONVERT), na typ bardziej ogólny – np. na varchar.

Przykład użycia operatora UNION – łączenie zbiorów

UNION oznacza sumę zbiorów. W wyniku otrzymamy elementy znajdujące się zarówno w zbiorze pierwszym jak i drugim, ale domyślnie jest to operacja UNION DISTINCT, czyli z usunięciem wszystkich duplikatów. W szczególności znajdujących się jako część wspólna zbiorów, a także duplikatów istniejących w ramach tabel wejściowych.

Use Northwind

```
-- pierwsze zapytanie zwraca 9 elementów (niektóre się powtarzają)
select Country from Employees
where Country like 'U%'
UNION-- drugie zapytanie zwraca 20 elementów (niektóre się powtarzają)
select Country from Customers where Country like 'U%'
```

Results		Messages
	Country	
1	UK	
2	USA	

Każde z zapytań biorących udział w operacji łączenia zbiorów zwraca w wyniku zdublowane wartości elementów (nie usuwamy ich za pomocą DISTINCT), robi to domyślnie operator UNION – czyli dostajemy tylko unikalne wartości elementów zbioru (A + B) – USA i UK. Usunięte są one zarówno z kwerend wejściowych, jak i z części wspólnej zbioru A i B).

Przykład użycia operatora UNION ALL – łączenie bez usuwania duplikatów

Drugim sposobem na dodawanie zbiorów jest UNION ALL – czyli bez usuwania duplikatów. Tym razem z każdego zbioru, bierzemy tylko 5 pierwszych wierszy (TOP 5) i pomimo, że elementy się powtarzają, w wyniku dostaniemy 10 wierszy.

Use Northwind

```
-- pierwsze zapytanie zwraca 9 elementów (niektóre się powtarzają)
-- tym razem bierzemy tylko 5 pierwszych - TOP 5
select top 5 Country
from Employees
where Country like 'U%'
UNION ALL-- drugie zapytanie zwraca 20 elementów (niektóre się powtarzają)
-- tym razem bierzemy tylko 5 pierwszych - TOP 5
select top 5 Country
from Customers
where Country like 'U%'
```

Results		Messages
	Country	
1	USA	
2	USA	
3	USA	
4	USA	
5	UK	
6	UK	
7	UK	
8	UK	
9	UK	
10	USA	

Przykład użycia operatora EXCEPT – odejmowanie zbiorów

Zasada działania jest prosta. Ze zbioru pierwszego (czyli po lewej stronie od operatora EXCEPT), odejmowane są wszystkie elementy wspólne ze zbiorem drugim (tabeli wynikowej, kwerendy po prawej stronie).

Odejmowanie zbiorów za pomocą EXCEPT zostało zaimplementowane w SQL Server tylko jako EXCEPT DISTINCT, czyli w zbiorze wynikowym, zawsze usuwane są wszystkie duplikaty rekordów. Przykład :

Use Northwind

```
-- pierwsze zapytanie zwraca 4 miasta - Seattle, Tacoma, Kirkland i Redmond
select city from Employees
where Country = 'USA'
```

```
EXCEPT -- operator odejmowania zbiorów drugie zapytanie zwraca znacznie
więcej miast, wśród nich są Seattle i Kirkland
select city from Customers
where Country = 'USA'
```

	city
1	Redmond
2	Tacoma

Kolejność Except odejmowania –jak w matematyce ma znaczenie:

```
select city from Customers
where Country = 'USA'
```

```
EXCEPT
select city from Employees
where Country = 'USA'
```

	city
1	Albuquerque
2	Anchorage
3	Boise
4	Butte
5	Elgin
6	Eugene
7	Lander
8	Portland
9	San Francisco
10	Walla Walla

Przykład użycia operatora INTERSECT – część wspólna zbiorów

Do wyznaczenia części wspólnej zbiorów, używamy operatora **INTERSECT**. Podobnie jak EXCEPT, zaimplementowany w SQL Server, został również tylko jako INTERSECT DISTINCT, czyli części wspólna dwóch zbiorów z usunięciem duplikatów.

```
-- Q1 pierwsze zapytanie zwraca 4 miasta - Seattle (dwa razy), Tacoma,
Kirkland i Redmond
select city
```

```

from Employees
where Country = 'USA'
INTERSECT-- Q2 drugie zapytanie zwraca znacznie więcej,
-- ale wśród nich są też Seattle (tylko raz) i Kirkland
select city
from Customers
where Country = 'USA'

```

Results		Message
	city	
1	Kirkland	
2	Seattle	

Kolejność wykonywania operacji

Możemy operować na wielu zbiorach w ramach jednej kwerendy i stosować różne operacje. Obowiązuje tutaj kolejność wykonywania działań – dokładnie tak jak w matematyce.

```

Select kol1, kol2, kol3 from tabela1
UNION
Select kol1, kol2, kol3 from tabela2
EXCEPT
Select kol1, kol2, kol3 from tabela3
INTERSECT
(
Select kol1, kol2, kol3 from tabela4
UNION
Select kol1, kol2, kol3 from tabela5
)

```

Najpierw zostaną wykonane działania w nawiasach, czyli UNION ostatnich dwóch kwerend (wyciągających dane z tabeli 4 i 5). Następnie mnożenie zbiorów, czyli INTERSECT – część wspólna pomiędzy wynikiem wyznaczonym w kroku pierwszym a kwerendą wyciągającą dane z tabeli3.

W końcu, jeśli nie ma już żadnych nawiasów i iloczynów, zostaną wykonane wszystkie pozostałe kroki od lewej do prawej, czyli w tym przypadku najpierw, pierwszy UNION i w końcu EXCEPT.

Poprzez stosowanie nawiasów, mamy pełną kontrolę nad logiczną kolejnością wykonywania działań.

36. Podzapytania.

Podzapytanie to zapytanie SQL, które umieszczone jest wewnątrz innego zapytania. Podzapytanie zawsze otoczone jest parą nawiasów ().

```

SELECT *
FROM (SELECT 1);

```

Podzapytania, jak sama nazwa wskazuje, są częścią podrzędną innego zapytania. Podzapytania najczęściej umieszcza się w warunkach w klauzulach WHERE i HAVING zapytania zewnętrznego, niektóre SZBD dopuszczają również stosowanie podzapytań w klauzulach SELECT i FROM. Możemy podzielić je na dwie kategorie ze względu na powiązanie z kwerendą nadrzędną :

- **niezależne** – funkcjonować mogą w całkowicie oderwanym kontekście. Można je uruchomić jako osobne kwerendy
- **skorelowane** – są bezpośrednio powiązane z zapytaniem nadrzędnym.
- Wykorzystamy podzapytanie skorelowane aby „dokleić” dodatkową informację o liczbie zleceń i pokazać w praktyce ich sposób działania :

Przykład podzapytania klauzuli WHERE

Aby wyświetlić tytuły oraz autorów książek, których cena jest wyższa od średniej ceny książek znajdujących się w bazie, należy zastosować polecenie:

```
USE BIBLIOTEKA
SELECT tytuł, imię_autora, nazwisko_autora, cena FROM książki
WHERE cena > (SELECT AVG(cena) FROM książki);
```

Przykład podzapytania klauzuli FROM

Aby wyświetlić tytuły oraz ceny książek, których rok wydania jest mniejszy lub równy 2000, należy zastosować polecenie:

```
USE BIBLIOTEKA
SELECT Stareksiazki.tytuł, Stareksiazki.cena FROM
(SELECT tytuł, cena, rokwydania FROM książki WHERE rokwydania <=2000) AS
Stareksiazki;
```

Przykład: zlicza numery zamówień i dla nich wyświetla nr_klientów i nazwę firmy, Sortuje po największej liczbie zleceń.

```
SELECT CustomerID, CompanyName,
(
    -- podzapytanie skorelowane
    SELECT COUNT(OrderID)
    FROM Orders as O
    WHERE O.CustomerID = C.CustomerID    -- faktyczna korelacja
) as LiczbaZleceń
FROM Customers as C
ORDER BY LiczbaZleceń desc
```

Będzie to pierwszy przykład z wykorzystaniem typowego podzapytania niezależnego we FROM :

```

SELECT *
FROM
(
    -- wstępna, selekcja elementów i atrybutów zbioru Customers może tu być
    dowolna skomplikowana kwerenda.
    SELECT CompanyName, City, Country FROM Customers where City =
    'London'
) AS MojePodzapytanie

WHERE CompanyName like '[A-C]%'

```

Results		Messages	
	CompanyName	City	Country
1	Around the Horn	London	UK
2	B's Beverages	London	UK
3	Consolidated Holdings	London	UK

W każdej chwili można to podzapytanie uruchomić zaznaczając tylko jego zakres. Jest ono niezależne w stosunku do zapytania zewnętrznego. Wykonane zostanie raz, w trakcie całego procesu logicznego przetwarzania tej kwerendy.

Każdy zbiór do którego odnosimy się we FROM musi być nazwany i w pełni określony. Stąd konieczność stosowania aliasów oraz unikalnych nazw kolumn w ramach podzapytań.

OPERATORY PODZAPYTANÍ:

Nazwa	Przykład	Opis
ANY	SELECT k1 FROM t1 WHERE k1 > ANY (SELECT k1 FROM t2)	Zwraca true, jeżeli porównanie będzie miało wartość true dla dowolnego wiersza z podzapytania
IN	SELECT k1 FROM t1 WHERE k1 IN ANY (SELECT k1 FROM t2)	Równoważnik konstrukcji SELECT k1 FROM t1 WHERE k1 = ANY (SELECT k1 FROM t2)
SOME	SELECT k1 FROM t1 WHERE k1 SOME (SELECT k1 FROM t2)	Alias dla ANY. Wkraczamy w niuanse języka angielskiego. Widocznie niektórzy wolą stosować ANY, a niektórzy SOME
ALL	SELECT k1 FROM t1 WHERE k1 ALL (SELECT k1 FROM t2)	Zwraca true, jeżeli porównanie będzie miało wartość true dla wszystkich wierszy z podzapytania.
EXISTS	SELECT klienci.nazwisko FROM klienci WHERE EXISTS (SELECT * FROM zamówienia WHERE klienci.id_klienta= zamowienia.id_zamowienia)	Zwraca true lub false. Jeśli zapytanie wewnętrzne wyświetli jakiegokolwiek wartości, to EXISTS zwróci prawdę, w przeciwnym wypadku fałsz.

Dodatkowy przykład na EXISTS

Operator EXISTS powoduje, że zwrócone są wyłącznie te wiersze, dla których podzapytanie zwróci co najmniej jeden wiersz. Przykład:

```
SELECT *
  FROM Employees AS a
 WHERE EXISTS (SELECT *
               FROM Employees AS b
               WHERE b.ReportsTo = a.EmployeeID);
```

W tym przypadku skorelowane podzapytanie zwraca wiersze, które połączone są relacją szef-podwładny. Wiersze, które zawierają pracowników nie posiadających podwładnych są pominięte. Dzieje się tak dlatego, że podzapytanie w ich przypadku nie zwróci ani jednego wiersza.

Przykłady podzapytań:

--1. Dla każdego produktu podaj maksymalną liczbę zamówionych jednostek

```
SELECT t1.ProductName, t2.maks
from
(SELECT ProductName, ProductID
FROM Products) as t1,

(SELECT MAX(Quantity) AS maks, ProductID
FROM [Order Details]
GROUP BY ProductID) as t2
WHERE t1.ProductID=t2.ProductID
```

--2. Podaj wszystkie produkty których cena jest mniejsza niż średnia cena produktu

```
SELECT ProductName, UnitPrice
FROM dbo.Products
WHERE (UnitPrice < (SELECT AVG(UnitPrice) AS srednia
FROM dbo.Products))
```

--3. Podaj wszystkie produkty których cena jest mniejsza niż średnia cena produktu danej kategorii

```
select t1.ProductName ,t1.UnitPrice,t1.CategoryID, t2.srednia from
(
SELECT ProductName, UnitPrice,CategoryID
FROM dbo.Products ) as t1,
```

```
(SELECT CategoryID, AVG(UnitPrice) AS srednia
FROM dbo.Products
GROUP BY CategoryID) as t2
where t1.CategoryID=t2.CategoryID and t1.UnitPrice<t2.srednia
```

--4. Dla każdego produktu podaj jego nazwę, cenę, średnią cenę wszystkich produktów oraz różnicę między ceną produktu a średnią ceną wszystkich produktów

```
select t1.ProductName,t1.UnitPrice,t2.srednia_cena, (t1.UnitPrice-
t2.srednia_cena)as roznica from
(SELECT ProductName, UnitPrice
FROM Products)as t1,
```

```
(SELECT AVG(UnitPrice) AS srednia_cena  
FROM Products) as t2  
--5. Czy są jacyś klienci którzy nie złożyli żadnego zamówienia w 1997 roku,  
jeśli tak to pokaż ich dane adresowe
```

```
select * from customers where CustomerID not in (SELECT CustomerID  
FROM dbo.Orders  
WHERE (YEAR(OrderDate) = 1997))
```

ⁱ <http://www.sqlpedia.pl/operacje-na-zbiorach/>