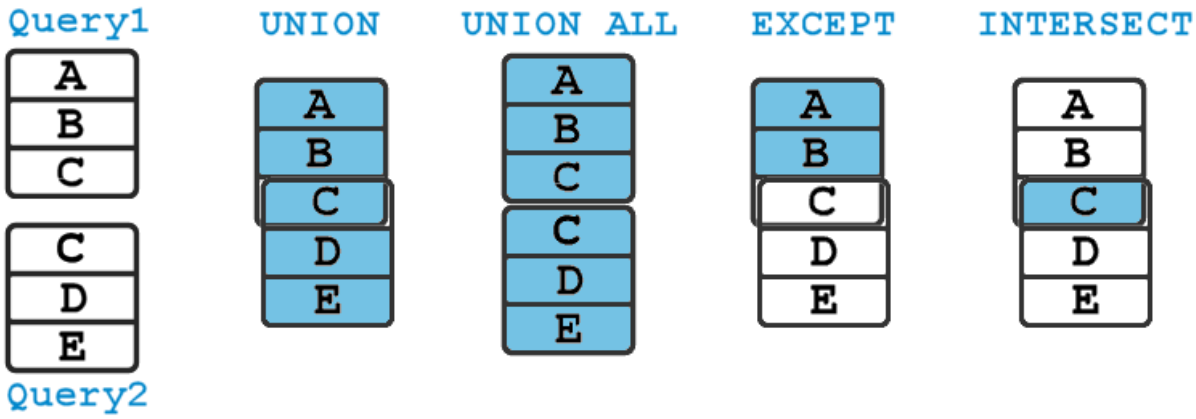
# Łączenie wyników zapytań w SQL

*Pisząc zapytania SELECT czasami zachodzi potrzeba złączenia pobranych wyników. W SQL mamy do dyspozycji operatory takie jak: suma (`UNION`), różnica (`EXCEPT`) oraz cześć 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;
```

1 SELECT kol1 FROM tab1

2 UNION

3 SELECT kol1 FROM tab2;

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

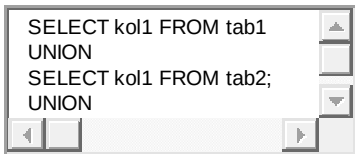
```
SELECT kol1 FROM tab1
UNION ALL
SELECT kol1 FROM tab2;
```

1 SELECT kol1 FROM tab1

2 UNION ALL

3 SELECT kol1 FROM tab2;

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




1 SELECT kol1 FROM tab1

2 UNION

3 SELECT kol1 FROM tab2;

4 UNION

5 SELECT kol1 FROM tab3

6 UNION

7 SELECT kol1 FROM tab4;

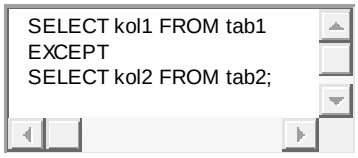
## Przykład:

```
use northwind
select CompanyName, ContactName, City, Country from Customers where
Country='UK'
UNION ALL

select CompanyName, ContactName, City, Country from Customers where
Country='USA' OR Country='UK'
```

## 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.



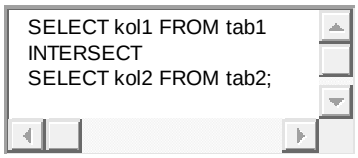

1 SELECT kol1 FROM tab1

2 EXCEPT

3 SELECT kol2 FROM tab2;

## Część wspólna

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




1 SELECT kol1 FROM tab1

2 INTERSECT

3 SELECT kol2 FROM tab2;

### Przykład:

```
use northwind
select CompanyName, ContactName, City, Country from Customers where
Country='USA'OR Country='UK'
INTERSECT
select CompanyName, ContactName, City, Country from Customers where
Country='UK'
```

### Warunki operacji na zbiorach

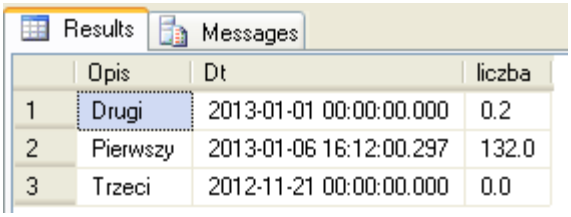
Jest kilka zasad, które muszą być spełnione. Warunkiem podstawowym, któregokolwiek ze sposobów operowania na zbiorach w sposób pionowy, jest podobna struktura tabel wejściowych.
**Liczba kolumn w każdym zbiorze (kwerendzie), musi być identyczna oraz typy danych poszczególnych kolumn, muszą do siebie 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, '2013-01-01' as DataZlecenia, 0.2

UNION  -- łączenie zbiorów, połączy wynik pierwszego UNION z kwerendą
trzecią
-- kwerenda trzecia (zbiór elementów)
SELECT 'Trzeci' as Opisik, '2012-11-21' as dt, 0
```



Results | Messages

| | Opis | Dt | liczba |
|---|---|---|---|
| 1 | Drugi | 2013-01-01 00:00:00.000 | 0.2 |
| 2 | Pierwszy | 2013-01-06 16:12:00.297 | 132.0 |
| 3 | Trzeci | 2012-11-21 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 spróbowalibyśmy, 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', '2013-01-01', 'sto dwa'
```

otrzymamy komunikat o błędzie :

```
Msg 245, Level 16, State 1, Line 2
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', '2013-01-01'
```

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

```
Msg 205, Level 16, State 1, Line 1
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
```

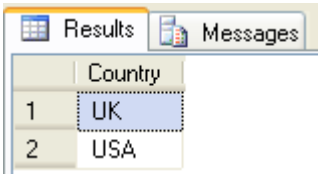
Podobnie jeśli typy danych nie są kompatybilne, zawsze można je zmienić (np. stosując funkcję CAST lub CONVERT), na typ bardziej ogólny – np. na varchar.

### *Operator 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
GO

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

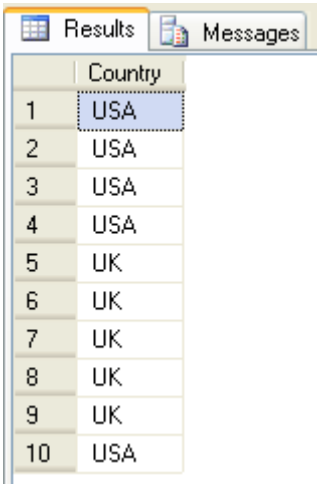



Każde z zapytań biorących udział w operacji łączenia zbiorów zwraca w wyniku zdublowane wartości elementów (nie usuwamy ich za pomocą DISTINCT), robi to domyślnie operator UNION – czyli dostajemy tylko unikalne wartości elementów zbioru (A + B) – USA i UK. Usunięte są one zarówno z kwerend wejściowych,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

### *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 (załatawia to TOP 5) i pomimo, że elementy się powtarzają, w wyniku dostaniemy 10 wierszy.

```
-- pierwsze zapytanie zwraca 9 elementów (niektóre się powtarzają)
-- tym razem bierzemy tylko 5 pierwszych - TOP 5
select top 5 Country
from [dbo].[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 [dbo].[Customers]
where Country like 'U%'
```



| | Country |
|---|---|
| 1 | USA |
| 2 | USA |
| 3 | USA |
| 4 | USA |
| 5 | UK |
| 6 | UK |
| 7 | UK |
| 8 | UK |
| 9 | UK |
| 10 | USA |

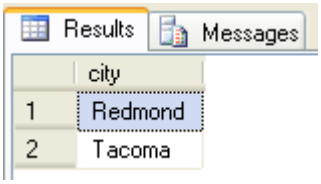
### *EXCEPT – odejmowanie zbiorów*

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 :

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

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



| | city |
|---|---|
| 1 | Redmond |
| 2 | Tacoma |

### *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 [dbo].[Employees]
where Country = 'USA'
INTERSECT-- Q2 drugie zapytanie zwraca znacznie więcej,
```

```
-- ale wśród nich są też Seattle (tylko raz) i Kirkland
select city
from [dbo].[Customers]
where Country = 'USA'
```

### 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 wykonywanania działań.

## Funkcje konwersji typów, CAST i CONVERT

Jednymi z częściej stosowanych skalarnych funkcji wbudowanych są **CAST** oraz **CONVERT**. Służą one do konwersji typów danych na inne (np. liczby na ciąg znaków). Szczególnie pomocne, gdy chcesz np. dokonać konkatenacji (złączenia) atrybutów w ciągi znakowe (stringi) z wartościami typu data czy liczbami.

W T-SQL trzeba zadbać o sensowność operacji pod względem typów danych na któych operujemy. Trudno porównać wartość tekstową ‚Seattle' z liczbą 2. Poniższe dwa zapytania zawierają typowe błędy, związane ze znaczącą różnicą porównywnaych typów czy przekształcacnych danych (składanie stringu).

```
use northwind
go

-- błąd konwersji w warunku WHERE
-- porównanie wartości tekstowej z liczbą
select * from dbo.Employees
where City = 2
```

```
-- błąd konwersji w SELECT
-- próba połączenia stringów z typem danych date/time
select LastName + ' urodzony : ' + BirthDate
from dbo.Employees
Msg 245, Level 16, State 1, Line 3
Conversion failed when converting the nvarchar value 'Seattle' to data type
int.

Msg 241, Level 16, State 1, Line 3
Conversion failed when converting date and/or time from character string.
```

W niektórych przypadkach, można działać na różniących się typach z tej samej rodziny. Wtedy gdy możliwa jest niejawna konwersja – np. wartość liczbowa typu całkowitego oraz zmiennoprzecinkowa. Jeśli jednak chcemy być pewni wyniku, powinniśmy zawsze zatroszczyć się o pełną zgodność typów. Funkcje CAST oraz CONVERT służą właśnie do osiągnięcia tego celu.

---

**CAST**(wartość_konwertowana AS typ_danych) jest podstawową funkcją konwersji zgodną ze standardem ANSI. Jej działanie ogranicza się do bezpośredniej konwersji danej wartości na inny typ danych podany jako drugi parametr funkcji (po słowie kluczowym AS). Z uwagi na jej kompatybilność z ANSI, powinna być stosowana zawsze, gdy nie potrzeba określać styli (np. w konwersji dat).

Jej możliwości są jednak nieco ograniczone w stosunku do CONVERT, ale w wielu sytuacjach nie ma to znaczenia. Możemy stosować je wymiennie.

Konwersję używamy np. do łączenia danych różnego typu, chcąc uzyskać w rezultacie ciąg znakowy.

```
USE Northwind
GO

SELECT LastName + ' hired : ' +  CAST (HireDate as varchar(100)) as
EmpInfo,
       HireDate
FROM dbo.Employees;
```

Results | Messages

| | EmpInfo | HireDate |
|---|---|---|
| 1 | Davolio hired : May 1 1992 12:00AM | 1992-05-01 00:00:00.000 |
| 2 | Fuller hired : Aug 14 1992 12:00AM | 1992-08-14 00:00:00.000 |
| 3 | Leverling hired : Apr 1 1992 12:00AM | 1992-04-01 00:00:00.000 |
| 4 | Peacock hired : May 3 1993 12:00AM | 1993-05-03 00:00:00.000 |
| 5 | Buchanan hired : Oct 17 1993 12:00AM | 1993-10-17 00:00:00.000 |
| 6 | Suyama hired : Oct 17 1993 12:00AM | 1993-10-17 00:00:00.000 |
| 7 | King hired : Jan 2 1994 12:00AM | 1994-01-02 00:00:00.000 |
| 8 | Callahan hired : Mar 5 1994 12:00AM | 1994-03-05 00:00:00.000 |
| 9 | Dodsworth hired : Nov 15 1994 12:00AM | 1994-11-15 00:00:00.000 |

Zauważ, że w tym przypadku wartość daty została przekonwertowana na określony styl.

Funkcja CAST nie posiada możliwości wyboru stylu, dlatego jeśli chcemy otrzymać samą datę (w formacie YYYY-MM-DD), powinniśmy w tej sytuacji skorzystać z funkcji CONVERT ze wskazanym stylem.

---

**CONVERT** (typ_danych, wartość_konwertowana, opcjonalnie_styl) potrafi wszystko to co CAST, dodatkowo możemy określić styl typu danych. Ma on znaczenie zwłaszcza w przypadku konwersji na typy danych związanych z datą i czasem, a także liczbowe (określenie precyzji), XML i binarne. Szczegółowy opis styli znajdziesz tutaj (MSDN).

```
USE Northwind
GO

SELECT 'Hired : ' +  CAST (HireDate as varchar(10)) as konwersjaCast,
        CONVERT(varchar(10),HireDate ,120) as KonwersjaConvertStyles,
        CAST(CAST (HireDate as date) as varchar(10)) as DblCast,
        HireDate
FROM dbo.Employees
```

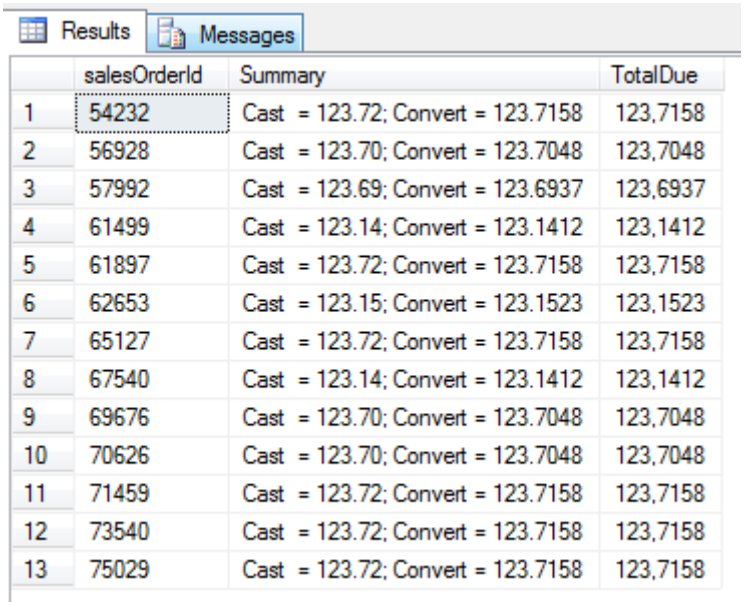
Results | Messages

| | konwersjaCast | KonwersjaConvertStyles | DblCast | HireDate |
|---|---|---|---|---|
| 1 | Hired : May 1 199 | 1992-05-01 | 1992-05-01 | 1992-05-01 00:00:00.000 |
| 2 | Hired : Aug 14 199 | 1992-08-14 | 1992-08-14 | 1992-08-14 00:00:00.000 |
| 3 | Hired : Apr 1 199 | 1992-04-01 | 1992-04-01 | 1992-04-01 00:00:00.000 |
| 4 | Hired : May 3 199 | 1993-05-03 | 1993-05-03 | 1993-05-03 00:00:00.000 |
| 5 | Hired : Oct 17 199 | 1993-10-17 | 1993-10-17 | 1993-10-17 00:00:00.000 |
| 6 | Hired : Oct 17 199 | 1993-10-17 | 1993-10-17 | 1993-10-17 00:00:00.000 |
| 7 | Hired : Jan 2 199 | 1994-01-02 | 1994-01-02 | 1994-01-02 00:00:00.000 |
| 8 | Hired : Mar 5 199 | 1994-03-05 | 1994-03-05 | 1994-03-05 00:00:00.000 |
| 9 | Hired : Nov 15 199 | 1994-11-15 | 1994-11-15 | 1994-11-15 00:00:00.000 |

Podobne operacje musimy wykonać jeśli chcemy łączyć ze sobą wartości liczbowe i tekstowe. Poniższy przykład obnaża po raz kolejny ograniczenia CAST (tym razem związane z zaokrąglaniem wartości).

```
USE AdventureWorks2008
GO

SELECT salesOrderId, 'Cast  = ' + CAST(TotalDue as varchar(100)) +
       '; Convert = ' + CONVERT(varchar(100),TotalDue,2) as Summary,
       TotalDue
FROM Sales.SalesOrderHeader
WHERE TotalDue BETWEEN 123 AND 124
```



Results | Messages

| | salesOrderId | Summary | TotalDue |
|---|---|---|---|
| 1 | 54232 | Cast = 123.72; Convert = 123.7158 | 123,7158 |
| 2 | 56928 | Cast = 123.70; Convert = 123.7048 | 123,7048 |
| 3 | 57992 | Cast = 123.69; Convert = 123.6937 | 123,6937 |
| 4 | 61499 | Cast = 123.14; Convert = 123.1412 | 123,1412 |
| 5 | 61897 | Cast = 123.72; Convert = 123.7158 | 123,7158 |
| 6 | 62653 | Cast = 123.15; Convert = 123.1523 | 123,1523 |
| 7 | 65127 | Cast = 123.72; Convert = 123.7158 | 123,7158 |
| 8 | 67540 | Cast = 123.14; Convert = 123.1412 | 123,1412 |
| 9 | 69676 | Cast = 123.70; Convert = 123.7048 | 123,7048 |
| 10 | 70626 | Cast = 123.70; Convert = 123.7048 | 123,7048 |
| 11 | 71459 | Cast = 123.72; Convert = 123.7158 | 123,7158 |
| 12 | 73540 | Cast = 123.72; Convert = 123.7158 | 123,7158 |
| 13 | 75029 | Cast = 123.72; Convert = 123.7158 | 123,7158 |

Trzeba pamiętać, że w ten sposób wykonywana konwersja, musi być możliwa dla wszystkich wartości w danej kolumnie. W przeciwnym razie, jeśli chodzi dla jednej wartości nie będzie możliwa zmiany typu danych – zapytanie zakończy się niepowodzeniem. Zdarza się tak zazwyczaj w „zabrudzonych" bazach, w których dane są niespójne na wskutek np. importów z pominięciem reguł itp. akcji.

---

W SQL Server 2012 wprowadzono nowe funkcje związane z konwersją typów danych. Szczególnie przydatne to – **TRY_CAST** oraz **TRY_CONVERT**. Ich pojawienie upraszcza „czyszczenie danych".
Funkcje te działają identycznie do CAST oraz CONVERT z tym że dla danych, dla których konwersja nie jest możliwa, zamiast błędu, zwracana jest wartość NULL.

Podzapytania, jak sama nazwa wskazuje, są częścią podrzędną innego zapytania. Możemy podzielić je na dwie kategorie ze względu na powiązanie z kwerendą nadrzędną :

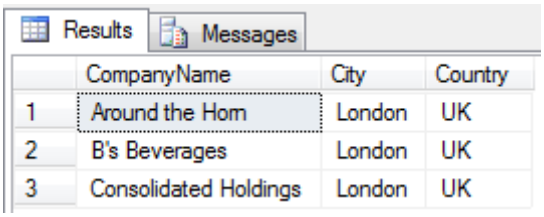
- **niezależne** – funkcjonować mogą w całkowicie oderwanym kontekście. Można je uruchomić jako osobne kwerendy – o nich właśnie jest ten artykuł.
- **skorelowane** – są bezpośrednio powiązane z zapytaniem nadrzędnym.
- Wykorzystamy podzapytanie skorelowane aby „dokleić" dodatkową informację o liczbie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

```
USE Northwind
GO

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

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

```
SELECT *
FROM
(
        -- wstępna, selekcja elementów i atrybutów zbioru dbo.Customers
        -- może tu być dowolna skomplikowana kwerenda.
        SELECT CompanyName, City, Country     FROM dbo.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 |

Zauważ, że w każdej chwili możesz 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.

Przykłady:

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

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

(SELECT MAX(Quantity) AS maks, ProductID
FROM     dbo.[Order Details]
GROUP BY ProductID)as t2
where t1.ProductID=t2.ProductID
--2.  Podaj wszystkie produkty których cena jest mniejsza niż ?rednia 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 ?rednia 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, średniš cenę wszystkich
produktów oraz różnicę między cenš produktu a średniš cenš 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))
```

http://www.sqlpedia.pl/operacje-na-zbiorach/