SQL01+SQL02 - Tworzenie zapytań do bazy danych na platformie MS SQL Server

ALTKOM AKADEMIA

Autor: Tomasz Grabowski

Spis treści

[Podział języka SQL (Structured Query Language) 4](#_Toc536645086)

[Elementy okna oraz dostosowanie opcji Management Studio (SSMS) 4](#_Toc536645087)

[Zagadnienia i pojęcia związane z językiem SQL oraz tworzeniem kwerend 10](#_Toc536645088)

[Tabele, pola, rekordy 10](#_Toc536645089)

[Klucze, relacje 11](#_Toc536645090)

[Więzy integralności 12](#_Toc536645091)

[Procedura składowana 12](#_Toc536645092)

[Importowanie danych 13](#_Toc536645093)

[Podstawowe elementy składni języka SQL 16](#_Toc536645094)

[Wybieranie danych: SELECT 16](#_Toc536645095)

[Źródła danych: FROM 16](#_Toc536645096)

[Sortowanie danych: ORDER BY 16](#_Toc536645097)

[Grupowanie danych: GROUP BY 17](#_Toc536645098)

[Filtrowanie danych: WHERE, HAVING 17](#_Toc536645099)

[Operatory logiczne 18](#_Toc536645100)

[Zapytania w oparciu o różne bazy danych 18](#_Toc536645101)

[Łączenie tabel: JOIN, LEFT JOIN, RIGHT JOIN, OUTER JOIN, UNION, UNION ALL 19](#_Toc536645102)

[UNION 21](#_Toc536645103)

[UNION ALL 22](#_Toc536645104)

[INTERSECT 22](#_Toc536645105)

[EXCEPT 22](#_Toc536645106)

[Funkcje przetwarzania tekstu 23](#_Toc536645107)

[Operacje na datach 27](#_Toc536645108)

[Konstrukcje warunkowe 30](#_Toc536645109)

[CASE 30](#_Toc536645110)

[IF 31](#_Toc536645111)

[Konwersja typów danych 31](#_Toc536645112)

[Zapytania funkcyjne 33](#_Toc536645113)

[Dodawanie danych do tabel: INSERT 33](#_Toc536645114)

[Modyfikowanie istniejących danych: UPDATE 33](#_Toc536645115)

[Usuwanie danych: DELETE 34](#_Toc536645116)

[Kasowanie tabel: DROP TABLE 35](#_Toc536645117)

[Podłączenie MS Excel do danych w SQL Server 36](#_Toc536645118)

[Połączenie standardowe (Starsze wersje MS Excel) 36](#_Toc536645119)

[Połączenie za pomocą MS Query 39](#_Toc536645120)

[CTE – Common Table Expression 45](#_Toc536645121)

[Podzapytania 46](#_Toc536645122)

[Widoki 46](#_Toc536645123)

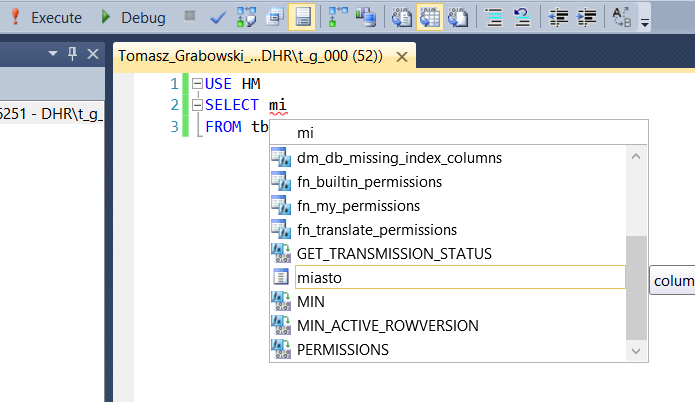
# Podział języka SQL (Structured Query Language)

Język SQL jest powszechnie stosowany i szeroko wykorzystywany w wielu rozwiązaniach bazodanowych. Można wręcz powiedzieć, że każdy analityk musi przynajmniej do pewno stopnia poznać ten język, aby sprawnie poruszać się świecie analizy danych. Podobnie jak inne języki zapytań   
i języki programowani posiada on ustandaryzowaną składnię. Należy jednak pamiętać, że istnieje wiele narzędzi i wiele mutacji i odmian języka SQL dla których mogą występować elementy charakterystyczne dla danej odmiany i/lub narzędzia (w szczególności stosowanie aliasów itp.).

Do zastosowań tego języka można oczywiście zaliczyć wybieranie i przetwarzanie danych, ale również dodawanie danych czy ich modyfikacja. Ze względu na swą rolę dzielimy SQL następująco:

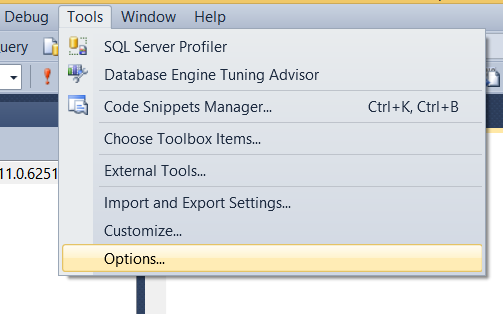
# Elementy okna oraz dostosowanie opcji Management Studio (SSMS)

Aby łatwiej korzystać z możliwości tworzenia zapytań do bazy danych możliwe jest wykorzystanie środowiska do pracy z bazami danych SQL czy SQL Server Management Studio. Jest to graficzny interfejs umożliwiający wywoływanie kwerend, śledzenie wyników i optymalizowanie wykonania kwerendy. Do najważniejszych zalet SSMS należy technologia IntelliSense, która pozwala na sprawniejsze wpisanie poleceń oraz zmniejsza znacząco liczbę pomyłek i literówek.

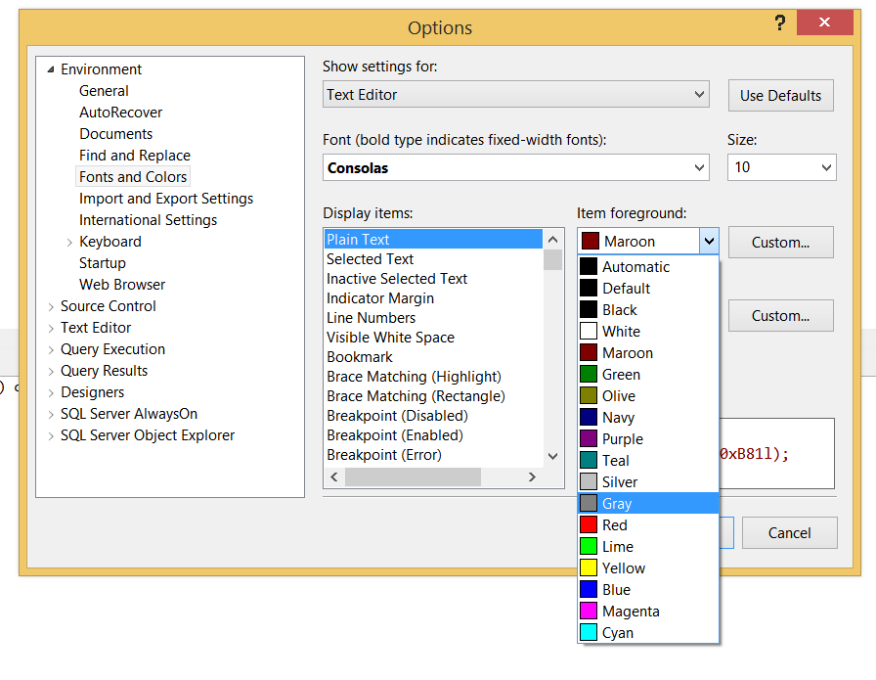


Technologia ta wywoływana jest samoczynnie automatycznie podczas wpisywania kwerendy lub ręcznie za pomocą skrótu **CTRL + spacja**

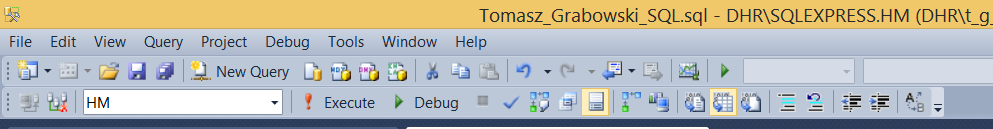
Ustawienia takie jak kolory kluczowych elementów itp. można zmienić za pomocą polecenia z menu Tools/Options:



A następnie Environment/Fonts and Colors



Aby utworzyć nową kwerendę należy użyć polecenia New Query

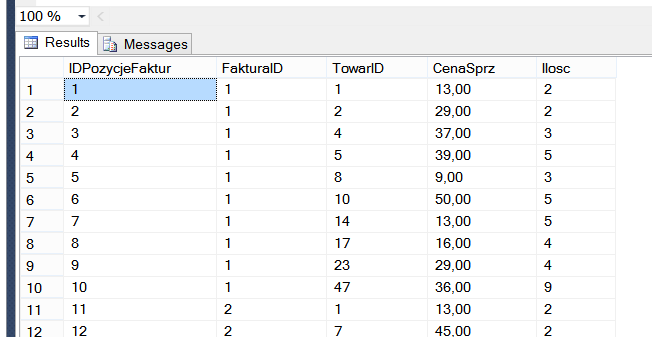


Aby zmienić sposób wyświetlania wyników należy użyć jednego z przycisków na pasku SQL Editor:

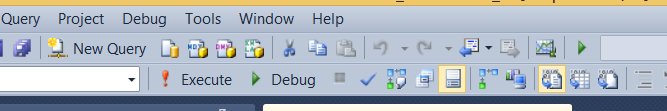
Results to GRID



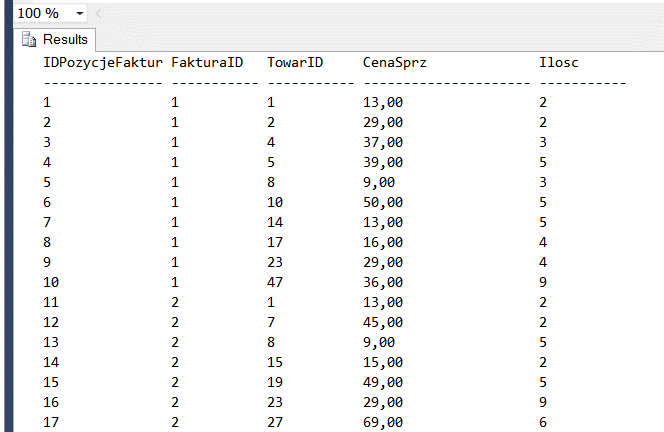
Następujący wygląd:



lub Results to TEXT

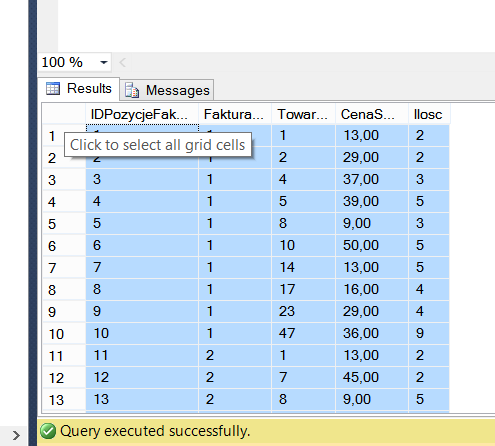


Następujący wygląd:

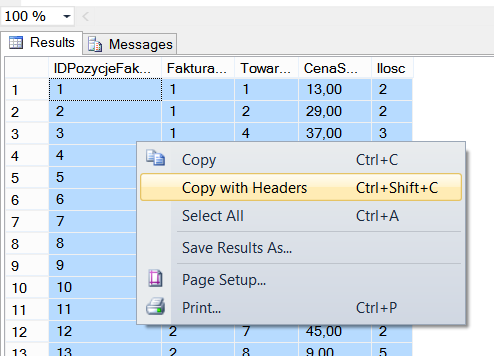


Aby łatwo skopiować wyniki kwerendy np. do MS Excel, należy w widoku Results do GRID

Zaznaczyć wszystkie kolumny np. klikając w lewym górnym rogu:

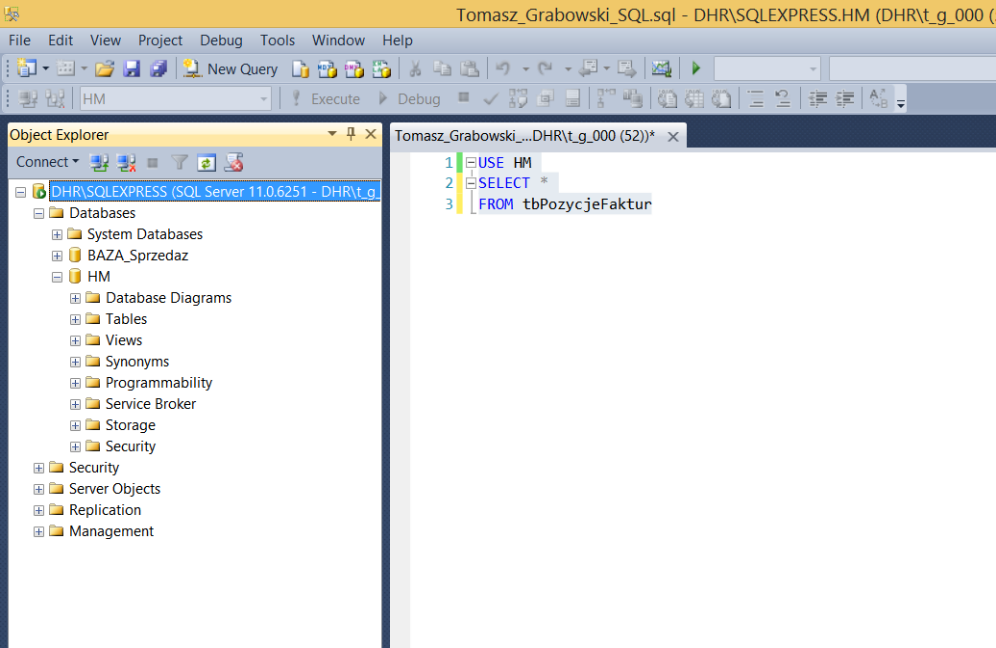


A następnie na zaznaczonym obszarze prawym przyciskiem myszy Copy with Headers:

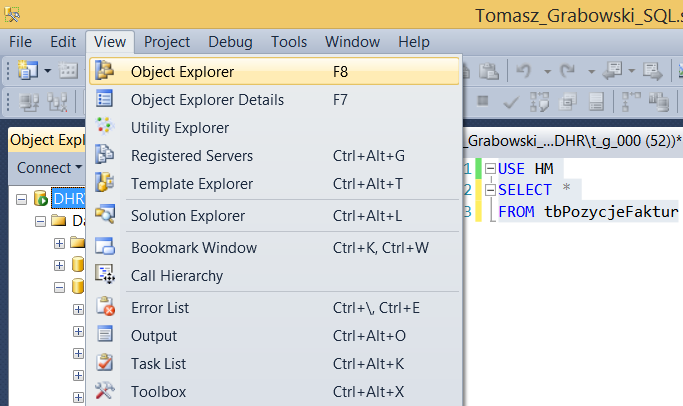


Taką zawartość można wkleić np. do MS Excel

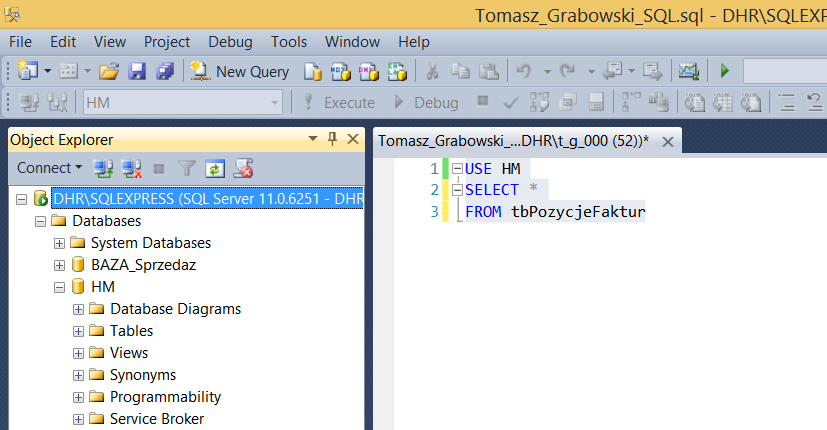
Do najważniejszych elementów okna Management Studio (SSMS) należy Object Explorer



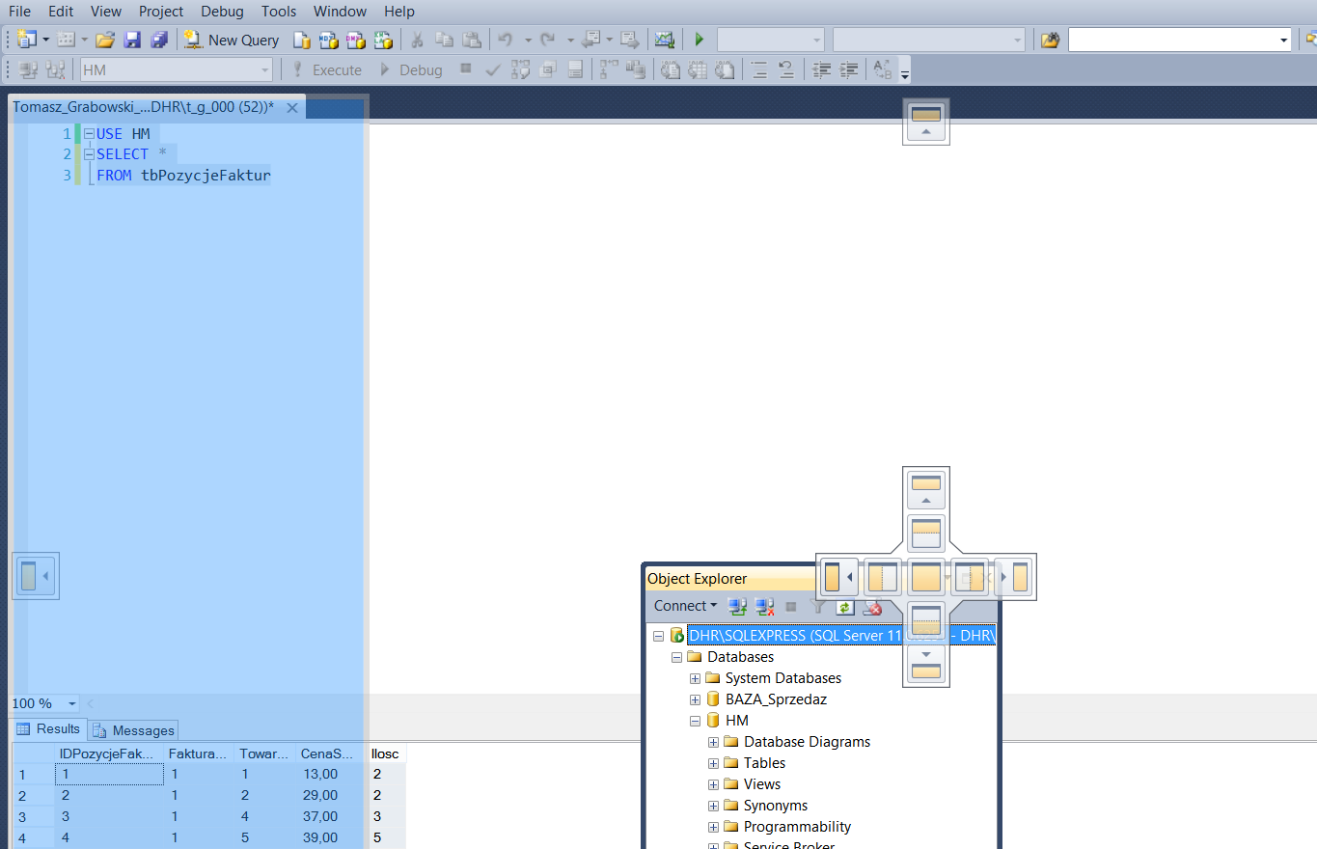
Aby go włączyć można użyć polecenia z menu VIEW/Object Explorer lub F8:



Aby określić lub zmienić lokalizację dowolnego okna należy złapać za jego górną belkę:

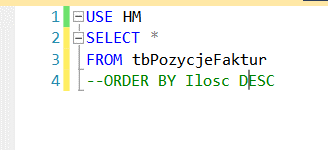


A następnie wybrać jeden dostępnych punktów poprzez najechanie kursorem:



Komentowanie fragmentów kwerendy

Jeżeli tymczasowo fragment kwerendy jest niepotrzebny, lub na potrzeby testów ma być chwilowo pomijany, można zastosować przed tym fragmentem zapis składający się z dwóch znaków   
– -- (minus minus)



Jeżeli wymagana jest zamiana na komentarz większej liczby linii kwerendy, możliwe jest zaznaczenie wielu linii i użycie przycisków do dodawania i usuwania znaku komentarza:



Aby uruchomić (wykonać kwerendę lub polecenie) należy użyć przycisku EXECUTE:



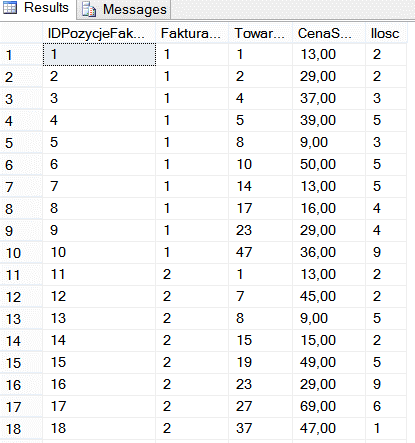
Możliwe jest również zastosowanie skrótu F5

# Zagadnienia i pojęcia związane z językiem SQL oraz tworzeniem kwerend

Podczas pracy z SQL na SQL Server warto poznać kilka pojęć i określeń, aby w pełni   
i świadomy sposób korzystać nie tylko z samego języka, ale i literatury związanej z tą tematyką.

### Tabele, pola, rekordy

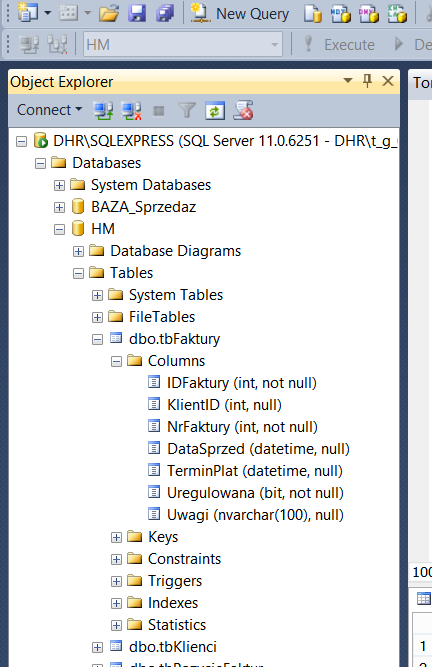
Podstawnym źródłem danych do kwerend są tabele, są to zbiory wartości ułożone   
w kolumny i wiersze. Pracując z bazami danych kolumny zwyczajowo określa się jako „pola” a wiersze jako „rekordy”. Tak więc wiele kolumn (pól) tworzy tabele, w której dane są zapisane w kolejnych wierszach (rekordach).



POLE

REKORD

Tabele oraz jej listę pól można odszukać w oknie Object Explorer:

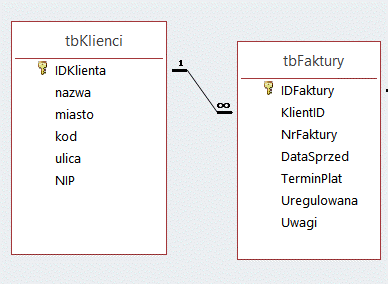


Jest to ważne miejsce, które pozwala na sprawdzenie typów danych wskazanych dla pola oraz jego rozmiaru, jest to szczególnie przydatne przy kwerendach zmieniających dane oraz przy dodawaniu nowych danych

### Klucze, relacje

Jako klucz określamy właściwość pola wskazującą i pilnującą, aby było ono unikatowe, obowiązkowo wypełnione i jeżeli to możliwe proste i szybkie. Rolą pola (lub zestawu pól) określonej jako klucz główny jest pilnowanie, aby istniała możliwość jednoznacznego odszukania danych, dla przykładu, gdy prowadzimy spis osób ciężko jest na temat każdej osoby odszukać wyróżniającą ją informację, nie będzie to ani imię, ani nazwisko ani nawet ich kombinacja, w takich przypadkach, gdy niemożliwe jest wyznaczenie takiej unikatowej informacji w tabeli tworzone jest dodatkowe pole najczęściej nazwane lub zawierające w nazwie „ID”. Taki klucz nazywamy kluczem głównym lub podstawowym.

W bazach danych istnieje również drugi rodzaj klucza – klucz obcy. Jego rolą jest przechowywanie w tabeli możliwie prostej informacji będącej odnośnikiem do konkretnego wiersza w innej tabeli. Dla przykładu tabela faktur nie zawiera wszystkich informacji o kliencie a jedynie numer (klucz obcy) prowadzący do innej tabeli i wskazujący na konkretny numer (klucz główny).



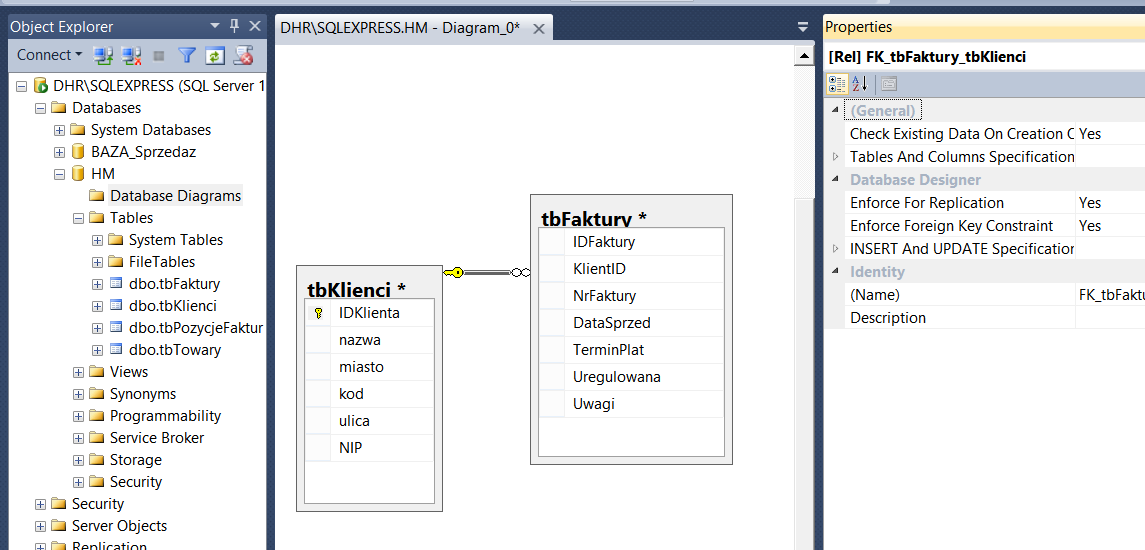
Klucz główny

Klucz obcy

Dzięki takiej konstrukcji w tabeli faktur, która może potencjalnie zawierać wiele wierszy (rekordów) zapisana jest tylko możliwe mała objętościowo informacja, która jest odnośnikiem do bardziej szczegółowej informacji, największą zaletą takiego rozwiązania jest oszczędność miejsca w bazie danych oraz mniejsza ilość błędnie wprowadzonych danych.

### Więzy integralności

Jest to temat ściśle związany z kluczami i polega na możliwości ustawienia bazy danych w taki sposób, aby pomagała w pilnowaniu spójności przechowywanych danych. Polega ono między innymi na tym, że jako wartość klucza obcego nie można używać danych nie mających swojego odpowiednika po stronie klucza głównego (np. nie można na fakturze podać numeru klienta, jeżeli klient o takim numerze nie został najpierw dodany w tabeli klientów). Drugą główną zaletą więzów integralności jest pilnowanie, aby nie było możliwości skasowania rekordu z kluczem głównym użytych chociaż raz po stronie tabeli będącej w relacji po stronie klucza obcego (np. brak możliwości skasowania klienta, który ma przynajmniej jedną fakturę).



### Procedura składowana

W swej najprostszej formie procedura składowana przypomina zwykłą kwerendę   
z możliwością uruchamiania jej z różnymi parametrami (np. możliwość wywołania kwerendy   
z podsumowaniem sprzedaży produktów za każdym razem za inny rok podany na początku jako zmienna).

Podczas tworzenia procedury składowanej można określić ilość i tym danych dla parametrów.

Oto przykładowa składnia tworzenia procedury składowanej:

CREATE PROCEDURE p\_Wybrana\_Data (@Data AS DATE) AS

SELECT \*

FROM tbFaktury

WHERE DataSprzed = @Data

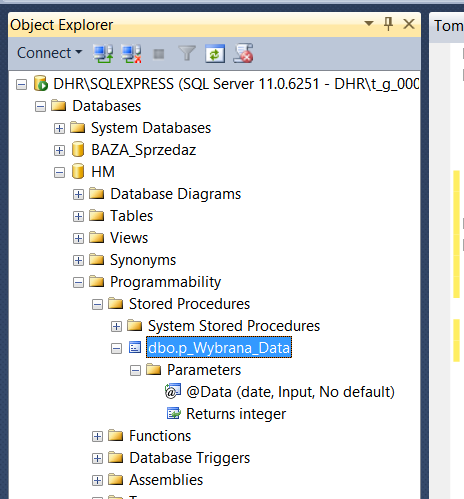
Procedura ta pozwala na wybranie faktur z konkretnego dnia podanego jako parametr

Jak wywołać procedurę

EXEC p\_Wybrana\_Data '2018-09-20'

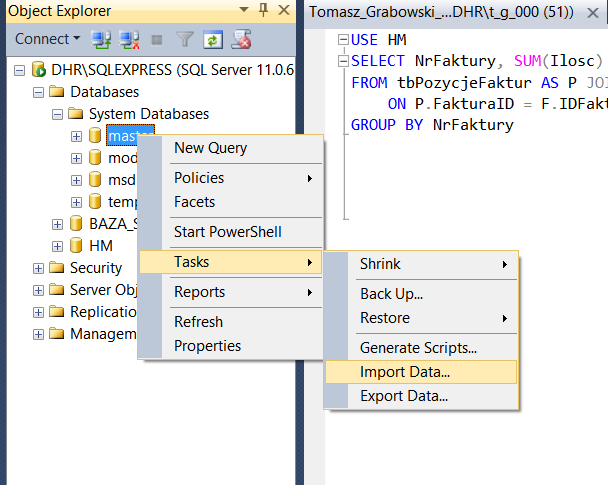
Procedura może być wywołana bez słowa EXEC, natomiast dzięki niemu aktywuje się technologia IntelliSense dzięki której łatwiej podać prawidłową nazwę procedury.

Gdzie odszukać już utworzone procedury:

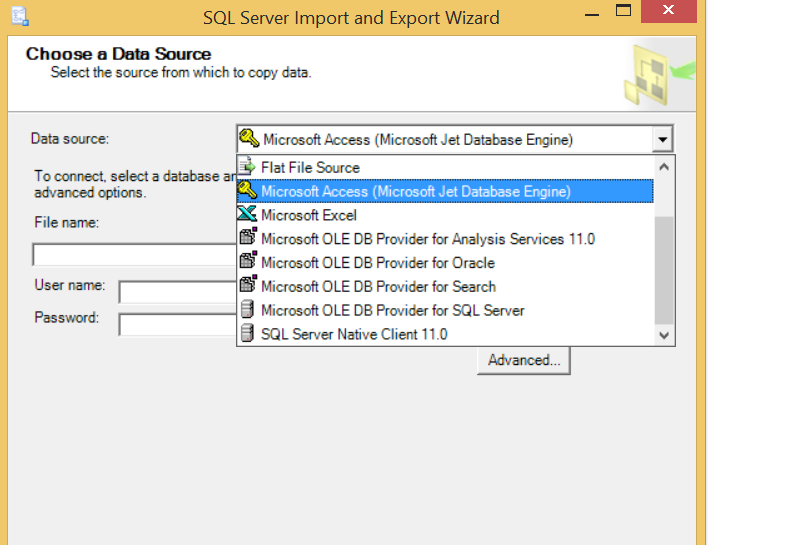


Importowanie danych  
Aby rozpocząć pracę potrzebne będzie źródło danych, jeżeli mamy już bazę np. w MS Access z łatwością można ją przenieść na Server SQL. Oto jakie kroki należy wykonać:

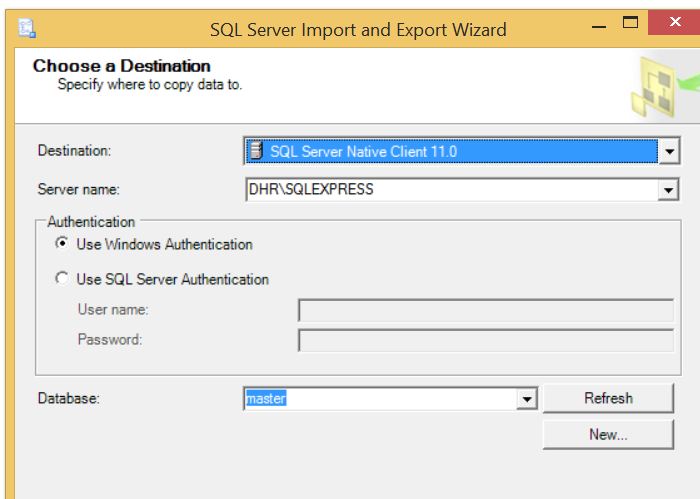
W oknie Object Explorer należy kliknąć prawym przyciskiem myszy na dowolnej bazie danych a następnie wybrać Tasks/Import Data…



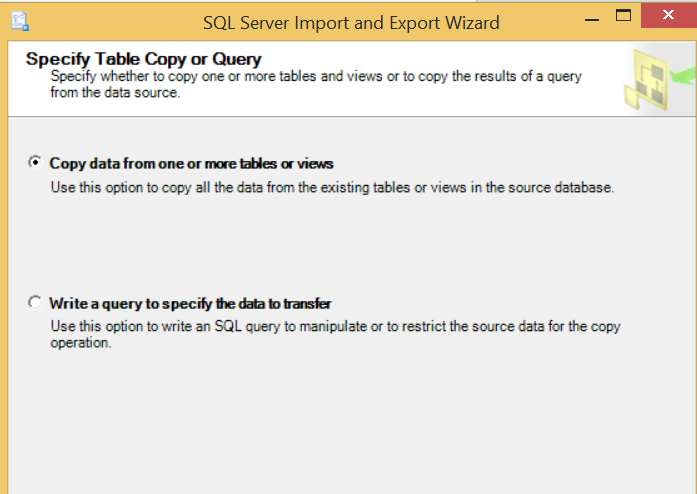
W kolejnych krokach należy wybrać z jakiego źródła ma nastąpić import i za pomocą przycisku Browse wybrać plik źródłowy:



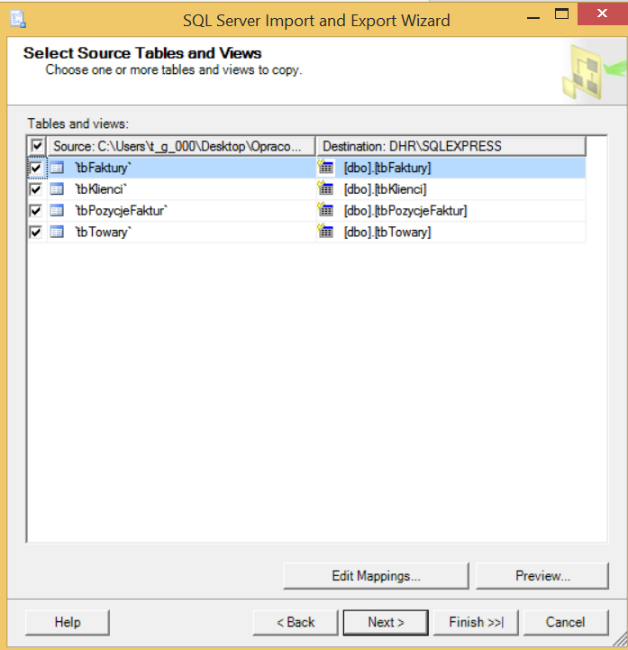
Kolejny krok to wybór serwera oraz bazy danych do której zaimportowane zostaną dane:



Wybór metody wskazania danych źródłowych to kwestia preferencji, lecz w wielu wypadkach wygodniej będzie wskazać tabele po prostu je wyklikując



Pozostaje już tylko wskazać potrzebne tabele i zakończyć kreator



# Podstawowe elementy składni języka SQL

Wykorzystanie możliwości języka SQL wymaga poznania kilku kluczowych słów, dzięki którym można budować skomplikowane wyrażenia.

### Wybieranie danych: SELECT

W tej sekcji kwerendy wskazujemy jakie kolumny (pola) mają być wyświetlone w wyniku kwerendy. Możliwe jest tu również używanie funkcji czy konstrukcji warunkowych. W tej części kwerendy możliwe jest również nadawanie aliasów czy zmiana nazwy istniejącej lub nadanie nazwy dla wyniku wyrażenia lub funkcji. Oto możliwe warianty nadawanie aliasów:

Nazwa Alias

Nazwa AS Alias

Nazwa ‘Alias wiele wyrazów’

Nazwa AS ‘Alias wiele wyrazów’

Nazwa [Alias wiele wyrazów]

Nazwa AS [Alias wiele wyrazów]

Przy nadawaniu nazw składających się z więcej niż 1 wyrazu należy zastosować nawiasy kwadratowe lub apostrof

Przykład:

SELECT imie, nazwisko, LTRIM(Kraj) AS Poprawiony\_Kraj

### Źródła danych: FROM

Jest to miejsce, w którym wskazać należy dane źródłowe używane w kwerendzie, budowę kwerendy zwykle zaczyna się od tej sekcji a następnie przechodzi się do budowania sekcji from dzięki zachowaniu takiej kolejności sekcja SELECT będzie „widziała” i podpowiadała kolumny ze wszystkich tabel ze źródła danych.

FROM tbPozycjeFaktur

FROM tbPozycjeFaktur JOIN tbFaktury

ON tbPozycjeFaktur.FakturaID = tbFaktury.IDFaktury

Możliwe jest tu podobnie jak w sekcji SELECT tworzenie Aliasów.

FROM tbPozycjeFaktur AS P JOIN tbFaktury AS F

ON P.FakturaID = F.IDFaktury

### Sortowanie danych: ORDER BY

Tu określana jest kolejność w jakiej wyświetlane są dane wynikowe

ORDER BY NrFaktury ASC - Sortowanie rosnące lub zgodne z kolejnością alfabetyczną

ASC może być pominięte ponieważ domyślnie właśnie ono jest stosowane

ORDER BY NrFaktury ASC jest jednoznaczne z zapisem ORDER BY NrFaktury

ORDER BY NrFaktury DESC - Sortowanie malejące lub odwrotne do kolejności alfabetycznej

W sekcji sortowania można wskazać również wiele kolumn. W takim wypadku sortowanie odbywa się w kolejności od lewej do prawej strony, czyli w każda kolejna kolumna jest sortowana wewnątrz powtarzających się wartości z poprzedniej kolumny.

### Grupowanie danych: GROUP BY

Ta część niezbędna jest, gdy w kwerendzie wykonywane są obliczenia. Umieszczone powinny w niej być wszystkie pola wymienione w sekcji SELECT a nie poddawane żadnej agregacji.

SELECT NrFaktury, SUM(Ilosc) AS Suma\_Ilosci

FROM tbPozycjeFaktur AS P JOIN tbFaktury AS F

ON P.FakturaID = F.IDFaktury

GROUP BY NrFaktury

UWAGA: SQL Server nie zezwala na umieszczanie w tej sekcji aliasów, więc konieczne jest umieszczenie całego wyrażenia np. wykonywanej funkcji itp..

### Filtrowanie danych: WHERE, HAVING

Umiejscowienie tych dwóch typów filtrowania w strukturze kwerendy może podpowiadać ich rolę.

SELECT

FROM

WHERE

GROUP BY

HAVING

WHERE jest wykonywany w początkowej fazie wykonania kwerendy i tym samym pozwala na organicznie liczby wierszy (rekordów) które do niej trafiają (np. tylko dane za określony dział), a HAVING jest filtrem, który stosowany jest jedynie dla kolumn, dla których wykonane było wcześniej obliczenie (np. tylko dane, gdzie suma przekracza jakąś wartość).

Operatory:

= - równa się

<> - różny od

=! - różny od

> - większy niż

< - mniejszy niż

>= - większy lub równy

<= - mniejszy lub równy

BETWEEN x AND y - Liczby w przedziale od x do y (włącznie z wartościami skrajnymi)

LIKE ‘wyrażenie‘

Jako część lub całość wyrażenia można stosować znaki wieloznaczne:

% - dowolna liczba dowolnych znaków

(np. Nazwisko LIKE ‘%a’ – nazwiska kończące się na literę „a”

\_ - jeden dowolny znak

(np. Nazwisko LIKE ‘Kowalsk\_’ – nazwisko z jedną nieznaną literą obowiązkowo występującą po „Kowalsk”)

[] - Pozwalają na wypisanie dopuszczalnych wartości

(np. Nazwisko LIKE ‘[kng]owalska’ – nazwisko Kowalska lub Nowalska lub Gowalska)

^ - wykluczenie

(np. Nazwisko LIKE ‘[^k]owalska’ – pierwszą literą może być każda inna niż „k”

### Operatory logiczne

Do najczęściej stosowanych operatorów logicznych należą OR i AND.

**AND** stosujemy, gdy wymagane jest sprawdzenie czy minimum dwa warunki są spełnione jednocześnie

WHERE IDFaktury >1 AND IDFaktury <10 (wybrane zostaną faktury z numerami od 2 do 9)

**OR** stosujemy, gdy chcemy sprawdzić czy przynajmniej jeden z warunków został spełniony

WHERE IDFaktury = 1 OR IDFaktury = 10 (wybrane zostaną faktury z numerem 1 i 10)

### Zapytania w oparciu o różne bazy danych

W niektórych przypadkach konieczne będzie odwołanie się do danych przechowywanych   
w bazie danych innej niż wskazana jako bieżąca, aby to zrobić w sekcji FROM należy zastosować odpowiednie odwołanie

Nazwa\_Bazy\_Danych.dbo.Nazwa\_Tabeli

(np. FROM HM.dbo.tbTowary)

### Łączenie tabel: JOIN, LEFT JOIN, RIGHT JOIN, OUTER JOIN, UNION, UNION ALL

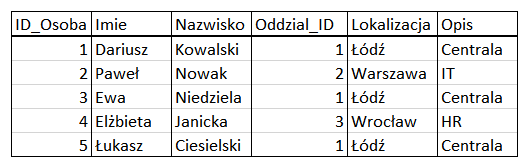
Nieodłącznym elementem bardziej skomplikowanych kwerend jest konieczność stosowania złączeń źródeł danych (najczęściej tabel, ale mogą to być również podzapytania lub wyrażenia common table expression)

Najpopularniejszym i najczęściej stosowanych jest JOIN (INNER JOIN) czyli połączenie, które jako wynik pokaże tylko dane, które znajdują się po obu stronach relacji

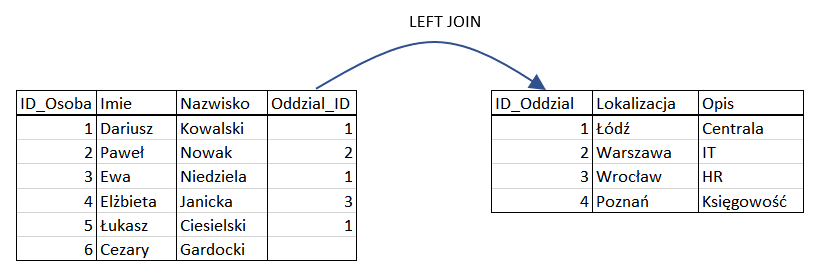
Obraz zawierający tekst

Opis wygenerowany przy bardzo wysokim poziomie pewności

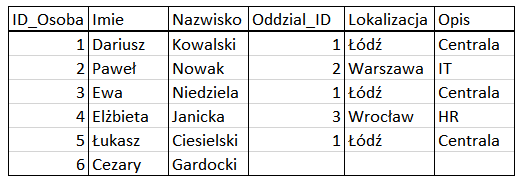
Wynikiem kwerendy wykonanej na takich tabelach nie będą wszystkie dane z obu tabel a jedynie dane występujące to jednej i po drugiej stronie sprzężenia



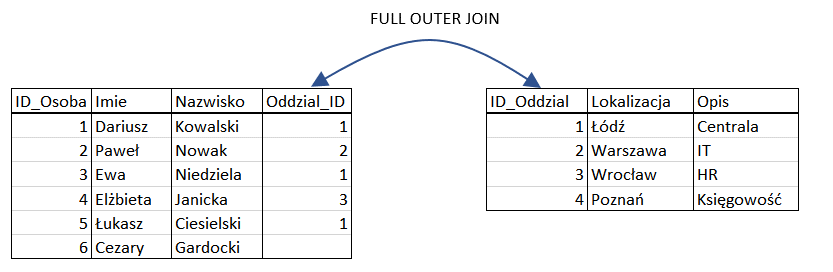
Aby wybrać wszystkie dane z jednej z tabel niezależnie od tego czy w drugiej przyłączonej tabeli istnieją odpowiadające dane należy zastosować LEFT JOIN lub RIGHT JOIN. Różnica pomiędzy nimi wyniki tylko z ułożenia tabel.



Wynikiem tego zapytania będą wszystkie dane z tabeli lewej nawet jeżeli nie mają swojego odpowiednika po stronie prawej.



Kolejnym typem sprzężenia, które jest często wykorzystywane jest FULL OUTER JOIN. Polega ono na wybraniu wszystkich danych wspólnych oraz danych występujących tylko w lewej i tylko w prawej tabeli.

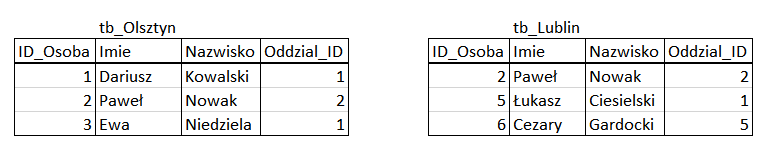


Wynikiem będą dane z lewej i prawej strony oraz część wspólna.

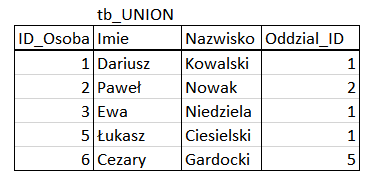


### UNION

Jeżeli niezbędne jest wykonanie operacji na więcej niż jednej tabeli lub wyniku kwerendy jednocześnie, można zastosować operację UNION, która powoduje połączenie kilku źródeł danych. Należy jednak pamiętać, aby łączone źródła miały tą samą liczbę kolumn oraz należy sprawdzić zgodność typów danych dla odpowiadających sobie kolumn.

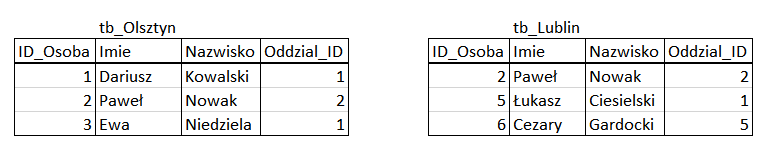


Wynikiem takiego zapytania będą dane pochodzące z obu tabel, przy jednoczesnym pomijaniu danych, które się powtarzają.

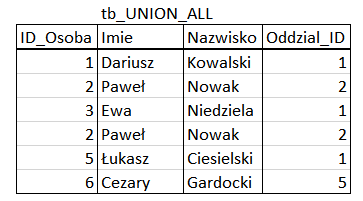


### UNION ALL

Różnica pomiędzy tym poleceniem a UNION polega wyłącznie na fakcie, że wynikowa tabela będzie zawierać wszystkie dane nawet jeżeli występują więcej niż raz.



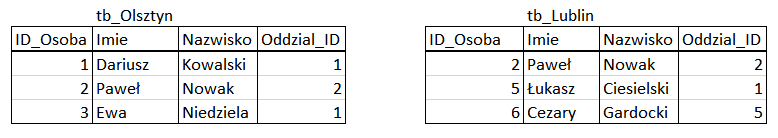
Wynik UNION ALL



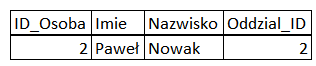
### INTERSECT

Jeżeli potrzebne jest odszukanie części wspólnej pomiędzy tabelami jedną z metod jest użycie polecenia INTERSECT. Jeżeli zostanie ono zastosowane na źródłach danych, w których występuje więcej niż jedna kolumna jednakość stwierdzana jest na podstawie dopasowania wartości we wszystkich kolumnach.

Początkowe dwa zestawy danych:

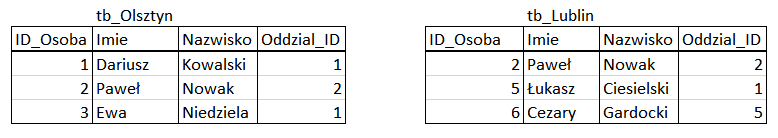


Wynikiem będą tylko dane znajdujące się we wszystkich tabelach źródłowych.

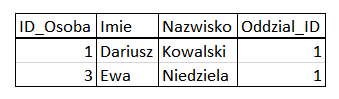


### EXCEPT

Jeżeli z jednego źródła danych chcemy wykluczyć wyniki pochodzące z innego źródła danych można zastosować polecenie EXCEPT, które po porównaniu danych z wszystkich kolumn wykluczy   
z pierwszego źródła danych rekordy występujące w drugim.



Wynik z zastosowaniem EXCEPT



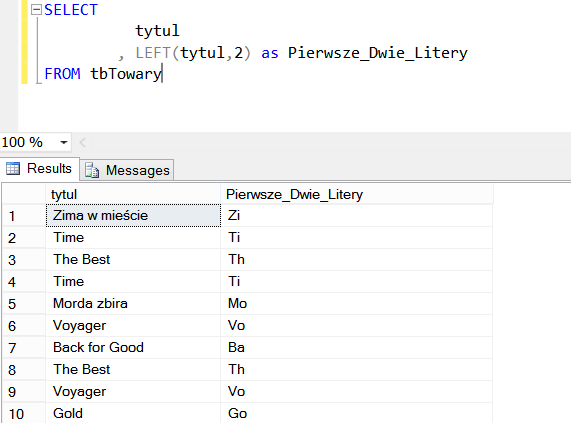
# Funkcje przetwarzania tekstu

Dane przechowywane w tabelach nie zawsze zapisywane są w sposób idealny i wymagają często poprawy i zmian. Z pomocą przychodzą tu funkcje tekstowe pozwalające na wycinanie niepotrzebnych spacji czy wycinanie tylko potrzebnego fragmentu.

LEFT

Funkcja pozwalająca na wycięcie określonej liczby znaków z lewej strony tekstu

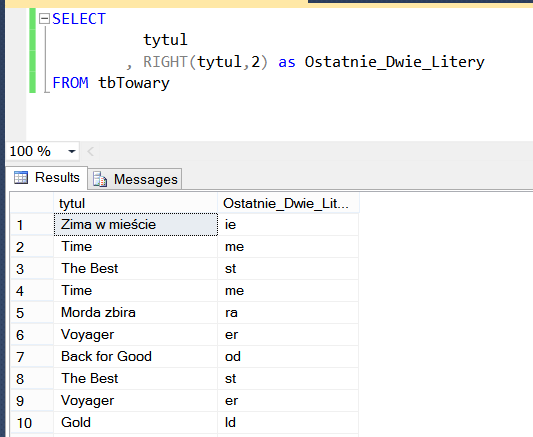
LEFT(Nazwa\_Pola lub Tekst, Liczba\_Znaków\_Do\_Wycięcia)



RIGHT

Funkcja pozwalająca na wycięcie określonej liczby znaków z prawej strony tekstu

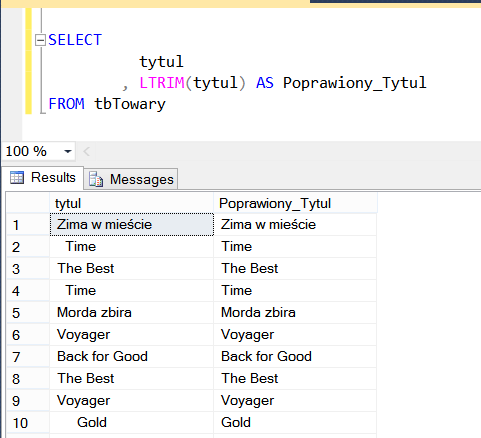
RIGHT(Nazwa\_Pola lub Tekst, Liczba\_Znaków\_Do\_Wycięcia)



LTRIM (RTRIM)

Dzięki tej funkcji można usunąć zbędne spacje pojawiające się na początku tekstu (LTRIM) lub na końcu tekstu (RTRIM)

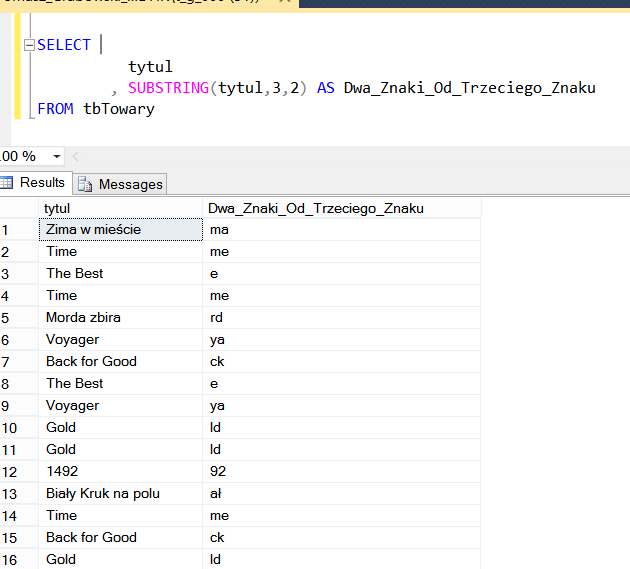
LTRIM(Kolumna\_Z\_Tekstem)



SUBSTRING

Ta funkcja powinna być zastosowana, jeżeli potrzebne jest wycięcie fragmentu tekstu znajdującego się w środki ciągu tekstowego.

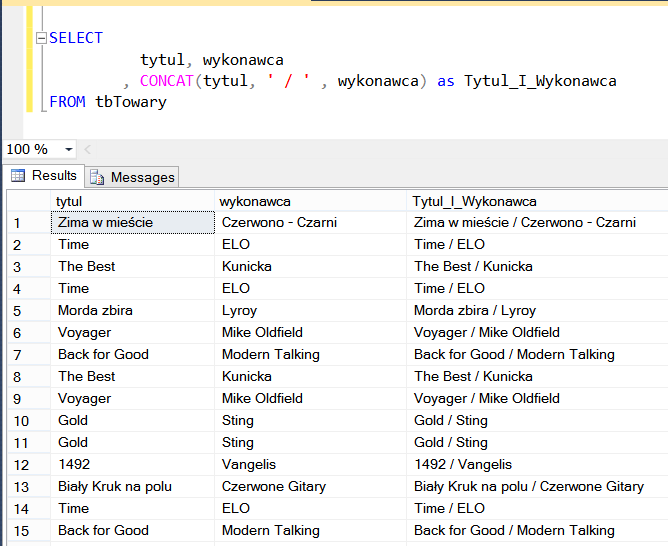
SUBSTRING(Tekst, Od\_Ktorego\_Znaku\_Wycinac, Ile\_Znakow\_Wycinac)



CONCAT ( + )

Do łączenia tekstów można stosować zarówno funkcję CONCAT jak i operator + , należy jednak pamiętać, o tym, że łączone fragmenty muszą być tekstami.

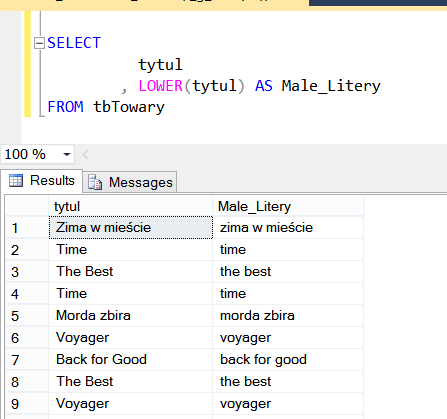
CONCAT(Tekst1, Tekst2, Tekst3, …)



LOWER

Funkcja ta dokonuje wielkości liter na małe.

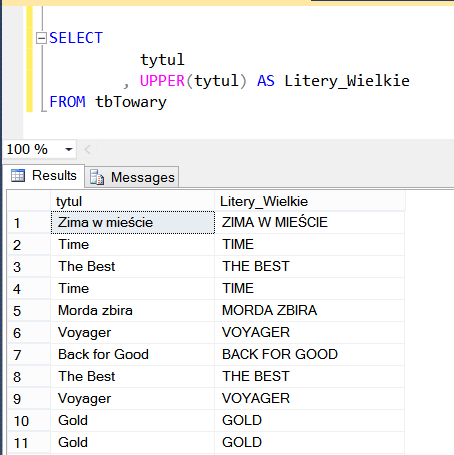
LOWER(Tekst)



UPPER

Funkcja ta dokonuje wielkości liter na duże.

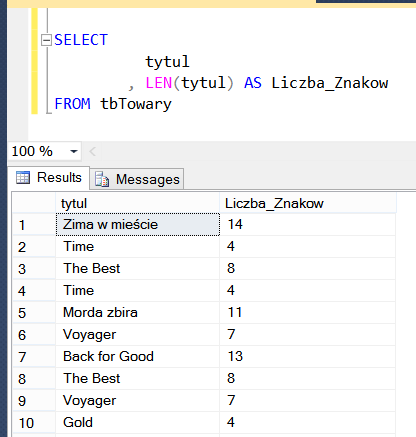
UPPER(Tekst)



LEN

Dzięki tej funkcji można sprawdzić łączną liczbę znaków w polu tekstowym (uwaga każdy znak np. spacja czy kropka jest wliczany)

LEN(Tekst)



# Operacje na datach

GETDATE

Funkcja, która jako wynik wyświetli aktualną datę i czas.

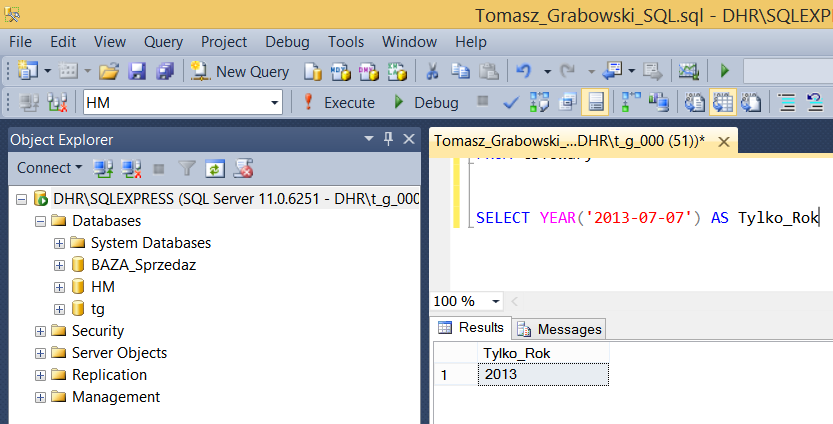
GETDATE()



YEAR

Funkcja wybierające rok z daty.

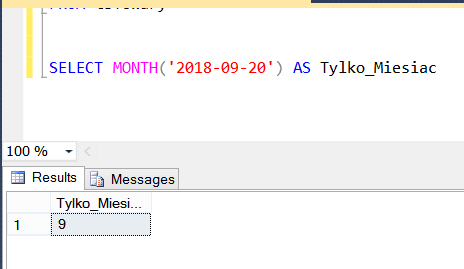
YEAR(Data)



MONTH

MONTH(Data)

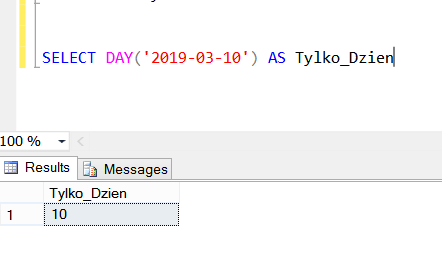
Funkcja wybierająca miesiąc z daty



DAY

DAY(Data)

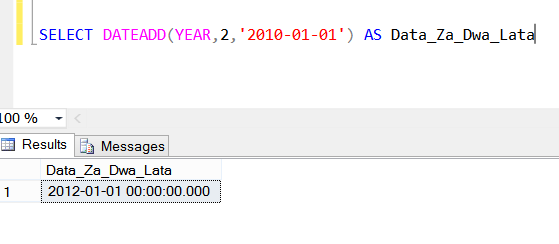
Funkcja wybierająca dzień z daty



DATEADD

DATEADD(Interwał, Liczba\_Interwalow, Data)

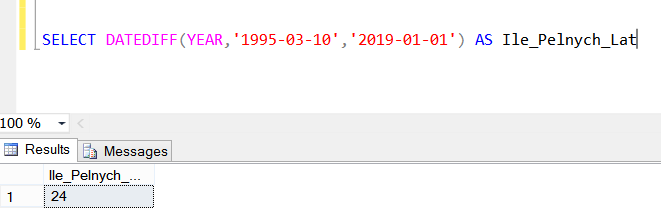
Dzięki tej funkcji można przesunąć datę o wskazaną liczbę interwałów. Jako interwał można stosować różne jednostki takie jak: rok, miesiąc, dzień, godzina, itp.



DATEDIFF

Umożliwia wyliczenie różnicy pomiędzy datami i pokazanie wyniku w pełnych skończonych interwałach.

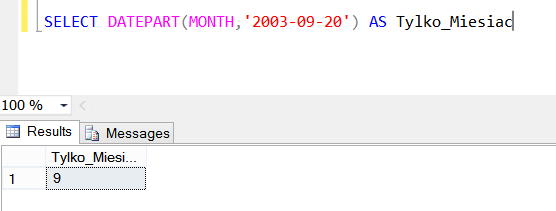
DATEDIFF(Interwal, Data\_Poczatkowa, Data\_Koncowa)



DATEPART

Bardzo często stosowana funkcja pozwalająca na wybranie z daty tylko jej fragmentu we wskazanej jednostce.

DATEPART(Interwal, Data)



Możliwe do wykorzystania interwały:

| **Nazwa** | **Opis** |
| --- | --- |
| year, yyyy, yy | Rok |
| quarter, qq, q | Kwartał |
| month, mm, m | Miesiąc |
| dayofyear | Dzień roku |
| day, dy, y | Dzień |
| week, ww, wk | Tydzień wg 1 stycznia |
| iso\_week | Tydzień wg pierwszego pełnego tygodnia |
| weekday, dw, w | Dzień tygodnia |
| hour, hh | Godzina |
| minute, mi, n | Minuta |
| second, ss, s | Sekunda |
| millisecond, ms | Milisekunda |

# Konstrukcje warunkowe

Gdy w zależności od warunku chcemy zwrócić jako wynik inne wartości z pomocą przyjdą konstrukcje warunkowe. Oto dwie, które są bardzo przejrzyste a jednocześnie bardzo przydatne. CASE jest konstrukcją stosowaną głównie, gdy zamierzamy przeprowadzać testy na poziomie wiersza (rekordu) i zwracać dla każdego wartość, IF stosowany jest głównie do podejmowania decyzji, który złożony blok poleceń wykonać.

### CASE

Konstrukcja CASE w łatwy i przejrzysty sposób pozwala sprawdzić warunek (lub wiele warunków) i na tej podstawie zwrócić wynik. Składnia konstrukcji CASE:

CASE

WHEN warunek THEN wynik

WHEN warunek 2 THEN wynik 2

OPCJONALNE

WHEN warunek 3 THEN wynik 3

…

ELSE wynik dla pozostałych

END

### IF

Alternatywą jest konstrukcja IF, która w podobny sposób pozwoli na stworzenie konstrukcji warunkowej. Składnia konstrukcji IF:

IF warunek

BEGIN

Co wykonać

Co wykonać

Co wykonać

END

ELSE

BEGIN

Co wykonać

Co wykonać

Co wykonać

END

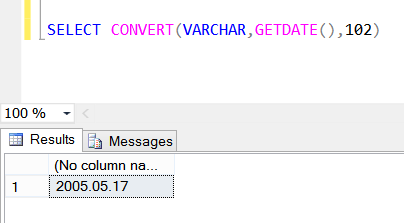
# Konwersja typów danych

CONVERT

Często stosowana funkcja pozwalająca na konwersję do innego typu danych oraz określenie wyglądu wynikowego danych.

CONVERT(Typ\_Danych, Dane\_Do\_Konwersji, Zastosowany\_Format)

Konwersja może nastąpić do jednego z tych typów: bigint, int, smallint, tinyint, bit, decimal, numeric, money, smallmoney, float, real, datetime, smalldatetime, char, varchar, text, nchar, nvarchar, ntext, binary, varbinary, image



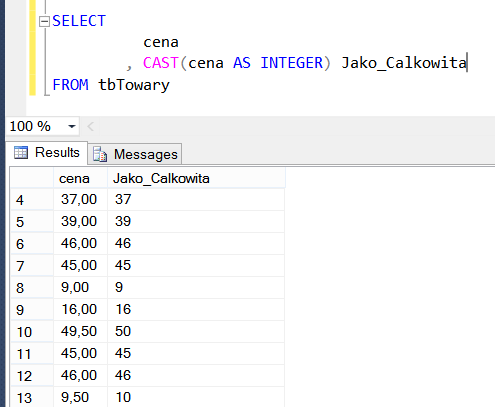
W zależności od zastosowanego ostatniego argumentu dane przyjmą inny format wyjściowy

CAST (TRY\_CAST od 2012)

Pozwala na szybkie i proste konwertowanie, bez konieczności definiowana wynikowego wyglądu. Funkcja z jednej strony jest prostsza w zastosowaniu niż CONVERT z drugiej strony daje mniejsze możliwości, jeżeli chodzi o formatowanie wyników.

Formaty, do których może nastąpić konwersja: bigint, int, smallint, tinyint, bit, decimal, numeric, money, smallmoney, float, real, datetime, smalldatetime, char, varchar, text, nchar, nvarchar, ntext, binary, varbinary, image

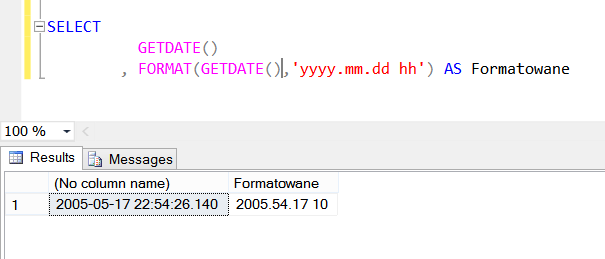
Jeżeli zastosujemy funkcję CAST na danych typu FLOAT lub NUMERIC i dokonamy konwersji do INTEGER, część dziesiętna liczby zostanie odcięta, dla pozostałych typów danych zostanie zaokrąglona.



FORMAT

Jest to funkcja dostępna dopiero od wersji 2012

FORMAT(Dane\_Do\_Formatowania, Format, Jezyk\_Region)



# Zapytania funkcyjne

### Dodawanie danych do tabel: INSERT

Jeżeli konieczne jest dostawienie nowych wierszy do tabeli z pomocą przyjdzie polecenie INSERT INTO, oto jego składnia oraz sposób na jego automatyczne generowanie:

INSERT INTO Nazwa\_Tabeli (Nazwa\_Pola\_1, Nazwa\_Pola\_2, Nazwa\_Pola\_3,…)

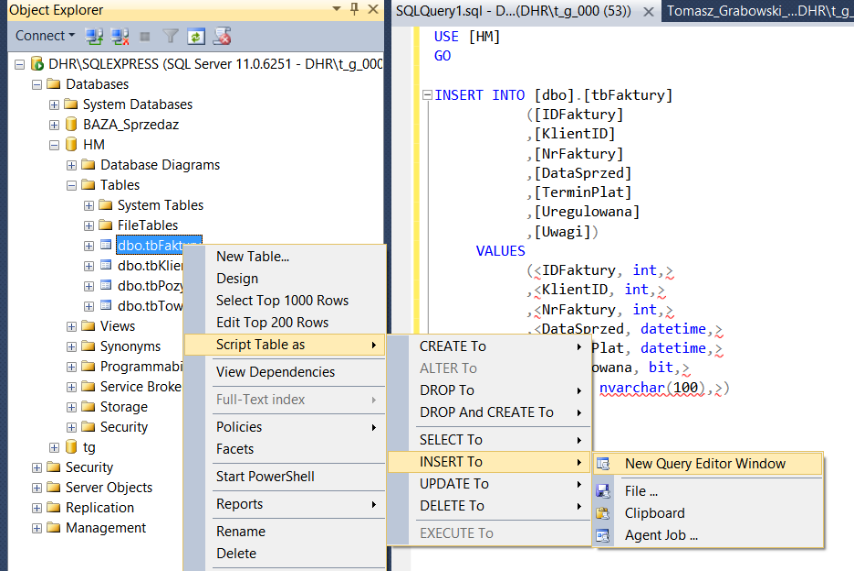
VALUES (Wartosc\_Dla\_Pola\_1, Wartosc\_Dla\_Pola\_2, Wartosc\_Dla\_Pola\_3,…)

Np.

INSERT INTO tb\_Osoba (ID\_Osoba, Imie, Nazwisko) VALUES (5,’Kalina’,’Nowak’)

Ponieważ ten zapytania nie jest stosowany tak często jak „Zwykły” SELECT, aby ułatwić sobie tworzenie jego składni można skorzystać z gotowego kreatora takich zapytań, jego dodatkową zaletą jest fakt automatycznego utworzenia listy typów pól oraz ich rozmiarów, co pozwala na uniknięcie błędów podczas dodawania nowych rekordów.

Aby go zastosować należy w oknie Object Explorer odszukać tabelę, do której mają być dodane dane   
i pod prawym przyciskiem myszy wybrać: Script table as / INSERT INTO / New Query Editor Window



### Modyfikowanie istniejących danych: UPDATE

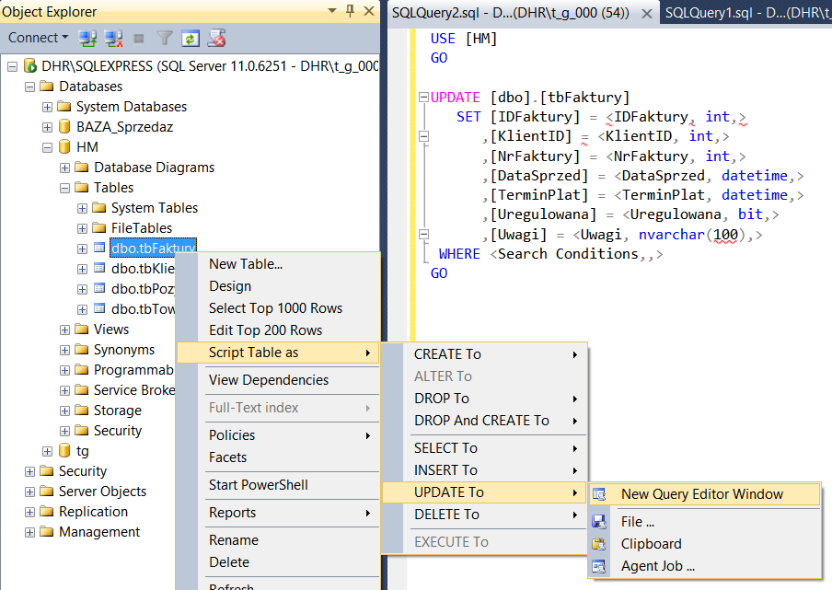
Gdy w grę wchodzi modyfikacja już istniejących danych niezbędna będzie konstrukcja UPDATE. Pozwala ona na zmianę wartości już istniejących (lub NULL) na nowe. Należy oczywiście pamiętać o wprowadzeniu danych o odpowiednim typie. Składnia polecenia UPDATE

UPDATE Nazwa\_Tabeli SET Pole\_1 = Nowa\_Wartość\_Dla\_Pola,

Pole\_2 = Nowa\_Wartość\_Dla\_Pola\_2,…

WHERE Warunek wybierający dla jakich rekordów dokonać aktualizacji.

Podobnie jak dla instrukcji INSERT można skorzystać z kreatora.

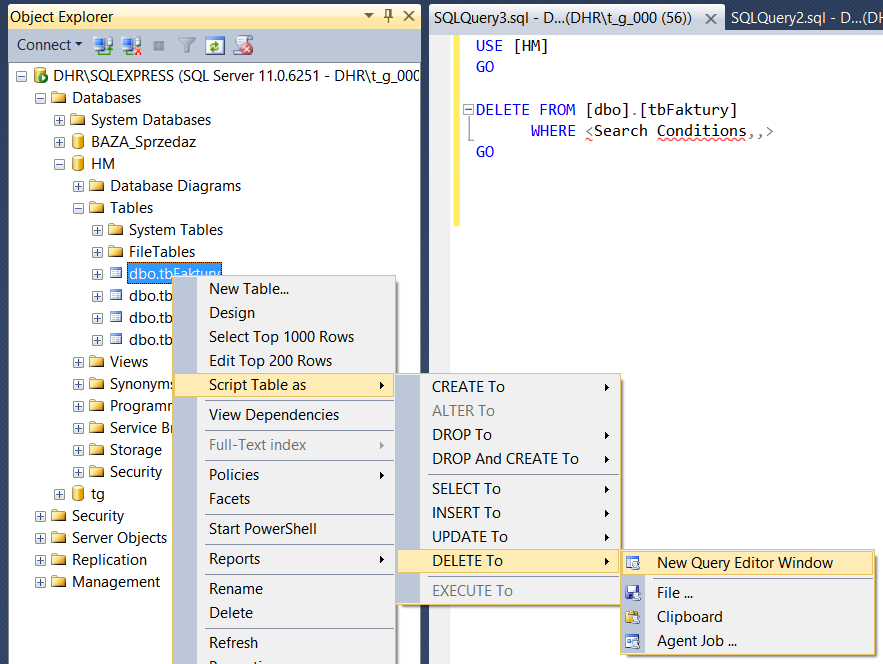


Kwerenda typu UPDATE prawie zawsze wykonywana jest z sekcją WHERE, gdyż brak jej zastosowania oznacza dokonanie aktualizacji we wszystkich wierszach. W ramach konstrukcji UPDATE możliwe jest zastosowanie konstrukcji warunkowej CASE.

### Usuwanie danych: DELETE

Sporadycznie stosowane, lecz wymagające odnotowania polecenie DELETE jest sposobem na usuwanie danych z istniejących tabel. Przy tej operacji należy jednak pamiętać, że jeżeli założone są więzy integralności, nie pozwolą one na usunięcie danych po stronie tabeli słownikowej, jeżeli rekord ten jest wykorzystany w tabeli po drugiej stronie relacji. Składnia polecenia DELETE

DELETE FROM Nazwa\_Tabeli WHERE Kryterium\_Wyboru\_Danych\_Do\_Kasowania



### Kasowanie tabel: DROP TABLE

Bardzo rzadko stosowana i wymagające specjalnych uprawnień polecenie zwykle nie jest dostępne dla każdego, tym niemniej wymagające wzmianki polecenie pozwalające na kasowanie tabeli. Składnia polecenia:

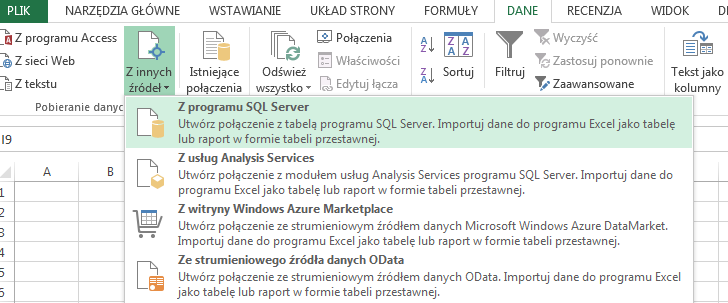
DROP TABLE Nazwa\_Tabeli

# Podłączenie MS Excel do danych w SQL Server

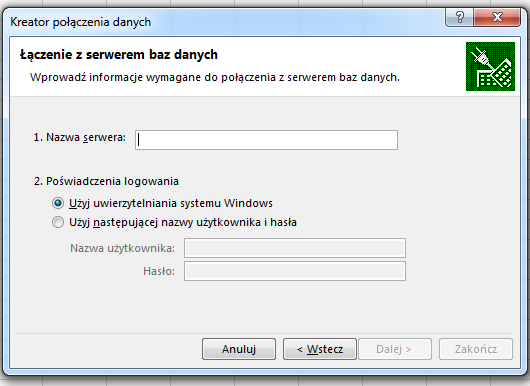
Jeżeli chcemy udostępnić dane z SQL Server dla szerszego grona osób bez konieczności instalowania i wdrażania SQL Server istnieje kilka możliwości podłączania się do danych SQL bez konieczności posiadania SQL Server.

### Połączenie standardowe (Starsze wersje MS Excel)

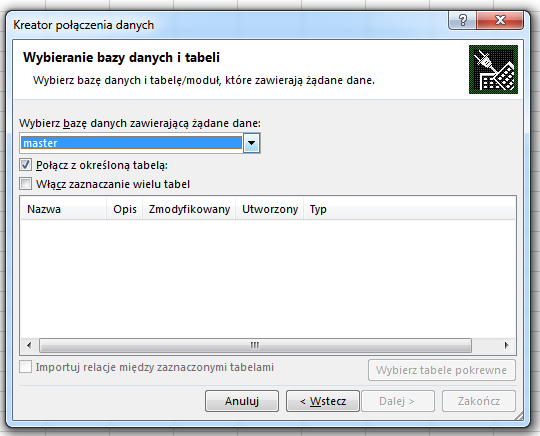
Dla użytkowników przyzwyczajonych do korzystania z MS Excel istnieje możliwość podłączania się do bazy danych MS SQL Server za pośrednictwem MS Excel. Oto jak wykonać takie połączenie:



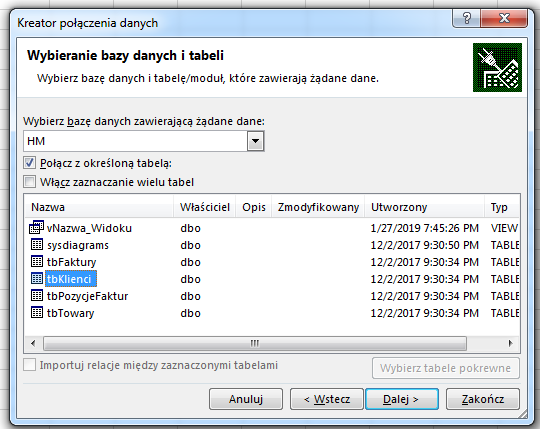
Następnie podajemy nazwę serwera i przechodzimy do kolejnego kroku:



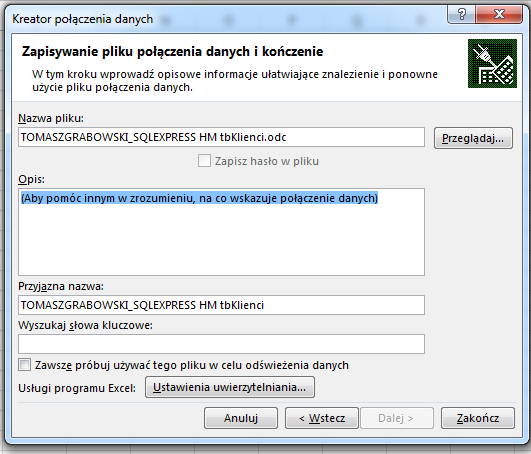
Kolejny krok to wybór bazy danych:



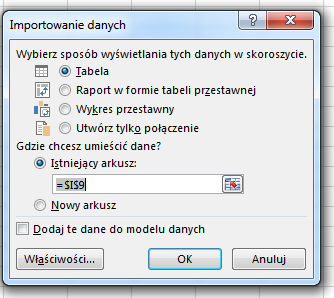
I wybór tabeli lub widoku



Pozostaje już tylko zapisanie ew. opisu i zakończenie

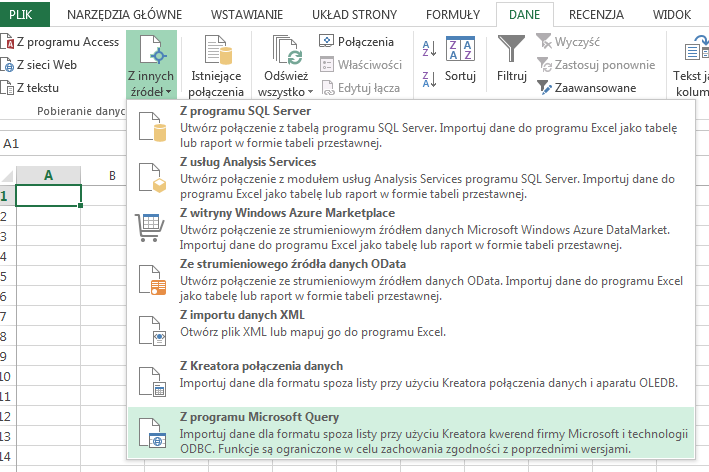


Wybieramy miejsce docelowe oraz w jakiej formie chcemy wyświetlić źródło danych z SQL Server

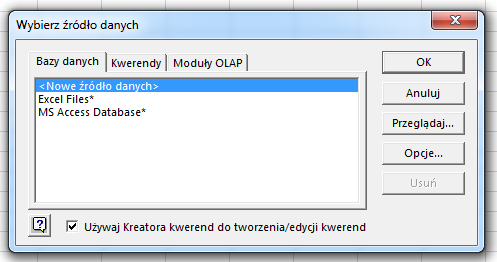


### Połączenie za pomocą MS Query

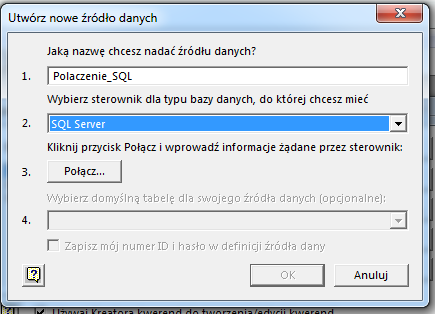
MS Query to bardzo stare, lecz ciągle spełniające swoją funkcję narzędzie pozwala na utworzenie połączenia do SQL Server wraz z możliwością pobierania parametru lub zaczytywania go   
z komórki MS EXCEL. Oto jak je wykorzystać:



Jeżeli połączenie wykonywane jest po raz pierwszy pierwszym krokiem będzie stworzenie nowego połączenia:

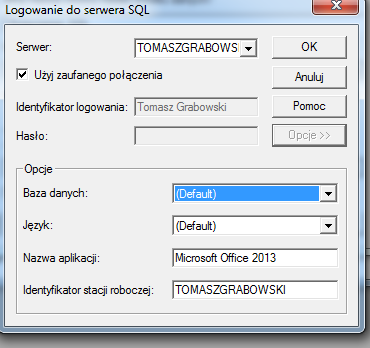


Tworzymy nazwę pod jaką będzie widoczne połączenie oraz wybieramy SQL jako sterownik

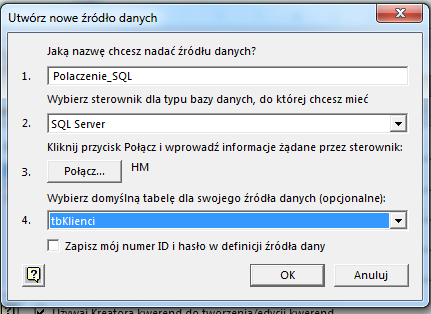


Po wybraniu przycisku Połącz…

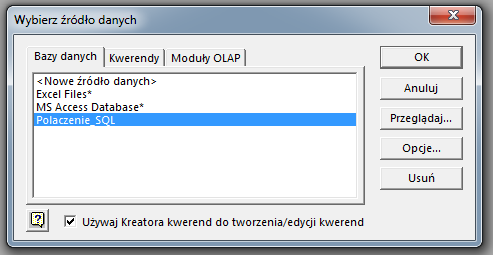
Wpisujemy nazwę serwera, po uzupełnieniu którego będzie można wybrać opcję a w nich bazę danych do połączenia



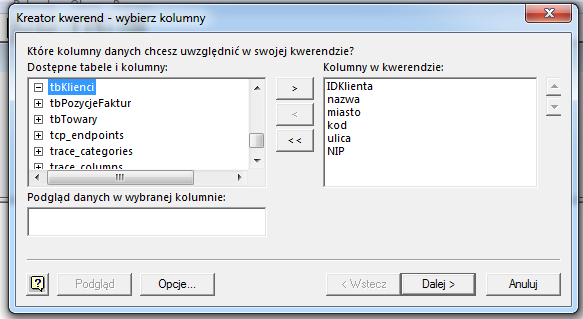
Po zatwierdzeniu tego wyboru przyciskiem OK wracamy do poprzedniego menu i wybieramy (opcjonalnie) tabelę, do której chcemy się podłączyć i OK



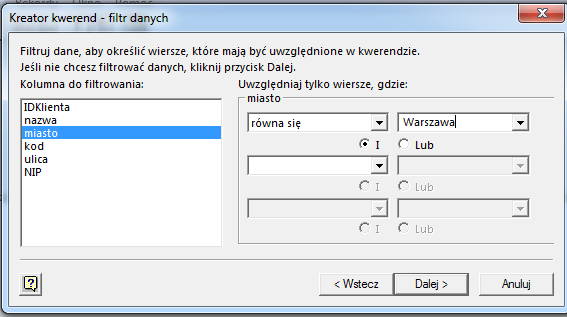
Gdy połączenie jest gotowe, można je wybrać z listy i przejść do następnego kroku poprzez OK



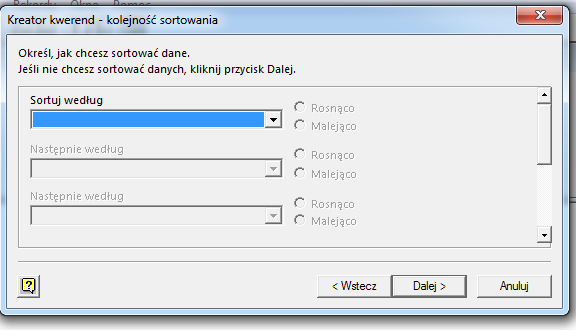
Wybieramy potrzebne kolumny i przechodzimy dalej



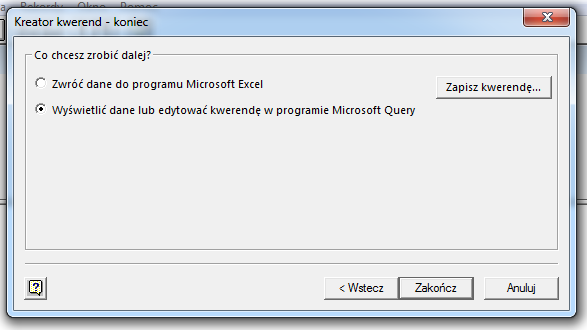
Jeżeli chcemy filtrować dane wybieramy kolumnę oraz rodzaj ograniczenia:



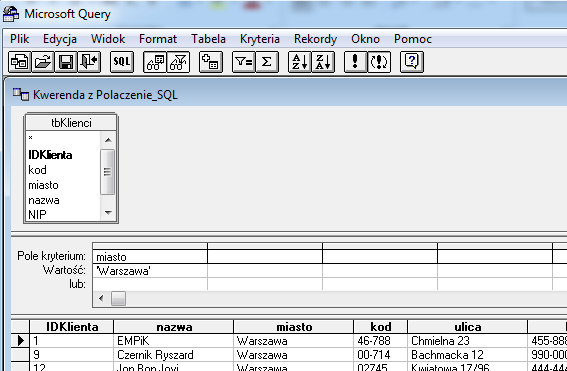
Następny krok daje możliwość sortowania

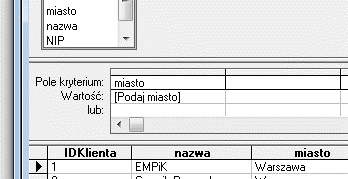


Jeżeli potrzebne jest wykonanie kwerendy parametrycznej należy wybrać opcję dalszej edycji w programie MS QUERY

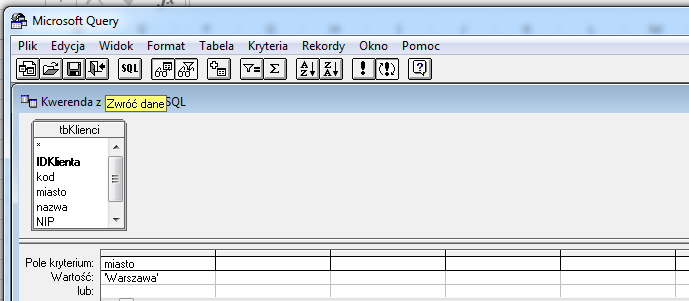


W miejsce wcześniej podanego kryterium sortowania należy podać dowolny komunikat zawarty w nawiasach kwadratowych

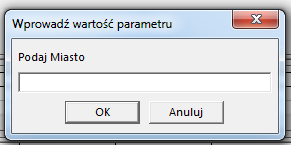




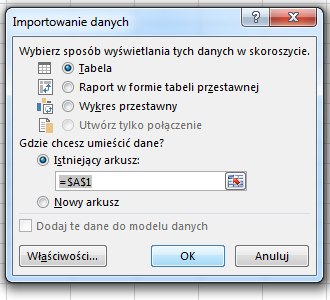
A następnie wybrać przycisk, który zwróci dane do programu MS Excel.



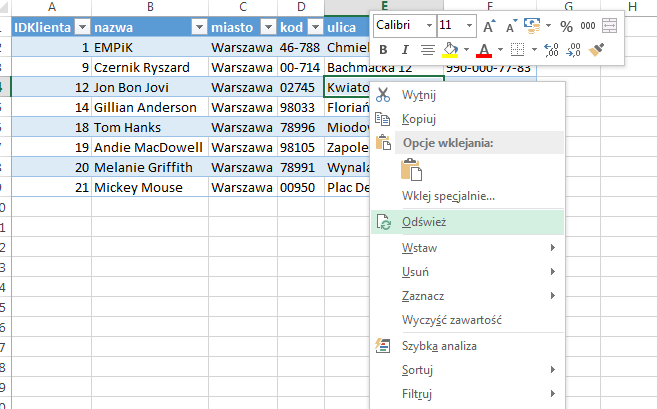
Należy podać wartość dla naszego parametru



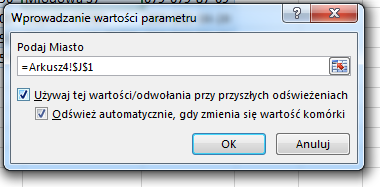
Wybieramy miejsce docelowe



Przy próbie odświeżania danych możemy zastąpić ręczne podawanie parametru na pobieranie go   
z komórki z możliwością automatycznego jego odświeżania po każdej zmianie. Aby to wykonać klikany prawy przycisk myszy na pobranych danych i odśwież



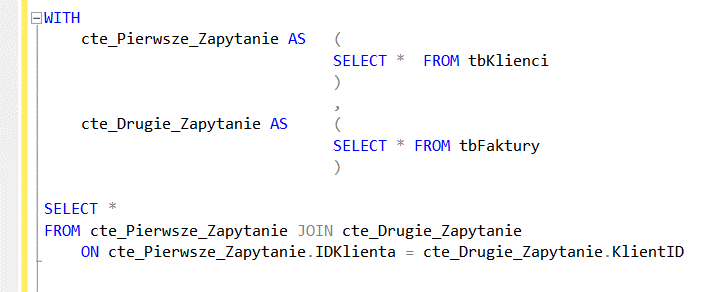
A następnie wybieramy opcje pobierania danych z komórki, wskazujemy komórkę i zaznaczamy automatyczne odświeżanie po zmianie



# CTE – Common Table Expression

Jest to metoda łączenia z wielu źródeł lub wykonywania kolejnego etapu kwerendy na wcześniej wykonanym zapytaniu. Metoda ta daje bardzo przejrzysty kod oraz pozwala na wykonywanie kwerendy element po elemencie, aby na koniec połączyć je w jedną całość. Jedną   
z większych zalet CTE jest brak konieczności posiadania dodatkowych specjalnych uprawnień do ich używania.

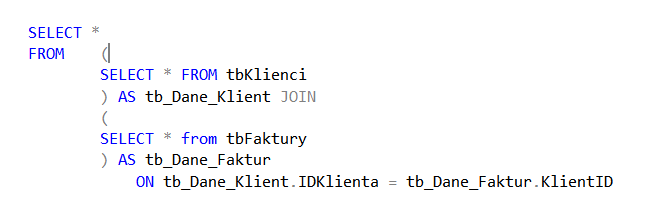
Składnia CTE:



# Podzapytania

Podobnie jak CTE podzapytania pozwalają na tworzenie złożonych kwerend w oparciu   
o wiele kwerend. Zalecane jest najpierw tworzenie wszystkich składowych a na koniec łącznie ich   
w całość, daje to możliwość przetestowania każdego elementu i zmniejsza liczbę pomyłek. Podzapytania tak jak CTE nie wymagają żadnych dodatkowych uprawnień. Należy pamiętać, że podzapytanie zawsze powinno być umieszczone w nawiasach okrągłych. Podzapytanie może stać się częścią sekcji FROM jako źródło danych, ale również częścią sekcji WHERE gdzie może posłużyć jako kryterium ograniczające wyniki.

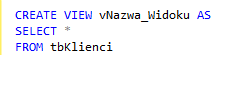
Składnia podzapytania w sekcji FROM:



# Widoki

Jednym z elementów SQL Server jest widok, jest to zapisana kwerenda, do wyników której można się odwołać poprzez nazwę widoku. Stosowane są głównie w momencie, gdy wiele osób wielokrotnie korzysta z podobnych zapytań. Zapisanie widoku nie powoduje zapisania jego wyników   
i tym samym ich zamrożenia a jedynie zapisuje zapytanie, które jest wykonywane za każdym razem, gdy odwołujemy się do tego widoku. Najczęściej tworzenie widoków dostępne jest tylko dla wybranych użytkowników, aby uniknąć nadmiernego zaśmiecania bazy. Widoki standardowo będą widoczne dla innych użytkowników mających do nich uprawnienia. Stosowanie widoków zamiast kwerendy nie przyspieszy wykonania kodu, lecz w wielu miejscach może znacząco skrócić i uprościć.

Składnia tworzenia widoku:



Gdzie widoczne są widoki

