Hurtownie danych  
Laboratorium Czw 11:15  
  
Lista 1  
  
Kajetan Pynka 254495

**Zad 1.1**

Reg/05 - Produkt może być sprzedawany w wielu sklepach

Reg/06 - Zakup może dotyczyć wielu produktów

Reg/07 - Zakup musi dotyczyć przynajmniej jednego produktu

Reg/08 - Zakup każdego produktu obarczony jest nabyciem określonej ilości danego produktu za konkretną cenę

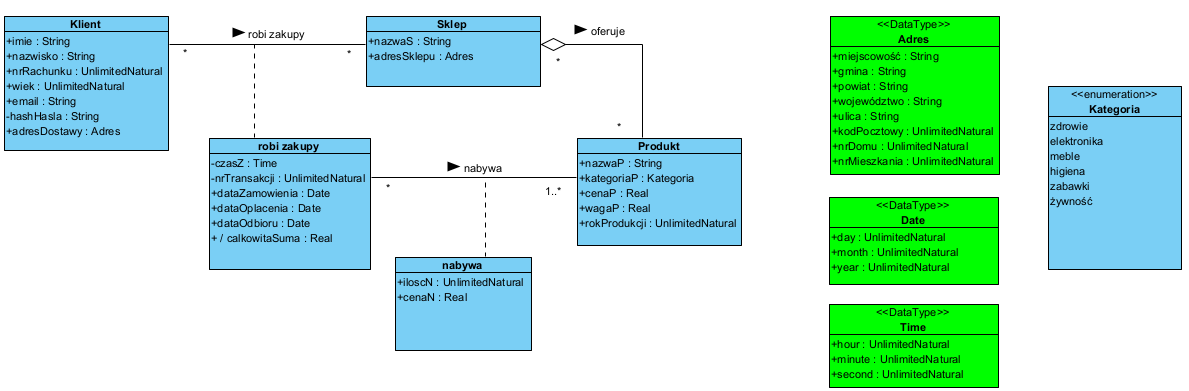
Reg/09 - Produkt może być kupowany wiele razy

Reg/10 – Klient może robić zakupy w wielu sklepach

Reg/11 – Sklep może posiadać wiele produktów

Pozbyłem się Reg/04

**Zad 1.2**



Dodałem atrybuty do większości klas. Utworzyłem klasy Adres, Date, Time (które są stereotypu DataType, tzn. mogą być wykorzystane jako typy atrybutów w zwykłych klasach). Dodatkowo wzbogaciłem produkt o kategorię, która jest typem wyliczeniowym konkretnych wartości. Pozbyłem się ograniczenia na „przynajmniej jeden produkt w sklepie”.

**Zad 1.3**

CREATE TABLE Adres (

idA INTEGER PRIMARY KEY,

miejscowosc VARCHAR(50) NOT NULL,

gmina VARCHAR(50) NOT NULL,

powiat VARCHAR(50) NOT NULL,

wojewodztwo VARCHAR(50) NOT NULL,

ulica VARCHAR(50) NOT NULL,

kodPocztowy INTEGER NOT NULL,

nrDomu INTEGER NOT NULL,

nrMieszkania INTEGER

);

CREATE TABLE Klient (

idK INTEGER PRIMARY KEY,

imie VARCHAR(50) NOT NULL,

nazwisko VARCHAR(50) NOT NULL,

nrRachunku VARCHAR(20) UNIQUE,

wiek INTEGER,

email VARCHAR(30) NOT NULL UNIQUE,

hashHasla VARCHAR(256) NOT NULL UNIQUE,

adresDostawy INTEGER FOREIGN KEY REFERENCES Adres(idA) UNIQUE

);

CREATE TABLE Sklep (

idS INTEGER PRIMARY KEY,

nazwaS VARCHAR(30) NOT NULL,

adresSklepu INTEGER FOREIGN KEY REFERENCES Adres(idA) NOT NULL UNIQUE

);

CREATE TABLE Kategoria (

nazwaKat VARCHAR(30) PRIMARY KEY

);

CREATE TABLE Produkt(

idP INTEGER PRIMARY KEY,

nazwaP VARCHAR(50) NOT NULL,

kategoriaP VARCHAR(30) REFERENCES Kategoria(nazwaKat) NOT NULL,

cenaP DECIMAL(30,4) NOT NULL,

wagaP REAL,

rokProdukcji INTEGER

);

CREATE TABLE ProduktSklepowy(

idPS INTEGER PRIMARY KEY,

idSklepu INTEGER REFERENCES Sklep(idS) NOT NULL,

idProduktu INTEGER REFERENCES Produkt(idP) NOT NULL

);

CREATE TABLE RobiZakupy (

nrTransakcji INTEGER PRIMARY KEY,

idKlienta INTEGER REFERENCES Klient(idK) NOT NULL,

idSklepu INTEGER REFERENCES Sklep(idS) NOT NULL,

czasZ TIME NOT NULL,

dataZamowienia DATE NOT NULL,

dataOplacenia DATE,

dataOdbioru DATE,

calkowitaSuma DECIMAL(30,4) NOT NULL

);

CREATE TABLE Nabywa (

idN INTEGER PRIMARY KEY,

idZakupu INTEGER REFERENCES RobiZakupy(nrTransakcji) NOT NULL,

idProd INTEGER REFERENCES Produkt(idP) NOT NULL,

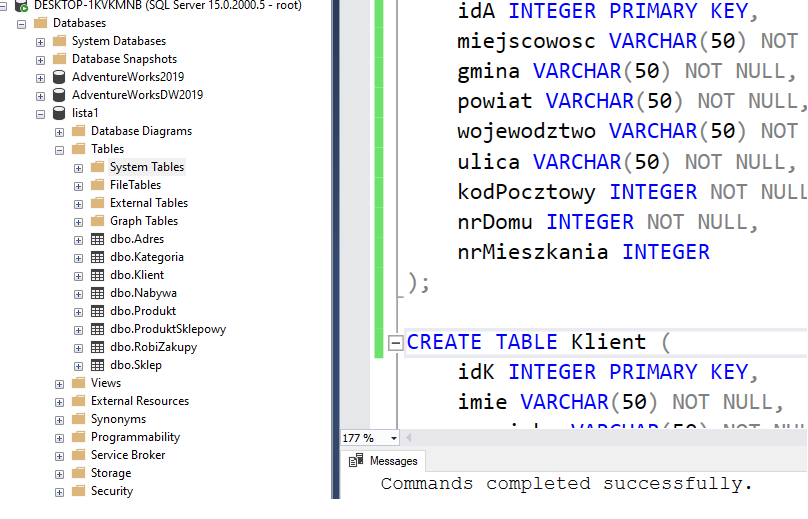
iloscN INTEGER NOT NULL,

cenaN DECIMAL(30,4) NOT NULL

);

**Zad 1.4**

Na podstawie powyższych zapytań pomyślnie utworzyłem bazę danych w systemie MS SQL 2019:

****

**Zad 1.5**

**Przykładowe wprowadzanie rekordów do bazy danych:**

INSERT INTO Adres VALUES(1, 'Wrocław', 'Wrocław', 'wrocławski', 'dolnośląskie', 'Kardynała Stefana Wyszyńskiego', 50332, 19, NULL);

INSERT INTO Kategoria VALUES('zdrowie');

INSERT INTO Kategoria VALUES('elektronika');

INSERT INTO Kategoria VALUES('meble');

INSERT INTO Kategoria VALUES('zabawki');

INSERT INTO Klient VALUES(1, 'Jan', 'Brzechwa', NULL, 23, 'jbrze@onet.pl', 'F$AFJGA$GAG!j32i4b', NULL);

INSERT INTO Sklep VALUES(1, 'H&M', 1);

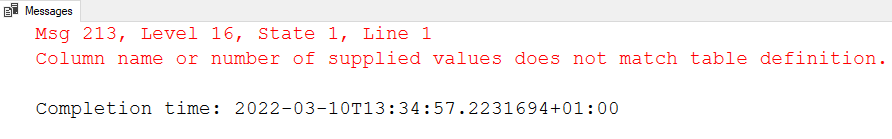
INSERT INTO Produkt VALUES(1, 'lalka Barbie', 'zabawki', 19.95, 50, 2017);

INSERT INTO ProduktSklepowy VALUES(1, 1, 1);

INSERT INTO RobiZakupy VALUES(1, 1, 1, '19:43', '2022-01-20', NULL, NULL, 79.8);

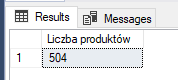
INSERT INTO Nabywa VALUES(1, 1, 1, 4, 19.95);

**Po wprowadzeniu błędnego rekordu (tutaj ‘wojewodztwo’ zostało podane jako NULL przy wprowadzaniu adresu):**



**Zad 2.1**

SELECT COUNT(\*) 'Liczba produktów' FROM Production.Product;



SELECT COUNT(\*) 'Liczba kategorii' FROM Production.ProductCategory;



SELECT COUNT(\*) 'Liczba podkategorii' FROM Production.ProductSubcategory;



**Zad 2.2**

SELECT Name 'Produkt' FROM Production.Product WHERE Color IS NULL;

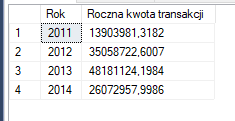


**Wniosek:** Prawie połowa produktów nie posiada określonego koloru.

**Zad 2.3**

SELECT Year(DueDate) 'Rok' , SUM(TotalDue) 'Roczna kwota transakcji'

FROM Sales.SalesOrderHeader GROUP BY Year(DueDate) ORDER BY 1;



**Wniosek:** Z każdym kolejnym rokiem sumaryczna kwota z transakcji wydaj się rosnąć (dla roku 2014 nie mamy transakcji ze wszystkich miesięcy, tylko do połowy roku -> w związku z tym można założyć, że jeszcze drugie tyle do tej sumy należałoby dodać)

**Zad 2.4**

SELECT COUNT(\*) 'Liczba klientów' FROM Sales.Customer;



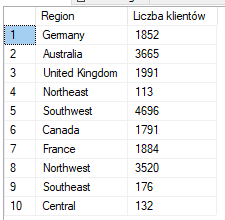
SELECT COUNT(\*) 'Liczba sprzedawców' FROM Sales.SalesPerson;



SELECT T.Name 'Region', COUNT(C.CustomerID) 'Liczba klientów'

FROM Sales.Customer C LEFT JOIN Sales.SalesTerritory T ON T.TerritoryID=C.TerritoryID

GROUP BY T.Name;

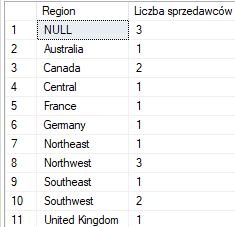


**Wniosek:** Najwięcej klientów pochodzi ze Stanów Zjednoczonych (Northwest + Southeast + Central + Northeast + Southwest), a następnie z Australii. Z Francji, Niemiec, UK i Kanady mamy do czynienia z podobnymi liczbami.

SELECT T.Name 'Region', COUNT(SP.BusinessEntityID) 'Liczba sprzedawców'

FROM Sales.SalesPerson SP LEFT JOIN Sales.SalesTerritory T ON T.TerritoryID=SP.TerritoryID

GROUP BY T.Name;



**Wniosek:** Trzech sprzedawców nie ma w ogóle przypisanego regionu. Ogólnie w USA pracuje najwięcej sprzedawców (co koreluje z liczbą klientów), dla pozostałych regionów są to pojedyncze osoby.

**Zad 2.5**

SELECT YEAR(DueDate) 'Rok', COUNT(SalesOrderID) 'Liczba transakcji' FROM Sales.SalesOrderHeader

GROUP BY YEAR(DueDate) ORDER BY 1;



**Wniosek:** Z każdym kolejnym rokiem zwiększa się liczba transakcji (uwzględniając wcześniej wspomnianą niepełność danych dla 2014 roku).

**Zad 2.6**

SELECT PC.Name 'Kategoria', PSC.Name 'Podkategoria', COUNT(\*) 'Liczba produktów' FROM

(SELECT ProductID FROM Production.Product

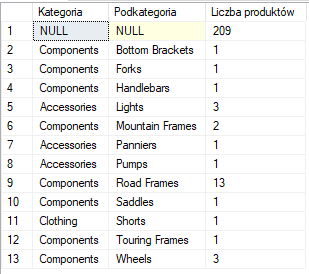
EXCEPT SELECT ProductID FROM Sales.SalesOrderDetail) PID

LEFT JOIN Production.Product P ON P.ProductID=PID.ProductID

LEFT JOIN Production.ProductSubcategory PSC ON PSC.ProductSubcategoryID=P.ProductSubcategoryID

LEFT JOIN Production.ProductCategory PC ON PC.ProductCategoryID=PSC.ProductCategoryID

GROUP BY PC.Name, PSC.Name;



**Wniosek:** Zdecydowana większość produktów, które nie zostały zakupione przez żadnego klienta, nie posiada ani kategorii ani podkategorii. Najwięcej niezakupionych produktów należy do kategorii ‘Components’ (części rowerowych) oraz ‘Accessories’ (czyli wszelakich akcesoriów). Ramy rowerowe zawierają najwięcej niezakupionych produktów ze wszystkich podkategorii.

**Zad 2.7**

SELECT PSC.Name 'Podkategoria',

MIN(SOD.UnitPriceDiscount \* SOD.OrderQty \* SOD.UnitPrice) 'Najmniejsza obnizka',

MAX(SOD.UnitPriceDiscount \* SOD.OrderQty \* SOD.UnitPrice) 'Najwieksza obnizka'

FROM Production.Product P

LEFT JOIN Production.ProductSubcategory PSC ON PSC.ProductSubcategoryID=P.ProductSubcategoryID

LEFT JOIN Sales.SalesOrderDetail SOD ON SOD.ProductID=P.ProductID

WHERE SOD.UnitPriceDiscount > 0

GROUP BY PSC.Name;

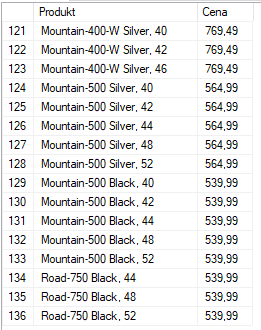


**Wniosek:** Największe wartości w maksymalnej obniżce dotyczą ram i rowerów górskich natomiast najmniejsze dla ubrań i akcesoriów. Wynika z tego, że im droższy dany produkt tym większe wartości obniżki (**w złotówkach a nie procentowo!)**.

**Zad 2.8**

SELECT Name 'Produkt', ListPrice 'Cena' FROM Production.Product

WHERE ListPrice > (SELECT AVG(ListPrice) FROM Production.Product);



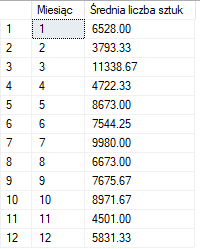
**Wniosek:** Produkty droższe od średniej to ramy i całe rowery.

**Zad 2.9**

SELECT Month(SOH.OrderDate) 'Miesiąc', CAST(CAST(SUM(SOD.OrderQty) AS FLOAT) / COUNT(DISTINCT YEAR(OrderDate)) AS DECIMAL(10,2)) 'Średnia liczba sztuk'

FROM Sales.SalesOrderHeader SOH JOIN Sales.SalesOrderDetail SOD ON SOD.SalesOrderID=SOH.SalesOrderID

GROUP BY Month(SOH.OrderDate) ORDER BY 1;



**Wniosek:** Najmniej sprzedawanych sztuk produktów jest w lutym a najwięcej w marcu. Ciężko doszukać się jakichś trendów na podstawie kolejnych miesięcy czy kwartałów.

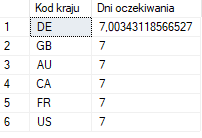
**Zad 2.10**

SELECT ST.CountryRegionCode 'Kod kraju', AVG(CAST(DATEDIFF(DAY, OrderDate, ShipDate) AS FLOAT)) 'Dni oczekiwania'

FROM Sales.SalesOrderHeader SOH

LEFT JOIN Sales.SalesTerritory ST ON ST.TerritoryID=SOH.TerritoryID

GROUP BY ST.CountryRegionCode;



**Wniosek:** W zasadzie w każdym kraju niezależnie od kontynentu średnio klient oczekuje tydzień na wysyłkę.

**Zadanie z zajęć:**

SELECT ST.CountryRegionCode 'Region', DateDiff(YEAR, E.BirthDate, SYSDATETIME()) 'Wiek',

SUM(LineTotal) 'Suma sprzedana' FROM Sales.SalesOrderDetail SOD

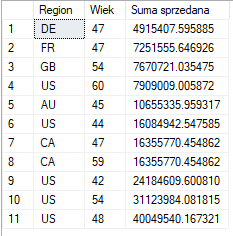
JOIN Sales.SalesOrderHeader SOH ON SOD.SalesOrderID=SOH.SalesOrderID

JOIN Sales.SalesTerritory ST ON ST.TerritoryID=SOH.TerritoryID

JOIN Sales.SalesPerson SP ON SP.TerritoryID=ST.TerritoryID

JOIN HumanResources.Employee E ON E.BusinessEntityID=SP.BusinessEntityID

GROUP BY ST.CountryRegionCode, DateDiff(YEAR, E.BirthDate, SYSDATETIME()) ORDER BY 3;



**Wniosek:** Pracownicy z Ameryki Północnej sprzedają więcej niż pracownicy z innych części świata. Wiek wydaje się nie mieć w tym przypadku znaczenia.

**Wnioski:**

* Zestaw reguł i ograniczeń biznesowych jest bardzo pomocny przy tworzeniu modelu danych (np. diagramu klas UML)
* Sam diagram klas w sposób wizualny (za pomocą asocjacji i atrybutów) pozwala intuicyjnie zwizualizować przyszłą bazę danych (klucze główne/obce i relacje same w sobie). Należy na tym etapie zachować precyzję i dobrze przemyśleć wszystkie funkcjonalności gdyż przełoży się to potem na funkcjonowanie całego systemu/całej bazy.
* Utworzenie fizycznej bazy danych na podstawie diagramu UML’owego nie stanowi już większego problemu (odpowiednie zapytania DDL SQL pozwalają dokładnie odwzorować wszelkie powiązania i ograniczenia). SZBD będzie dokładnie przestrzegać wszystkich wyznaczonych przez nas reguł.
* Za pomocą SQL’a możemy „wyciągnąć” proste informacje z naszej bazy danych (o liczbie rekordów czy o konkretnych rekordach). Dodatkowo możemy tworzyć kwerendy, które pozwolą nam zaobserwować korelacje między pewnymi danymi naszej bazy. Należy w tym wypadku być ostrożnym, musimy być świadomi struktury naszej bazy, tego co chcemy tak naprawdę uzyskać oraz jak należy interpretować otrzymane wyniki.