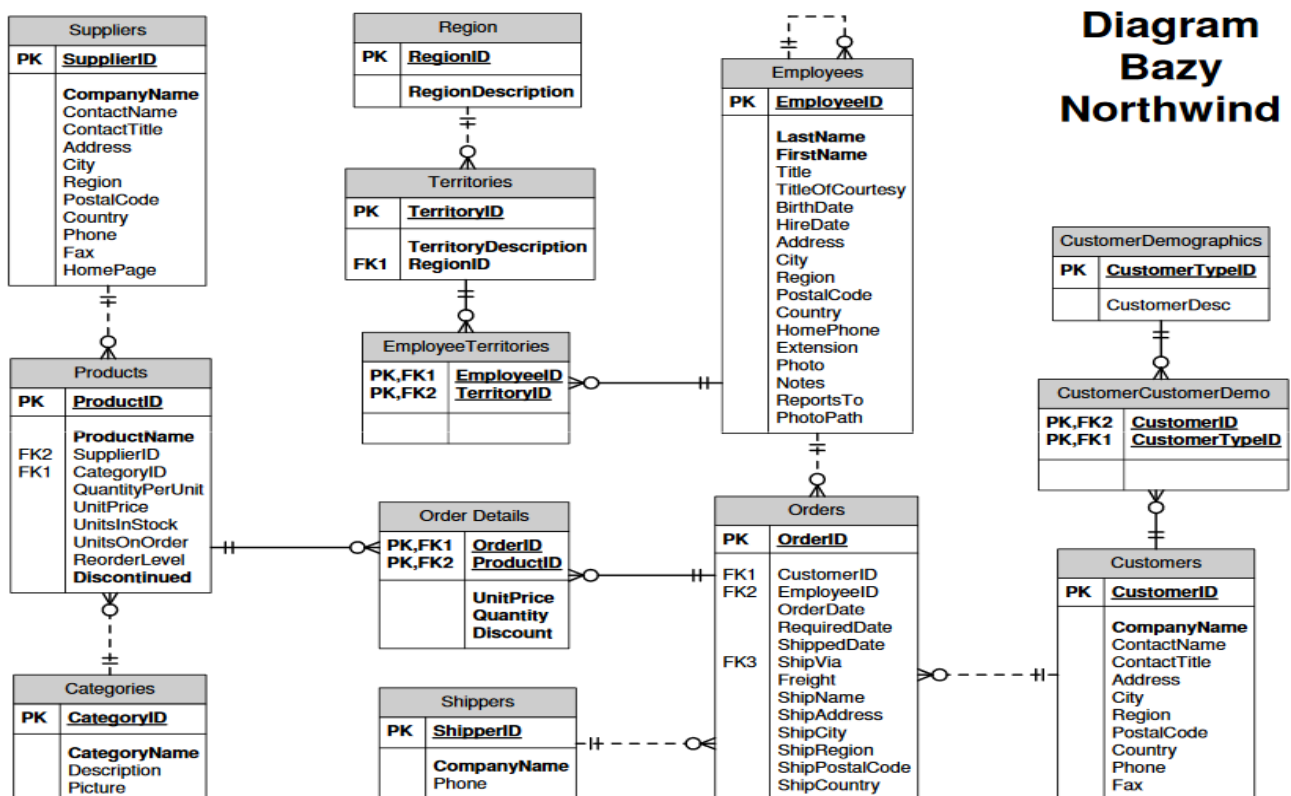


Tematy zajęć:

- 29. Stosowanie perspektyw (widoków).
- 30. Modyfikacja danych za pomocą funkcji.
- 31. Funkcje grupujące. COUNT(), MIN(), MAX(), SUM(), AVG().
- 32. Grupowanie danych. Klauzula GROUP BY. Klauzula HAVING.
- 33. Łączenie tabel. Połączenie wewnętrzne –INNER JOIN.
- 34. Połączenia zewnętrzne –LEFT OUTER JOIN, RIGHT OUTER JOIN. Połączenia krzyżowe –CROSS JOIN.

Omówienie materiału na podstawie bazy Northwind



Stosowanie perspektyw (widoków).

Widok (perspektywa) to logiczny byt (obiekt), osadzony na serwerze baz danych. Umożliwia dostęp do podzbioru kolumn i wierszy tabel lub tabeli na podstawie zapytania w języku SQL, które stanowi część definicji tego obiektu. Przy korzystaniu z widoku jako źródła danych należy odwoływać się identycznie jak do tabeli. Operacje wstawiania, modyfikowania oraz usuwania rekordów nie zawsze są możliwe (np. w sytuacji gdy widok udostępnia część kolumn dwóch tabel `tb_A` oraz `tb_B` bez kolumny z kluczem głównym tabeli `tb_B`). W niektórych SZBD widok służy tylko i wyłącznie do pobierania wyników i ograniczania dostępu do danych. Widokom również można nadać uprawnienia, np. niektórzy użytkownicy nie będą mieli dostępu do tabel, a do widoku zbudowanym na zdanej tabeli już tak.

W odróżnieniu od tabel, widoki w SQL Server nie przechowują danych. Widoków używamy w przypadkach, gdy chcemy zapisać nasze zapytanie SQL, a następnie odwołać się do danych przy użyciu krótkiego

polecenia **SELECT * FROM NAZWA_WIDOKU**. Tworzenie widoku rozpoczynamy poleceniem **CREATE VIEW**. Listę widoków możemy szybko sprawdzić w SSMS wybierając w drzewie Serwer, bazę, a następnie drzew **Views**.

Polecenie na tworzenie widoku:

```
CREATE VIEW NAZWA_WIDOKU AS  
Select * from Tabela;
```

Zmian w istniejącym widoku dokonujemy za pomocą polecenia **ALTER VIEW**. W instrukcji **SELECT** wprowadzamy zmodyfikowany kod i uruchamiamy zapytanie. Czy w takim razie nie łatwiej skasować widok, a potem utworzyć go na nowo? Tak oczywiście możemy zrobić. Pamiętaj jednak, że w miarę korzystania z bazy danych, różni użytkownicy mogą zyskać uprawnienia do obiektów takich jak tabele, czy widoki. Po skasowaniu obiektu uprawnienia także są usuwane. Stosując polecenie **ALTER** zachowujemy uprawnienia.

Sam widok usuwany jest przy pomocy polecenia **DROP**.

```
DROP VIEW NASZ_WIDOK;
```

W SQL Server możesz wykonywać polecenia DML jak **UPDATE**, **DELETE**, **INSERT** w oparciu o widoki.

UWAGA

Przy tworzeniu widoków nie można używać klauzuli **ORDER BY**! Jeżeli potrzebujemy posortowane dane, należy operację tą wykonać posługując się już stworzonym widokiem.

Przykład

Utwórz widok **v_londyn**, który wyświetli: nazwy i adresy wszystkich klientów mających siedziby w Londynie

```
CREATE VIEW v_londyn AS select CompanyName, Address from Customers where City  
= 'London';
```

Wywołanie widoku:

```
Select * from v_londyn;
```

CompanyName	Address
Around the Horn	120 Hanover Sq.
B's Beverages	Fauntleroy Circus
Consolidated Holdings	Berkeley Gardens 12 Brewery
Eastern Connection	35 King George
NorthSouth	South House 300 Queensbridge

Zadania z tworzenia widoków, wykonamy po przerobieniu materiału z łączenia tabel, i funkcji grupujących.

```
/*CREATE VIEW NAZWA_WIDOKU AS Select*.. -tworzenie widoku
DROP VIEW NAZWA_WIDOKU -usunięcie widoku
ALTER VIEW NAZWA_WIDOKU -modyfikacja widoku
*/

--1. Utwórz widok, który wyświetli ile jest klientów z danego kraju
CREATE VIEW v_liczba_klientow AS
SELECT Country, count(CustomerID) AS liczba
FROM Customers
GROUP BY country

--MODYFIKACJA WIDOKU DODANIE DODATKOWO MIASTA
ALTER VIEW v_liczba_klientow AS
SELECT Country, City, count(CustomerID) AS liczba
FROM Customers
GROUP BY country, City

-- wyświetlenie widoku i posortowanie po liczbie
SELECT * FROM v_liczba_klientow
ORDER BY liczba DESC

/* POLECENIE ORDER BY - do sortowania nie można używać przy tworzeniu widoku.
Natomiast stosuje się go na utworzonym widoku*/
--2. Usuń utworzony widok
DROP VIEW v_liczba_klientow
```

Modyfikacja danych za pomocą funkcji.

Materiał oparty na ćwiczeniach:

--konkatenacja pól- łączenie, dołączanie

```
select (Imie+ ' '+ Nazwisko) as dane_osobowe from Klienci
```

/*Funkcje tekstowe: Left, Right, Upper, Lower, Rand, Len*/

--PRZYKŁAD 1

```
select left('poniedziałek',2) as wynik;
```

```
select right('poniedziałek',3) as wynik;
```

```
select upper('poniedziałek') as wynik;
```

```
select lower(' poniedziałek ') as wynik;
```

```
select len(' poniedziałek ') as ilosc;
```

```
SELECT RAND() AS LICZBA_LOSOWA;
```

--PRZYKŁAD 2

```
USE Northwind
```

```
SELECT ProductName,
```

```
UPPER( ProductName ) AS UpperName,
```

```
LOWER( ProductName ) AS LowerName,
```

```
LEN( ProductName ) AS NameLength
```

```
FROM dbo.Products
```

```
WHERE LEN( ProductName ) < 10
```

--SUBSTRING(tekst1, numer1, numer2)- Funkcja wycina z ciągu znaków string1 ciąg rozpoczynający się w pozycji numer1 o długości numer2.

```
SELECT Country, substring(Country,5,2) as wynik1 FROM Customers
```

Results		Messages	
	wynik		
1	po		
	wynik		
1	lek		
	wynik		
1	PONIEDZIALEK		
	wynik		
1	poniedziałek		
	ilosc		
1	12		

Rysunek 1. Wyniki z przykładu 1

Results					Messages
	ProductName	UpperName	LowerName	NameLength	
1	Chai	CHAI	chai	4	
2	Chang	CHANG	chang	5	
3	Chocolate	CHOCOLADE	chocolate	9	
4	Filo Mix	FILO MIX	filo mix	8	
5	Geitost	GEITOST	geitost	7	

Rysunek 2. Wyniki z przykładu 2

Results			Messages
	Country	wynik1	
1	Germany	an	
2	Mexico	co	
3	Mexico	co	

Rysunek 3. Przykład z substring

--REPLACE(tekst1, tekst2, tekst3) – Funkcja zamienia w ciągu tekstowym tekst1 znaki podane w argumencie tekst2 na znaki podane w argumencie tekst3. Zamienić możemy zarówno pojedynczy znak, jak i całe frazy.

```
SELECT COUNTRY,
replace(COUNTRY, 'France', 'Francja') as wynik FROM CUSTOMERS;
```

	COUNTRY	wynik
1	Germany	Germany
2	Mexico	Mexico
3	Mexico	Mexico
4	UK	UK
5	Sweden	Sweden
6	Germany	Germany
7	France	Francja
8	Spain	Spain
9	France	Francja

--reverse -lustro

```
SELECT REVERSE('KATARZYNA') AS LUSTRO;
```

-- Zadanie: ODCZYTAJ SVOJE IMIĘ JAKO ODBICIE LUSTRZANE

/*Funkcje daty: getdate(),datediff */

--Funkcje daty

```
select getdate() as data_biezaca;
```

	data_biezaca
1	2021-01-04 22:48:23.110

--datediff(element_daty,dataPoczątkowa>DataKońcowa)- różnica pomiędzy datami

```
select datediff(day, '2021-01-05','2021-01-31') as roznica_dni;
```

```
select datediff(week, '2021-01-31','2021-02-05') as
```

```
roznica_tygodni;
```

```
select datediff(MONTH, '2020-09-01','2020-12-31') as
```

```
roznica_miesiecy;
```

```
select datediff(YEAR, '1990-01-01','2021-01-01') as roznica_lat;
```

-- ZADANIE: POLICZ ILE MASZ LAT

--wyświetl wg wzoru: I. Nazwisko mieszka Miasto z tabeli pracownicy

```
select (left(FirstName,1)+ '. ' + LastName+ ' mieszka w ' +City)
as dane_osobowe from Employees
```

	roznica_dni
1	26
	roznica_tygodni
1	0
	roznica_miesiecy
1	3
	roznica_lat
1	31

	dane_osobowe
1	N. Davolio mieszka w Seattle
2	A. Fuller mieszka w Tacoma
3	J. Leverling mieszka w Kirkland
4	M. Peacock mieszka w Redmond
5	S. Buchanan mieszka w London

/* Funkcje matematyczne:

ABS() – Zwraca wartość bezwzględną.

Pi() – Wartość liczby PI ~3,14159265358979

SQUARE() – Kwadrat liczby.

POWER(arg1, arg2) – Potęgowanie. Argument 1 określa podstawę potęgi. Argument 2 to wykładnik potęgi.

SQRT() – Pierwiastek kwadratowy z podanej wartości liczbowej.

ROUND(liczba, arg 2) – Funkcja zaokrągla liczbę do miejsc po przecinku wprowadzonych w argumencie 2. Argument może przyjmować wartość 0 – uzyskamy w ten sposób liczbę całkowitą. Argument może także przyjąć wartości ujemne – zaokrągla w ten sposób liczbę przed przecinkiem.

Należy znać te 6 funkcje- pozostałe do przeanalizowania i sprawdzenia jak działają.

SIGN() – Zwraca 3 wartości w zależności od znaku liczby: -1 / 0 / 1

RAND() – Losuje liczbę FLOAT z przedziału (0, 1)

CEILING() – Zwraca liczbę całkowitą najbardziej zbliżona do argumentu.

Zaokrągla do góry.

FLOOR() – Zwraca liczbę całkowitą najbardziej zbliżona do argumentu. Zaokrągla do dołu.

EXP() – Funkcja zwraca wartość wykładniczą. Dla przykładu EXP(1) zwróci nam liczbę Eulera ~2,71828182845905

SIN(), COS(), TAN(), COT(), ASIN(), ACOS(), ATAN(), ACOT() – Zestaw funkcji trygonometrycznych i cyklometrycznych.

LOG(), LOG10() – Funkcje wykładnicze.

DEGREES() – Zwraca wartość kąta w stopniach.

RADIANS() – Zwraca wartość kąta w radianach.

ATN2(x,y) – Zwraca wartość w radianach pomiędzy punktami na osiach x i y.

~~zmienić

*/

--Przykład użycia funkcji matematycznych

select

ABS(-10) as o1,

PI() as o2,

SQUARE(4) as o3,

POWER(2, 3) as o4,

SQRT(100) as o5,

ROUND(2222.5544, 2) as o6,

SIGN(-123) as o7,

RAND() as o8,

CEILING(\$333.23456) as o9,

FLOOR(\$333.23456) as o10,

exp(1) as o11

;

	o1	o2	o3	o4	o5	o6	o7	o8	o9	o10	o11
1	10	3,14159265358979	16	8	10	2222.5500	-1	0,447312436151511	334,00	333,00	2,71828182845905

```

/*
SIN(), COS(), TAN(), COT(), ASIN(), ACOS(), ATAN(), ACOT() – Zestaw funkcji
trygonometrycznych i cyklometrycznych.
LOG(), LOG10() – Funkcje wykładnicze.
DEGREES() – Zwraca wartość kąta w stopniach.
RADIANS() – Zwraca wartość kąta w radianach.
ATN2(x,y) – Zwraca wartość w radianach pomiędzy punktami na osiach x i y.
~~zmienić
*/

```

```

select
SIN(0) AS RES1,
COS(0) AS RES2,
TAN(0) AS RES3,
COT(PI()/4) AS RES4,
POWER(4, LOG(2)) AS RES5,
LOG10(10) AS RES6,
DEGREES(PI()/2) AS RES7,
RADIANS(360) AS RES8,
ATN2(1,6) AS RES9;

```

Results		Messages							
	RES1	RES2	RES3	RES4	RES5	RES6	RES7	RES8	RES9
1	0	1	0	1	2	1	90	6	0,165148677414627

Funkcje grupujące. COUNT(), MIN(), MAX(), SUM(), AVG(). Grupowanie danych. Klauzula GROUP BY. Klauzula HAVING.

Funkcje agregujące, to takie funkcje, które zwracają jedną wartość wyliczoną na podstawie wielu wierszy.

<i>Funkcja agregująca</i>	<i>Opis</i>
AVG	średnia wartości w wyrażeniu numerycznym
COUNT	liczba wartości w wyrażeniu
COUNT (*)	liczba wybranych wierszy
MAX	największa wartość w wyrażeniu
MIN	najmniejsza wartość w wyrażeniu
SUM	ogólna wartość w wyrażeniu numerycznym

Użycie funkcji agregujących z wartościami Null

- Większość funkcji agregujących ignoruje wartości Null
- Funkcja COUNT(*) zlicza wiersze z wartościami Null

Przykłady użycia funkcji agregujących:

USE Northwind

- `select AVG(UnitPrice) AS MAKSYMALNA FROM Products; --wyświetla średnią cenę jednostkową dla wszystkich produktów`
- `select SUM(Quantity) AS ILOSC FROM [Order Details];-- wyświetla sumaryczną ilość sprzedanych produktów`
- `SELECT MAX(UnitPrice)AS MAKSYMALNA, MIN(UnitPrice)AS MINIMALNA, AVG(UnitPrice) AS SREDNIA, SUM(UnitPrice) AS SUMA, COUNT(UnitPrice)AS ILOSC FROM Products;`
- `SELECT COUNT(*)AS LICZNA_PRODUKTOW FROM Products WHERE UnitPrice > 20; --SELECT COUNT('cokolwiek -dowolna kolumna') FROM Products`

161 %	
Results Messages	
LACZNA_CENA	2222,71
LACZNA_CENA_BEZ_DUPLIKATOW	1947,81

Użycie klauzuli DISTINCT w powyższej funkcji powoduje, że wiersze, dla których agregowane wyrażenie się powtarza, agregacji podlegają tylko jeden raz. Przykładowo jeśli do przykładu pokazanego powyżej

dodamy „distinct” przed unitprice, otrzymamy sumę cen bez powtórek – tj. jeśli cena 2 produktów będzie taka sama – wartość tych cen produktów policzona zostanie tylko raz.

Wyświetlenie n początkowych wartości (TOP n)

- Wyświetlanie tylko pierwszych n wierszy zbioru wynikowego
- Określenie zakresu wartości w klauzuli ORDER BY
- Przykład: wyświetl 4 najdroższe produkty – wynikiem ma być: nazwa produktu, cena i nr_kategorii.

USE Northwind

```
select TOP 4 ProductName, UnitPrice, CategoryID FROM Products
ORDER BY UnitPrice DESC;
```

Results		Messages	
	ProductName	UnitPrice	CategoryID
1	Côte de Blaye	263,50	1
2	Thüringer Rostbratwurst	123,79	6
3	Mishi Kobe Niku	97,00	6
4	Sir Rodney's Marmalade	81,00	3

Ćwiczenie 1 z grupowania:

Grupowanie, funkcje agregujące - Baza NORTHWIND

- Policz średnią cenę jednostkową dla wszystkich produktów w tabeli products.
- Zsumuj wszystkie wartości w kolumnie quantity w tabeli order details
- Podaj liczbę produktów o cenach mniejszych niż 10\$ lub większych niż 20\$
- Podaj maksymalną cenę produktu dla produktów o cenach poniżej 20\$
- Podaj maksymalną i minimalną i średnią cenę produktu dla produktów o produktach sprzedawanych w butelkach (bottle)
- Wypisz informację o wszystkich produktach o cenie powyżej średniej
- Podaj sumę zamówienia o numerze 10250

Grupowanie -GROUP BY

- Użycie klauzuli GROUP BY
- Użycie klauzuli GROUP BY z klauzulą HAVING

Grupowanie polega na podzieleniu zbioru wierszy na grupy, które mają pewną wspólną cechę. Grupowania dokonuje się w celu zastosowania funkcji agregujących nie w stosunku do całego zbioru wierszy, ale do poszczególnych grup wierszy. W celu zgrupowania rekordów należy dodać nową klauzulę GROUP BY wraz z wyspecyfikowaniem kolumny lub wyrażenia, według którego mają być pogrupowane wiersze. Na liście klauzuli SELECT mogą się znaleźć tylko kolumny i wyrażenia, według których zapytanie jest grupowane, oraz wywołania funkcji agregujących.

Przed dokonaniem grupowania można zastosować klauzulę WHERE, która wybierze tylko część wierszy z tabeli.

Klauzula having

Klauzula WHERE wykonuje się przed grupowaniem, a zatem nie można w tej klauzuli sprecyzować warunku zawierającego funkcje grupowe. Aby taki warunek zawrzeć w zapytaniu należy zastosować dodatkową klauzulę HAVING wraz z odpowiednim warunkiem. Jest ona odpowiednikiem klauzuli WHERE, tylko, że wykonuje się ona po procesie grupowania.

Przykłady:

```
select count(CustomerID) AS LICZBA_KLIENTÓW, Country from Customers
GROUP BY Country
ORDER BY LICZBA_KLIENTÓW DESC;
```

-- Wyświetla kraje i ich liczbę klientów, których jest więcej lub równo 5, oprócz Francji

```
select count(CustomerID) AS LICZBA_KLIENTÓW, Country from Customers
WHERE Country <> 'France'
GROUP BY Country
HAVING count(CustomerID) >= 5
ORDER BY LICZBA_KLIENTÓW DESC;
```

Przykład: Napisz polecenie, które zwraca informacje o zamówieniach

- zapytanie ma grupować i wyświetlać identyfikator każdego produktu, a następnie obliczać ogólną zamówioną ilość
- ogólna ilość jest sumowana funkcją agregującą SUM i wyświetlana jako jedna wartość dla każdego produktu.

USE Northwind

```
select ProductID, SUM(Quantity) AS ILOSC_CAŁKOWITA FROM [Order Details]
GROUP BY ProductID;
```

Przykład:

```
--6. Wypisz informację o wszystkich produktach o cenie powyżej średniej
SELECT AVG( UnitPrice ) FROM Products;
SELECT * FROM Products
WHERE
UnitPrice > (SELECT AVG( UnitPrice ) FROM Products)
ORDER BY UnitPrice;
```

Ćwiczenie 2 z grupowania:

7. Podaj sumę zamówienia o numerze 10250
8. Napisz polecenie, które zwraca informacje o zamówieniach z tablicy order details. Zapytanie ma grupować i wyświetlać identyfikator każdego produktu a następnie obliczać ogólną zamówioną ilość. Ogólna ilość jest sumowana funkcją agregującą SUM i wyświetlana jako jedna wartość dla każdego produktu.
9. Podaj maksymalną cenę zamawianego produktu dla każdego zamówienia
10. Posortuj zamówienia wg maksymalnej ceny produktu
11. Podaj maksymalną i minimalną cenę zamawianego produktu dla każdego zamówienia
12. Podaj liczbę zamówień dostarczanych przez poszczególnych spedytorów
13. Który z spedytorów był najaktywniejszy w 1997 roku

Przykład na użycie HAVING:

--Wyświetl listę identyfikatorów produktów i ilość dla tych produktów, których zamówiono ponad 1200 jednostek

USE Northwind

```
select ProductID, SUM(Quantity) AS ILOSC_CAŁKOWITA
FROM [Order Details]
GROUP BY ProductID
HAVING SUM(Quantity)>1200;
```

Ćwiczenie 3 z grupowania:

14. Wyświetl zamówienia dla których liczba pozycji zamówienia jest większa niż 5
15. Wyświetl klientów którzy dla których w 1998 roku zrealizowano więcej niż 8 zamówień (wyniki posortuj malejąco wg łącznej kwoty za dostarczenie zamówień dla każdego z klientów)

33. Łączenie tabel. Połączenie wewnętrzne –INNER JOIN.

Złączenia tabel

Dane, które chcemy uzyskać z bazy danych, zazwyczaj znajdują się w więcej niż jednej tabeli bazodanowej. W takim przypadku nasze zapytanie musi się odpowiednio odwołać do więcej niż jednej tabeli. Mechanizm ten nazywa się w języku SQL złączeniami.

Iloczyn kartezjański (ang. Cross Join)

```
SELECT Products.ProductName, Products.UnitPrice, Categories.CategoryName
FROM Categories, Products ;
```

W tym przykładzie zapomniano o warunku złączeniowym. Efektem jest iloczyn kartezjański relacji **Products** oraz **Categories**. Wynik liczy 616 rekordów (77 rekordów w tabeli produkty razy 8 rekordów w tabeli kategorie). Zapytanie zwracające iloczyn kartezjański nazywa się w terminologii bazodanowej **cross-join**.

W taki sposób jak iloczyn kartezjański- nie będziemy łączyć tabel.

Połączenia INNER JOIN

Operator **INNER JOIN** jest jednym z najczęściej wykorzystywanych operatorów złączeń i w pewnym sensie stanowi on odpowiednik znaku równości pomiędzy tabelami. Przykładowo, jeśli w jednej tabeli mamy dane o produktach a w drugiej ich nazwy kategorii, to gdy chcemy wyświetlić w jednym wyświetleniu wyniku zapytania te informacje, musimy połączyć obie tabele, używając elementu wspólnego.

Przykład:

Wyświetla nazwy i ceny produktów z tabeli **produkty**, i nazwę kategorii z tabeli **kategorie**.

```
SELECT Products.ProductName, Products.UnitPrice, Categories.CategoryName
FROM      Categories INNER JOIN
```

```
Products ON Categories.CategoryID = Products.CategoryID;
```

Wyjaśnienie: tabele **Categories** i **Products** łączą się po **CategoryID**. Najczęściej klucz główny z jednej tabeli łączy się z kluczem obcym w drugiej tabeli. Kolejność wyświetlania złączenia nie jest istotna.

W poleceniu **SELECT** nazwy kolumn poprzedziliśmy nazwą tabeli. Gdy nazwy kolumn są unikalne, to nie jest to konieczne, aczkolwiek zalecane.

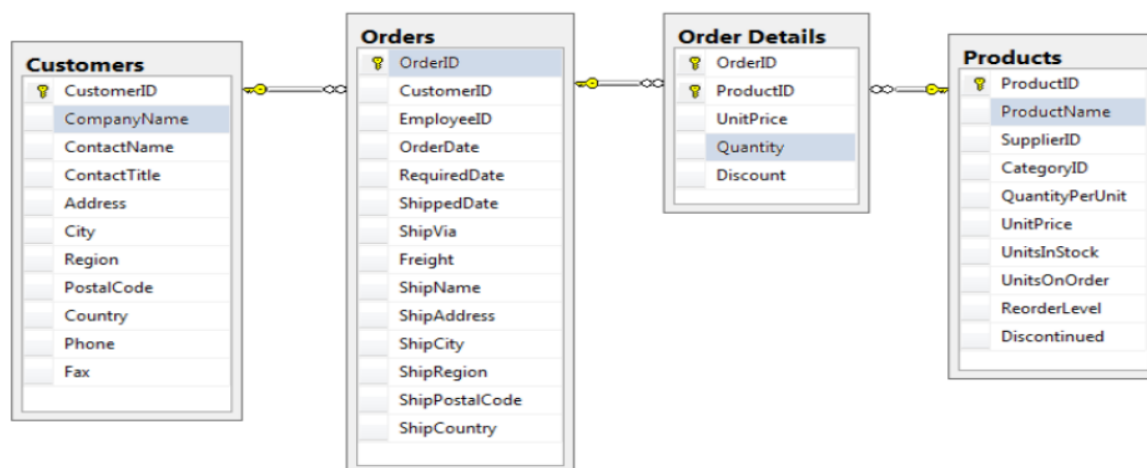
Dla zwartości zapisu można zamiast całych nazw tabel zdefiniować aliasy.

Ten sam przykład z użyciem aliasów:

```
SELECT P.ProductName, P.UnitPrice, C.CategoryName
FROM      Categories AS C INNER JOIN
           Products AS P ON Categories.CategoryID =
Products.CategoryID;
```

Przykład na łącznie więcej niż 2 tabel:

Interesują nas informacje o Klientach z określonego miasta –Madrytu. Chcemy znać detale ich zamówień, złożonych we wrześniu 1996 –nazwy i ilości produktów, które kupili.



```

SELECT Customers.CompanyName, Orders.OrderID, [Order Details].Quantity,
Products.ProductName, Orders.OrderDate
FROM      Customers INNER JOIN
           Orders ON Customers.CustomerID = Orders.CustomerID INNER
JOIN
           [Order Details] ON Orders.OrderID = [Order Details].OrderID
INNER JOIN
           Products ON [Order Details].ProductID = Products.ProductID

WHERE (Customers.City = N'Madrid') AND (Orders.OrderDate BETWEEN '1996-09-
01' AND '1996-09-30');
  
```

161 %

Results Messages

	CompanyName	OrderID	Quantity	ProductName	OrderDate
1	Romero y tomillo	10306	10	Nord-Ost Matjeshering	1996-09-16 00:00:00.000
2	Romero y tomillo	10306	10	Perth Pasties	1996-09-16 00:00:00.000
3	Romero y tomillo	10306	5	Tourtière	1996-09-16 00:00:00.000

Rysunek 4. Wynik powyższego zapytania

Przykład: Napisz polecenie zwracające jako wynik nazwy klientów, którzy złożyli zamówienia po 01 marca 1998 / 2 sposoby

--A)

```

SELECT DISTINCT CompanyName
FROM
Customers, Orders
WHERE
Customers.CustomerID = Orders.CustomerID
AND
( (YEAR(Orders.OrderDate) = 1998
AND
MONTH(Orders.OrderDate) > 2) OR
YEAR(Orders.OrderDate) > 1998 )
  
```

--B)

```

select distinct Customers.CompanyName
from Customers
  
```

```

join Orders
on Orders.CustomerID = Customers.CustomerID
where Orders.OrderDate > '3/1/98'

```

Ćwiczenie 1 z łączenia tabel. Baza NORTHWIND

1. Napisz polecenie zwracające nazwy produktów i firmy je dostarczające (tak aby produkty bez dostarczycieli i dostarczyciele. bez produktów nie pojawiały się w wyniku).
2. Napisz polecenie zwracające wszystkich klientów z datami zamówień.
1. Wybierz nazwy i ceny produktów o cenie jednostkowej pomiędzy 20.00 a 30.00, dla każdego produktu podaj dane adresowe dostawcy
2. Wybierz nazwy produktów oraz inf. o stanie magazynu dla produktów dostarczanych przez firmę Tokyo Traders
3. Wybierz nazwy i numery telefonów dostawców, dostarczających produkty, których aktualnie nie ma w magazynie
4. Napisz polecenie zwracające listę produktów zamawianych w dniu 1996
5. Wybierz nazwy i ceny produktów o cenie jednostkowej pomiędzy 20.00 a 30.00, dla każdego produktu podaj dane adresowe dostawcy, interesują nas tylko produkty z kategorii Meat/Poultry.
6. Wybierz nazwy i ceny produktów z kategorii Confections dla każdego produktu podaj nazwę dostawcy.
7. Wybierz nazwy i numery telefonów klientów, którym w 1997 roku przesyłki dostarczała firma United Package.
8. 8. Wybierz nazwy i numery telefonów klientów, którzy kupowali produkty z Kategorii Confections.

34. Połączenia zewnętrzne –LEFT OUTER JOIN, RIGHT OUTER JOIN. Połączenia krzyżowe –CROSS JOIN.

LEFT OUTER JOIN, RIGHT OUTER JOIN

Złączenie typu **OUTER JOIN**, zwane inaczej złączeniami zewnętrznymi, pozwala nam na uwzględnienie w wyniku danych, które nie posiadają swoich odpowiedników w złączanych tabelach. Oznacza to, że jeśli w jednej tabeli pojawiają się wiersze, które nie posiadają odpowiedników w drugiej tabeli to zostaną wzięte pod uwagę podczas złączenia ale puste kolumny zostaną wypełnione wartościami null.

Aby wykonać takie złączenie musimy skorzystać z operatora złączenia zewnętrznego, czyli słowa **outer**, a także wskazać z której tabeli mają zostać pobrane wiersze, które nie posiadają dopasowania w innej tabeli. Tabelę taką skazujemy za pomocą słowa **left**, jeśli znajduje się ona na lewo od słów **outer join** w konstrukcji naszego zapytania lub **right**, jeśli tabela ta znajduje się po prawej stronie od tych słów. Opcjonalnie możemy zamiast słów **left** bądź **right** użyć słowa **full**, wtedy z obu tabel zostaną pobrane wszystkie rekordy, nawet jeśli nie będą one miały dopasowania.

LEFT OUTER JOIN	RIGHT OUTER JOIN	FULL OUTER JOIN

Przykład: Wyświetl klientów, którzy nie składali zamówień:

```
SELECT Customers.CompanyName, Orders.CustomerID
FROM Customers LEFT JOIN
      Orders ON Customers.CustomerID = Orders.CustomerID
WHERE (Orders.CustomerID IS NULL)
```

Lub można zobaczyć zastosowanie full join:

```
SELECT Customers.CompanyName, Orders.CustomerID
FROM Customers FULL JOIN
      Orders ON Customers.CustomerID = Orders.CustomerID
WHERE (Orders.CustomerID IS NULL)
```

WYNIK:

	CompanyName	CustomerID
1	FISSA Fabrica Inter. Salchichas S.A.	NULL
2	Paris spécialités	NULL

Zadanie połączenia zewnętrzne:

1. Czy są jacyś klienci którzy nie złożyli żadnego zamówienia w 1997 roku, jeśli tak to pokaż ich dane adresowe.
2. Czy są jakieś produkty, które nie zostały zamówione?
3. Czy są produkty, bez przypisanej kategorii?

Połączenia SELF JOIN

O złączeniu **self join** mówimy, gdy zachodzi potrzeba odwołania się w jednym zapytaniu dwukrotnie do tej samej tabeli a więc po lewej i prawej stronie złączenia występuje ta sama tabela. W tym typie złączeń za niezbędne uważa się użycie aliasów.

Połączenia typu SELF JOIN to zawsze jedno z już poznanych – INNER, CROSS lub OUTER JOIN, w T-SQL nie stosuje się zapisu SELF JOIN.

Iloczyn kartezjański (ang. Cross Join)

```
SELECT Products.ProductName, Products.UnitPrice, Categories.CategoryName
FROM Categories, Products ;
```

Zamiast domyślnego wyświetlania tabel po przecinku wpisuje się **cross join**

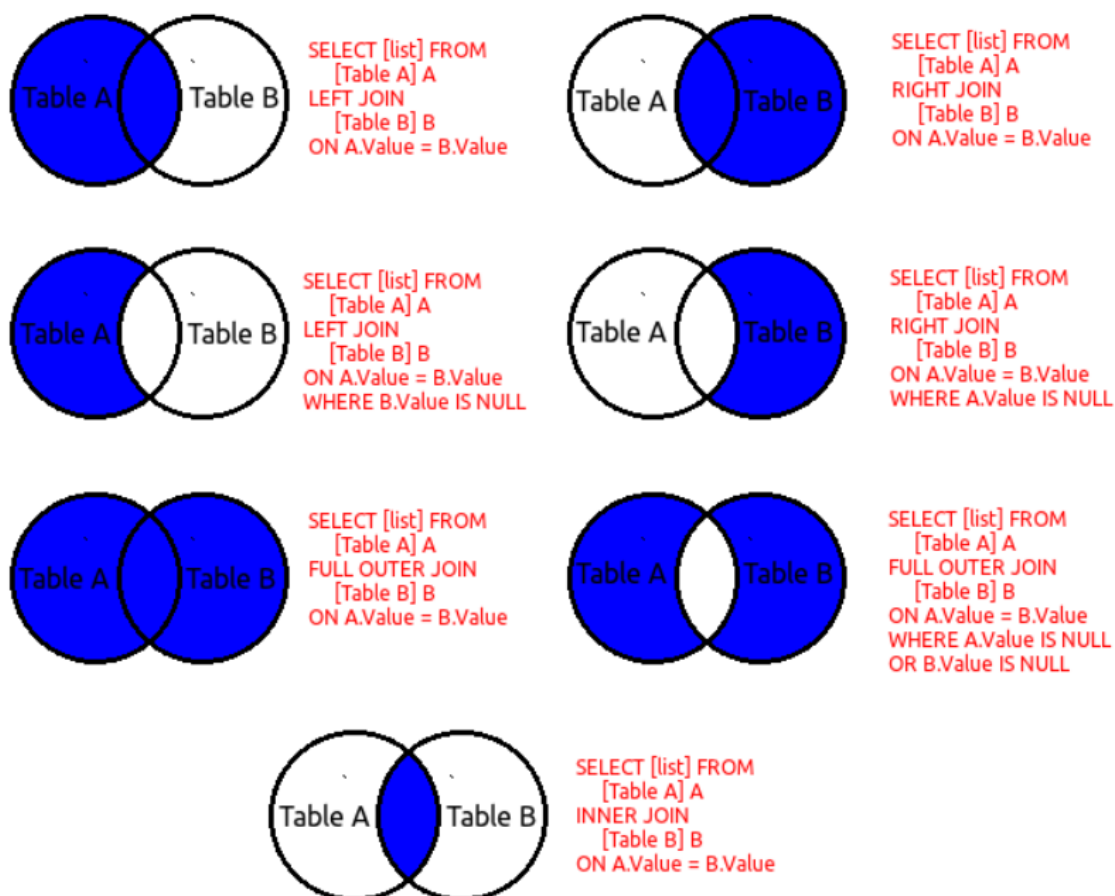
```
SELECT Products.ProductName, Products.UnitPrice, Categories.CategoryName
FROM Categories CROSS JOIN Products ;
```

W tym przykładzie zapomniano o warunku złączeniowym. Efektem jest iloczyn kartezjański relacji Products oraz Categories. Wynik liczy 616 rekordów (77 rekordów w tabeli produkty razy 8 rekordów w tabeli kategorie). Zapytanie zwracające iloczyn kartezjański nazywa się w terminologii bazodanowej **cross-join**.

Ćwiczenie: Napisz polecenie wyświetlające CROSS JOIN między shippers i suppliers. (wszystkie możliwe sposoby w jaki dostawcy mogą dostarczyć swoje produkty).

Podsumowanie informacji o JOIN-ACH:

- JOIN to to samo co INNER JOIN,
- LEFT JOIN to to samo co LEFT OUTER JOIN,
- RIGHT JOIN to to samo co RIGHT OUTER JOIN,
- FULL JOIN to to samo co FULL OUTER JOIN,
- CROSS JOIN to to samo co iloczyn kartezjański.



JOIN (INNER JOIN)

- Jest to złączenie wewnętrzne i jako wynik otrzymujemy tylko dopasowane wiersze.
- Stosując samego JOIN'a SZBD domyślnie stosuje INNER, czyli typ złączenia wewnętrznego.
- W złączeniu tego typu nie ma znaczenia kolejność tabel, w każdym przypadku wynik będzie tożsamy.
- Kolejność warunków umieszczonych w ON także nie ma znaczenia.

OUTER JOIN

- Jest to złączenie zewnętrzne będące rozszerzeniem złączenia wewnętrznego.
- Działa analogicznie jak złączenie wewnętrzne, z tym, że wymagane jest dodatkowe określenie nie spełniające warunków zapytania.
- Dodatkowo możemy zastosować: LEFT, RIGHT oraz FULL.

LEFT JOIN (LEFT OUTER JOIN)

- Jest to złączenie zewnętrzne lewostronne.
- W wyniku zwracane są wszystkie wiersze występujące w tabeli po lewej stronie.
- Do powyższego wyniku dopasowywane są wiersze z prawej tabeli.
- W przypadku braku połączenia otrzymujemy wartości puste, czyli NULL'e.

RIGHT JOIN (RIGHT OUTER JOIN)

- Jest to złączenie zewnętrzne prawostronne.
- W wyniku zwracane są wszystkie wiersze występujące w tabeli po prawej stronie.
- Do powyższego wyniku dopasowywane są wiersze z lewej tabeli.
- W przypadku braku połączenia otrzymujemy wartości puste, czyli NULL'e.

FULL JOIN (FULL OUTER JOIN)

- Jest to złączenie zewnętrzne pełne.
- W wyniku zwracane są wszystkie wiersze wewnętrznym dopasowane.
- Do powyższego wyniku dopisywane są wszystkie niepołączone wiersze z obydwu tabel uzupełnione o wartości puste, czyli NULL'e.

CROSS JOIN

- Jest to złączenie wszystkich wierszy jednej tabeli ze wszystkimi wierszami z drugiej tabeli.
- Tak zwany iloczyn kartezjański
- Nie ma sensu korzystania z warunków w ON, ponieważ z założenia ma być łączone wszystko.
- Trzeba zachować szczególną ostrożność przy stosowaniu łączenia tego typu. Zakładając, że każda z tabel zawiera po 1 000 wierszy, to w rezultacie otrzymamy 1 000 000 wierszy!

SELF JOIN

- Jest to połączenie tabeli samej z sobą.
- Zapis SELF JOIN nie jest dostępny, używa się w tym celu aliasów (AS).
- W tym rodzaju złączenia można dokonywać połączeń po wszystkich kolumnach.

Wszystkie rodzaje złączeń dla zainteresowanych- więcej info

https://pl.wikibooks.org/wiki/SQL/Typy_z%C5%82%C4%85cze%C5%84

- 1 Typy złączeń, informacje wstępne
 - 1.1 Wewnętrzne złączenia (INNER JOIN)
 - 1.1.1 Notacja **JOIN ON**
 - 1.1.2 Złączenie typu **INNER JOIN**
 - 1.1.3 Złączenie typu **Self JOIN**
 - 1.1.4 Złączenie typu **Equi-JOIN**
 - 1.1.5 Złączenie typu **Natural Join**
 - 1.1.6 Złączenie typu **Theta Join**
 - 1.1.7 Złączenie typu **Anti Join**
 - 1.1.8 Złączenie typu **Semi Join**
 - 1.1.9 Złączenie typu **CROSS JOIN**
 - 1.2 Zewnętrzne złączenia (OUTER JOIN)
 - 1.2.1 Złączenie typu **LEFT OUTER JOIN**
 - 1.2.2 Złączenie typu **RIGHT OUTER JOIN**
 - 1.2.3 Złączenie typu **FULL OUTER JOIN**

Rysunek 5. Dla zainteresowanych- materiał dodatkowy
