Filip Strózik 260377

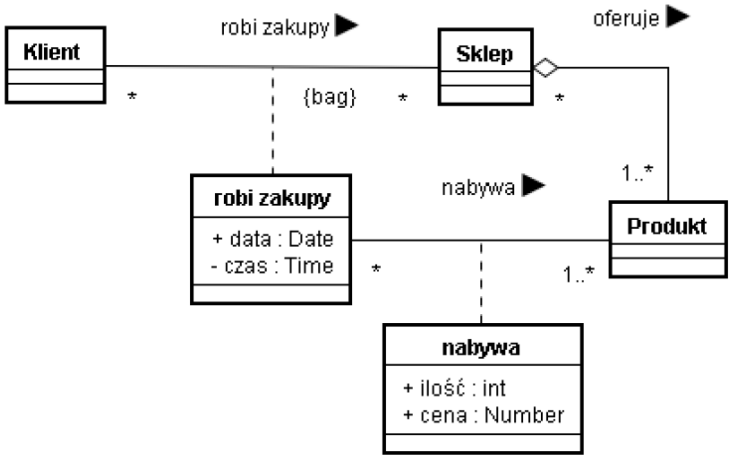
Hurtownie Danych - laboratorium
Lista 1

Modelowanie danych i podstawy SQL

Zad 1. Modelowanie danych
Proszę przeanalizować konceptualny model danych „Usługi" (Rys. 1), który jest niekompletny, ale klasy i relacje między nimi reprezentują rozpatrywany wycinek rzeczywistości. Następnie proszę wykonać następujące zadania:

1. Zweryfikować model danych w kontekście podanego zbioru reguł i ograniczeń dziedzinowych modyfikując zbiór reguł i ograniczeń (uzupełniając lub poprawiając ich definicję)
2. Przedstawić uzupełnioną i poprawioną wersję modelu danych (kompletny diagram klas UML)
3. Utworzyć logiczny model danych w postaci skryptu w języku DDL SQL (uwzględniając reguły i ograniczenia dziedzinowe), starając się zachować zgodność ze standardem języka SQL (pomijając, o ile to możliwe, natywne konstrukcje implementacji języków SQL)
4. Utworzyć bazę danych w systemie MS SQL 2019, która jest fizycznym modelem danych modelowanego wycinka rzeczywistości
5. Wprowadzić klika rekordów do każdej tabeli sprawdzając poprawność implementacji (zarówno poprawne dane, jak i niezgodne z obowiązującymi regułami – komentując i wyjaśniając uzyskane komunikaty z systemu SZBD)

Reguły i ograniczenia dziedzinowe:
Reg/01 – klient może wielokrotnie robić zakupy w tym samym sklepie
Reg.02 – W sklepie może robić zakupy dowolny klient
Reg.03 – Każdy zakup realizowany jest przez klienta w sklepie w określonym dniu i godzinie
Reg/04 – Sklep musi oferować co najmniej jeden produkt



Rysunek 1. Konceptualny model danych "Usługi"

Rozwiązanie:

1. Zweryfikować model danych w kontekście podanego zbioru reguł i ograniczeń dziedzinowych modyfikując zbiór reguł i ograniczeń (uzupełniając lub poprawiając ich definicję)

Regułę pierwszą:
Reg/01 – klient może wielokrotnie robić zakupy w tym samym sklepie.

Można zamienić na dwie reguły, które bardziej określają krotność między encjami.

Reg/01 – klient może robić zakupy w sklepie.
Reg/02 – kleint może robić zakupy w wielu sklepach.

Regułę drugą:
Reg/02 – W sklepie może robić zakupy dowolny klient.

Można zamienić na dwie reguły, które bardziej określają krotność między encjami.

Req/03 – W sklepie może robić zakupy klient.
Reg/04 - W sklepie może robić zakupy wiele klientów.

Regułę czwartą:
Reg/04 – Sklep musi oferować co najmniej jeden produkt.

Można zamienić na cztery reguły, które bardziej określają krotność między encjami Sklep oraz Produkt.

Reg/05 – Sklep musi oferować produkt.
Reg/06 – Sklep musi oferować co najmniej jeden produkt.

Reg/07 – Produkt może być oferowany w sklepie
Reg/08 – Produkt może być oferowany w wielu sklepach.

Z dodatkowych ograniczeń możemy dopisać:

Reg/09 – Cena nabycia musi być większa od zera.
Req/010 – Ilość nabycia nie może być ujemna.

2. Przedstawić uzupełnioną i poprawioną wersję modelu danych (kompletny diagram klas UML)

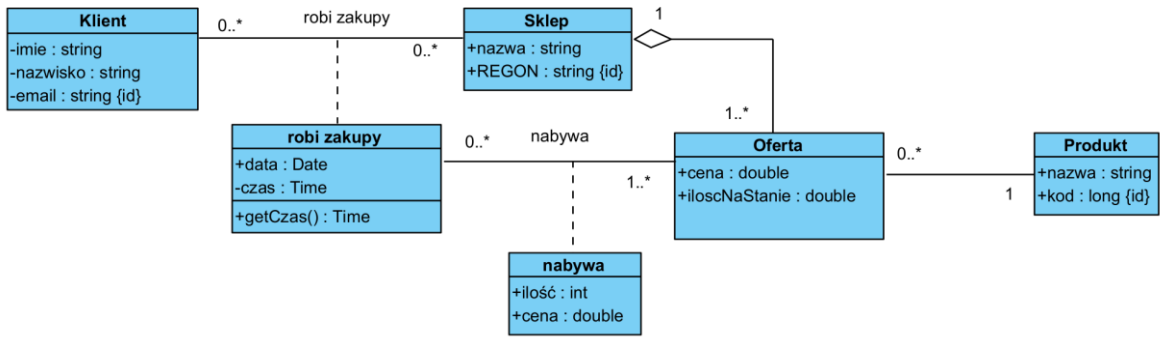
Zdecydowałem się na dodanie klasy Oferta, dzięki której dla konkretnego sklepu możemy ustalić różną dla tego samego produktu w innych sklepach. Dlatego to w tej klasie jest podana cena, a nie w encji Produkt. Dodatkowo przydajnym może być informacja o ilości produktu na stanie w danym sklepie.

Sklep otrzymał jako identyfikacyjny numer REGON oraz nazwę.

Klient otrzymał dodatkowy indentyfikacyjny adres email w celu wytworzenia naturalnego klucza głównego dla klienta. Zakładamy dla tego rozwiązania, że do dokonania zakupu klient musi podać swój tylko swój adres email. Klasa posiada też imię oraz nazwisko klienta.

Do klasy asocjacyjnej „robi zakupy“ dodałem getter dla atrybutu czas, którego modyfikator dostępu jest prywatny.

Klasa produkt, otrzymała indetyfikacyjny kod produktu oraz nazwę produktu.



Teraz po tych zmianach możemy dodać kolejne ograniczenia dziedzinowe:

Reg/011 – ilośćNaStanie nie może być ujemna.
Req/012 – cena Oferty musi być wieksza od zera.

3. Utworzyć logiczny model danych w postaci skryptu w języku DDL SQL (uwzględniając reguły i ograniczenia dziedzinowe), starając się zachować zgodność ze standardem języka SQL (pomijając, o ile to możliwe, natywne konstrukcje implementacji języków SQL)

```
CREATE DATABASE Uslugi;

CREATE TABLE Klienci (
        Imie VARCHAR(255),
        Nazwisko VARCHAR(255),
        Email VARCHAR(255),
        PRIMARY KEY (Email)
);

CREATE TABLE Sklepy (
        Nazwa VARCHAR(255),
```

```
	REGON VARCHAR(255),
	PRIMARY KEY (REGON)
);

CREATE TABLE Oferty (
	IdOferty INT NOT NULL,
	Cena DOUBLE NOT NULL,
	IloscNaStanie INT NOT NULL,
	KodProduktu LONG NOT NULL,
	IdSklepu VARCHAR(255) NOT NULL,
	PRIMARY KEY (IdOferty),
	CONSTRAINT Oferty_Cena_Con CHECK (Cena>0),
	CONSTRAINT Oferty_Ilosc_Con CHECK (IloscNaStanie>=0)
);

CREATE TABLE Produkty (
	KodProduktu LONG NOT NULL,
	Nazwa VARCHAR(255),
	PRIMARY KEY (KodProduktu)
)

CREATE TABLE Nabycia (
	IdNabycia INT NOT NULL,
	Ilosc INT NOT NULL,
	Cena DOUBLE NOT NULL,
	IdOferty INT NOT NULL,
	IdZakupu INT NOT NULL,
	PRIMARY KEY (IdNabycia),
	CONSTRAINT Nabycia_Ilosc_Con CHECK (Ilosc>=0),
	CONSTRAINT Nabycia_Cena_Con CHECK (Cena>0)
);

CREATE TABLE Zakupy (
	IdZakupu INT NOT NULL,
	DataZakupu DATE NOT NULL,
	CzasZakupu TIME NOT NULL,
	IdKlienta VARCHAR(255) NOT NULL,
	IdSklepu VARCHAR(255) NOT NULL,
	PRIMARY KEY (IdZakupu)
);

ALTER TABLE Oferty
ADD FOREIGN KEY (IdSklepu) REFERENCES Sklepy(REGON);

ALTER TABLE Oferty
ADD FOREIGN KEY (KodProduktu) REFERENCES Produkty(KodProduktu);

ALTER TABLE Nabycia
ADD FOREIGN KEY (IdOferty) REFERENCES Oferty(IdOferty);
```

```
ALTER TABLE Nabycia
ADD FOREIGN KEY (IdZakupu) REFERENCES Zakupy(IdZakupu);

ALTER TABLE Zakupy
ADD FOREIGN KEY (IdSklepu) REFERENCES Sklepy(REGON);

ALTER TABLE Zakupy
ADD FOREIGN KEY (IdKlienta) REFERENCES Klienci(Email);
```

4. Utworzyć bazę danych w systemie MS SQL 2019, która jest fizycznym modelem danych modelowanego wycinka rzeczywistości

Aby utworzyć taką bazę danych pierwszym co uniemożliwia jej stworzenie to niekompatybilne typy LONG oraz DOUBLE, naprawa polegała na zamianie typów na odpowiednio: BIGINT oraz FLOAT.

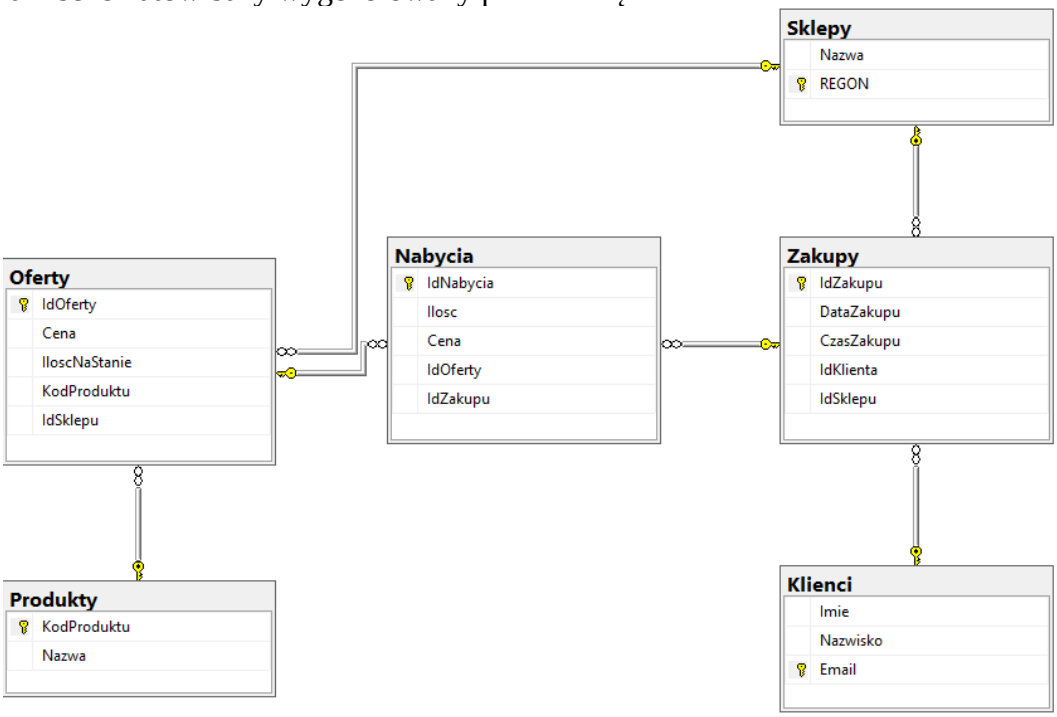
Poprawiony skrypt DDL dla MSSQL:

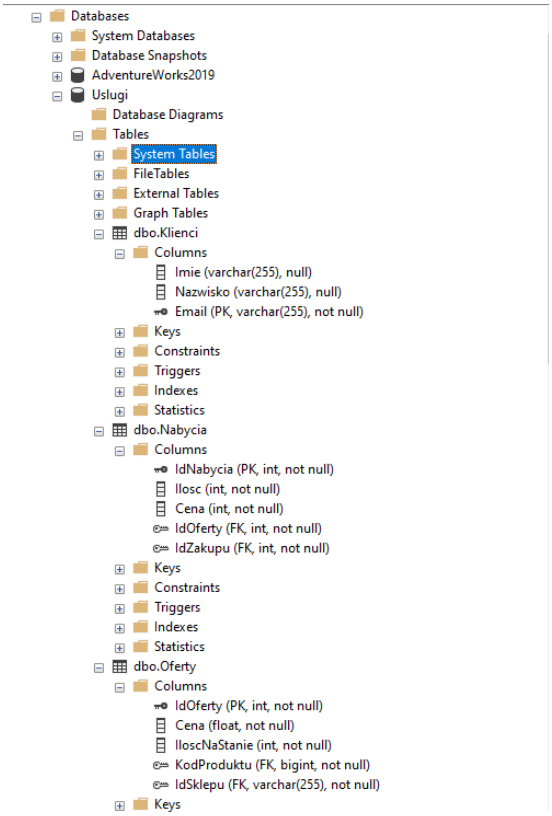
```
CREATE DATABASE Uslugi;


CREATE TABLE Klienci (
	Imie VARCHAR(255),
	Nazwisko VARCHAR(255),
	Email VARCHAR(255),
	PRIMARY KEY (Email)
);

CREATE TABLE Sklepy (
	Nazwa VARCHAR(255),
	REGON VARCHAR(255),
	PRIMARY KEY (REGON)
);

CREATE TABLE Oferty (
	IdOferty INT NOT NULL,
	Cena FLOAT NOT NULL,
	IloscNaStanie INT NOT NULL,
	KodProduktu BIGINT NOT NULL,
	IdSklepu VARCHAR(255) NOT NULL,
	PRIMARY KEY (IdOferty),
	CONSTRAINT Oferty_Cena_Con CHECK (Cena>0),
	CONSTRAINT Oferty_Ilosc_Con CHECK (IloscNaStanie>=0)
);

CREATE TABLE Produkty (
	KodProduktu BIGINT NOT NULL,
	Nazwa VARCHAR(255),
	PRIMARY KEY (KodProduktu)
```

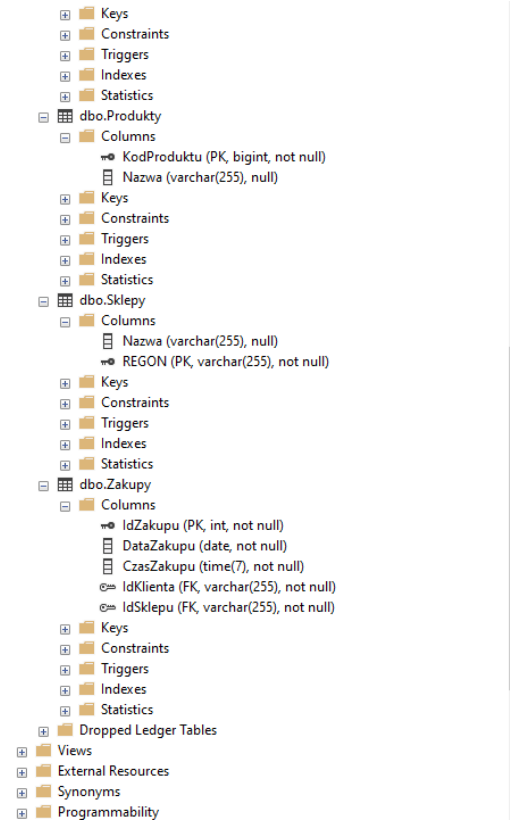
```
);

CREATE TABLE Nabycia (
        IdNabycia INT NOT NULL,
        Ilosc INT NOT NULL,
        Cena DOUBLE NOT NULL,
        IdOferty INT NOT NULL,
        IdZakupu INT NOT NULL,
        PRIMARY KEY (IdNabycia),
        CONSTRAINT Nabycia_Ilosc_Con CHECK (Ilosc>=0),
        CONSTRAINT Nabycia_Cena_Con CHECK (Cena>0)
);

CREATE TABLE Zakupy (
        IdZakupu INT NOT NULL,
        DataZakupu DATE NOT NULL,
        CzasZakupu TIME NOT NULL,
        IdKlienta VARCHAR(255) NOT NULL,
        IdSklepu VARCHAR(255) NOT NULL,
        PRIMARY KEY (IdZakupu)
);

ALTER TABLE Oferty
ADD FOREIGN KEY (IdSklepu) REFERENCES Sklepy(REGON);

ALTER TABLE Oferty
ADD FOREIGN KEY (KodProduktu) REFERENCES Produkty(KodProduktu);

ALTER TABLE Nabycia
ADD FOREIGN KEY (IdOferty) REFERENCES Oferty(IdOferty);

ALTER TABLE Nabycia
ADD FOREIGN KEY (IdZakupu) REFERENCES Zakupy(IdZakupu);

ALTER TABLE Zakupy
ADD FOREIGN KEY (IdSklepu) REFERENCES Sklepy(REGON);

ALTER TABLE Zakupy
ADD FOREIGN KEY (IdKlienta) REFERENCES Klienci(Email);
```

Filip Strózik 260377

Diagram schematów bazy wygenerowany przez narzędzie SSMS:







5. Wprowadzić klika rekordów do każdej tabeli sprawdzając poprawność implementacji (zarówno poprawne dane, jak i niezgodne z obowiązującymi regułami – komentując i wyjaśniając uzyskane komunikaty z systemu SZBD)

--dodawanie poprawnych danych
--aby dodać dane potrzeba chwilowo wyłączyć ograniczenia związane z tabelami.
ALTER TABLE Nabycia NOCHECK CONSTRAINT all;
ALTER TABLE Zakupy NOCHECK CONSTRAINT all;

-- DANE DLA TABELI KLIENCI
INSERT INTO Klienci (Imie, Nazwisko, Email)
VALUES
('Anna', 'Kowalska', 'anna.kowalska@example.com'),
('Krzysztof', 'Nowak', 'k.nowak@example.com'),
('Jan', 'Nowicki', 'jan.nowicki@example.com'),
('Maria', 'Malinowska', 'maria.malinowska@example.com'),
('Piotr', 'Wójcik', 'piotr.wojcik@example.com');

Imie	Nazwisko	Email
Anna	Kowalska	anna.kowalska@example.com
Jan	Nowicki	jan.nowicki@example.com
Krzysztof	Nowak	k.nowak@example.com
Maria	Malinowska	maria.malinowska@example.com
Piotr	Wójcik	piotr.wojcik@example.com

-- DANE DLA TABELI SKLEPY
INSERT INTO Sklepy (Nazwa, REGON)
VALUES
('Społem', '123456789'),
('Biedronka', '987654321'),
('Carrefour', '456789123'),
('Lidl', '321654987'),
('Auchan', '789123456');

Nazwa	REGON
Spolem	123456789
Lidl	321654987
Carrefour	456789123
Auchan	789123456
Biedronka	987654321

-- DANE DLA TABELI PRODUKTY
INSERT INTO Produkty (KodProduktu, Nazwa)
VALUES
(1234567890123, 'Chleb'),
(2345678901234, 'Mleko'),
(3456789012345, 'Ser'),
(4567890123456, 'Jogurt'),

(5678901234567, 'Jajka');

KodProduktu	Nazwa
1234567890123	Chleb
2345678901234	Mleko
3456789012345	Ser
4567890123456	Jogurt
5678901234567	Jajka

-- DANE DLA TABELI OFERTY
INSERT INTO Oferty (IdOferty, Cena, IloscNaStanie, KodProduktu, IdSklepu)
VALUES
(1, 2.99, 100, 1234567890123, '123456789'),
(2, 4.99, 50, 2345678901234, '987654321'),
(3, 3.49, 75, 3456789012345, '456789123'),
(4, 1.99, 120, 4567890123456, '321654987'),
(5, 5.99, 30, 5678901234567, '789123456');

IdOferty	Cena	IloscNaStanie	KodProduktu	IdSklepu
1	2.99	100	1234567890123	123456789
2	4.99	50	2345678901234	987654321
3	3.49	75	3456789012345	456789123
4	1.99	120	4567890123456	321654987
5	5.99	30	5678901234567	789123456

-- DANE DLA TABELI ZAKUPY
INSERT INTO Zakupy (IdZakupu, DataZakupu, CzasZakupu, IdKlienta, IdSklepu)
VALUES
(1, '2022-12-12', '10:30:00', 'anna.kowalska@example.com', '123456789'),
(2, '2022-12-13', '12:15:00', 'k.nowak@example.com', '456789123'),
(3, '2022-12-14', '15:00:00', 'jan.nowicki@example.com', '789123456'),
(4, '2022-12-15', '11:45:00', 'maria.malinowska@example.com', '987654321');

IdZakupu	DataZakupu	CzasZakupu	IdKlienta	IdSklepu
1	2022-12-12	10:30:00.0000000	anna.kowalska@example.com	123456789
2	2022-12-13	12:15:00.0000000	k.nowak@example.com	456789123
3	2022-12-14	15:00:00.0000000	jan.nowicki@example.com	789123456
4	2022-12-15	11:45:00.0000000	maria.malinowska@example.com	987654321

-- DANE DLA TABELI NABYCIA
INSERT INTO Nabycia (IdNabycia, Ilosc, Cena, IdOferty, IdZakupu)
VALUES
(1, 2, 55.55, 1, 1),
(2, 3, 7.34, 2, 1),
(3, 1, 4.66, 3, 2),
(4, 5, 66.3, 4, 2),

(5, 2, 6.4, 5, 3);

IdNabycia	Ilosc	Cena	IdOferty	IdZakupu
1	2	55.55	1	1
2	3	7.34	2	1
3	1	4.66	3	2
4	5	66.3	4	2
5	2	6.4	5	3

ALTER TABLE Nabycia WITH CHECK CHECK CONSTRAINT all;
ALTER TABLE Zakupy WITH CHECK CHECK CONSTRAINT all;

--sprawdzenie ograniczeń:
--NOT NULL
INSERT INTO Zakupy
VALUES
(5, '2023-12-15', NULL , 'maria.malinowska@example.com', '987654321');

Msg 515, Level 16, State 2, Line 151
Cannot insert the value NULL into column 'CzasZakupu', table 'Uslugi.dbo.Zakupy'; column does not allow nulls. INSERT fails.
The statement has been terminated.

Ograniczenie NOT NULL nie pozwala na wpisywanie wartości NULL dla atrybutu.

INSERT INTO Oferty
VALUES (6, -0.11, 1, 5678901234567, '321654987');

Msg 547, Level 16, State 0, Line 155
The INSERT statement conflicted with the CHECK constraint "Oferty_Cena_Con". The conflict occurred in database "Uslugi", table "dbo.Oferty", column 'Cena'.
The statement has been terminated.

INSERT INTO Oferty
VALUES (6, 0.1, -1, 5678901234567, '321654987');

Msg 547, Level 16, State 0, Line 158
The INSERT statement conflicted with the CHECK constraint "Oferty_Ilosc_Con". The conflict occurred in database "Uslugi", table "dbo.Oferty", column 'IloscNaStanie'.
The statement has been terminated.

INSERT INTO Produkty (KodProduktu, Nazwa)
VALUES
(1234567890123, 'Chleb');

Msg 2627, Level 14, State 1, Line 162
Violation of PRIMARY KEY constraint 'PK__Produkty__0863E9DF171004A7'. Cannot insert duplicate key in object 'dbo.Produkty'. The duplicate key value is (1234567890123).
The statement has been terminated.

INSERT INTO Zakupy VALUES
(1, '2022-12-12', '10:30:00', 'anna.kowalska@example.com');

Msg 213, Level 16, State 1, Line 167
Column name or number of supplied values does not match table definition.

Baza danych: **AdventureWorks**

Zad. 2. Podstawy SQL
Proszę zapisać zapytania SQL, które dadzą odpowiedź na poniższe pytania. Proszę zinterpretować wyniki.

Rozwiązania:

1. Ile jest produktów w bazie? Ile kategorii i podkategorii?

SQL:

```
SELECT COUNT(Production.Product.ProductID) FROM Production.Product;
```

WYNIK:

504

INTERPRETACJA:

Aby dostać ilość produktów w bazie, najprościej odwołać się do tabeli produkt zliczyć krotki wybierając klucz główny.

SQL:

```
SELECT COUNT(Production.ProductCategory.ProductCategoryID) FROM Production.ProductCategory;
```

WYNIK:

4

INTERPRETACJA:

Aby dostać ilość kategorii najprościej jest się odwołać do tabeli produkt kategorii analogicznie zliczyć ilość.

SQL:

```
SELECT COUNT(Production.ProductSubcategory.ProductSubcategoryID) FROM Production.ProductSubcategory;
```

WYNIK:

37

INTERPRETACJA:

Mamy 37 podkategorii w bazie danych wybór struktury zapytania sql jest analogiczny do powyższych.

2. Wypisz produkty, które nie mają zdefiniowanego koloru.

SQL:

SELECT * FROM Production.Product WHERE Production.Product.Color IS NULL;

WYNIK:

ProductID	Name
1	Adjustable Race
2	Bearing Ball
3	BB Ball Bearing
4	Headset Ball Bearings
316	Blade
323	Crown Race

... w sumie 248 wierszy.

INTERPRETACJA:

Aby rozróżniać produkty zdecydowano się wypisać jego ID oraz nazwę, uzyskujemy to stawiając warunek który sprawia że wypisz wypisujemy tylko produkty dla których wartość koloru jest NULL.

3. Podaj roczną kwotę transakcji (SalesOrderHeader.TotalDue) w poszczególnych latach.

SQL:

SELECT SUM(TotalDue) AS Kwota, YEAR(DueDate) AS Rok FROM Sales.SalesOrderHeader GROUP BY YEAR(DueDate) ORDER BY YEAR(DueDate);

WYNIK:

Kwota	Rok
13903981.3182	2011
35058722.6007	2012
48181124.1984	2013
26072957.9986	2014

INTERPRETACJA:

Aby obliczyć sumę wszystkich transakcji w poszczególnych latach stosujemy funkcję agregacji SUM(). Aczkolwiek funkcja agregacja i potrzebuje mieć wartości zgrupowane dla-tego grupujemy według roku transakcji.

4. Ilu jest klientów, a ilu sprzedawców w sklepie? Ilu w poszczególnych regionach?

SQL:

SELECT COUNT(CustomerID) FROM Sales.Customer;

WYNIK:

19820

INTERPRETACJA:

Aby dostać ilość wszystkich klientów w sklepie wybieramy tabelę customer oraz do zliczania wybieramy klucz główny który zapewni nam optymalna wykonywania zapytania.

SQL:

SELECT COUNT(BusinessEntityID) FROM Sales.SalesPerson;

WYNIK:

17

INTERPRETACJA:

Analogicznie co do powyższego, tylko wybieramy sprzedawca z tabeli Sales.SalesPerson.

SQL:

```
SELECT COUNT(DISTINCT CustomerID) AS "Liczba Klientow",
MIN(SalesTerritory.Name) AS Region, COUNT(DISTINCT SalesPerson.BusinessEntityID)
AS "Liczba Sprzedawcow"
FROM Sales.Customer JOIN Sales.SalesTerritory
ON Customer.TerritoryID = SalesTerritory.TerritoryID
JOIN Sales.SalesPerson ON SalesTerritory.TerritoryID = SalesPerson.TerritoryID
GROUP BY SalesTerritory.TerritoryID;
```

WYNIK:

Liczba Klientow	Region	Liczba Sprzedawcow
132	Central	1
1791	Canada	2
3665	Australia	1
3520	Northwest	3
1884	France	1
1991	United Kingdom	1
4696	Southwest	2
176	Southeast	1
113	Northeast	1
1852	Germany	1

INTERPRETACJA:

Aby dostać zestawienie o liczbie klientów oraz o liczbie sprzedawców w poszczególnych regionach musimy połączyć tabelę klientów oraz tabelę sprzedawców za pomocą złączenia JOIN z tabelą Sales.SalesTerritory i pogrupować po TerritoryID.

5. Ile było wykonanych transakcji w poszczególnych latach?

SQL:

```
SELECT YEAR(SalesOrderHeader.OrderDate) AS "Rok Transakcji",
COUNT(SalesOrderID) AS "Liczba Transakcji"
FROM Sales.SalesOrderHeader
GROUP BY YEAR(SalesOrderHeader.OrderDate)
ORDER BY YEAR(SalesOrderHeader.OrderDate) ASC;
```

WYNIK:

Rok Transakcji	Liczba Transakcji
2011	1607
2012	3915
2013	14182
2014	11761

INTERPRETACJA:

Aby uzyskać liczbę wykonanych transakcji w poszczególnych latach musimy zliczyć wszystkie transakcje w grupach gdzie każda grupa to jest poszczególny, to uzyskujemy grupując po roku z daty dokonania zamówienia.

6. Podaj produkty, które nie zostały kupione przez żadnego klienta. Zestawienie pogrupuj według kategorii i podkategorii.

SQL:

```
SELECT
   p.Name AS "Nazwa Produktu",
   pc.Name AS "Nazwa Kategorii",
   psc.Name AS "Nazwa Podkategorii"
FROM
   Production.Product AS p
   JOIN Production.ProductSubcategory AS psc
      ON p.ProductSubcategoryID = psc.ProductSubcategoryID
   JOIN Production.ProductCategory AS pc
      ON psc.ProductCategoryID = pc.ProductCategoryID
WHERE
   p.ProductID NOT IN
   (SELECT DISTINCT sod.ProductID
    FROM Sales.SalesOrderDetail AS sod)
ORDER BY
   pc.Name ASC,
   psc.Name ASC;
```

WYNIK:

Nazwa Produktu	Nazwa Kategorii	Nazwa Podkategorii
Taillights - Battery-Powered	Accessories	Lights
Headlights - Dual-Beam	Accessories	Lights
Headlights - Weatherproof	Accessories	Lights
Touring-Panniers, Large	Accessories	Panniers
Mountain Pump	Accessories	Pumps
Men's Sports Shorts, XL	Clothing	Shorts
ML Bottom Bracket	Components	Bottom Brackets
ML Fork	Components	Forks

...

INTERPRETACJA:

Aby podać produkty które nie zostały zakupione przez żadnego klienta i pogrupować je zgodnie z kategoriami oraz podkategoriami należy wypisać wszystkie produkty wraz z ich kategoriami i podkategoriami stosując złączenia tabel które nie występują w tabeli order detale w których są zapisy zamówionych produktów. Ta wersja sequela jest lżejsza niż stosowanie różnicy wyników o tych samych schematach relacji to jest w przypadku użycia funkcji EXCEPT.

7. Oblicz minimalną i maksymalną kwotę rabatu udzielonego na produkty w poszczególnych podkategoriach.

SQL:

```
SELECT
ps.Name AS Podkategoria,
MIN(UnitPrice * UnitPriceDiscount) AS "Minimalna Kwota Rabatu",
MAX(UnitPrice * UnitPriceDiscount) AS  "Maksymalna Kwota Rabatu"
FROM
   Sales.SalesOrderDetail sod
   JOIN Sales.SpecialOffer so ON sod.SpecialOfferID = so.SpecialOfferID
        JOIN Production.Product p ON sod.ProductID = p.ProductID
   JOIN Production.ProductSubcategory ps ON p.ProductSubcategoryID =
ps.ProductSubcategoryID
WHERE
        UnitPriceDiscount != 0.00
GROUP BY
   ps.Name
ORDER BY
   ps.Name;
```

WYNIK:

Podkategoria	Minimalna Kwota Rabatu	Maksymalna Kwota Rabatu
Bib-Shorts	1.0439	2.4747
Bike Racks	1.392	6.00
Bottles and Cages	0.0579	0.2495
Brakes	1.2354	1.2354
Caps	0.1003	0.4495

...

INTERPRETACJA:

Aby obliczyć minimalną i maksymalną kwotę rabatu dla produktów w każdej podkategorii należało, wykorzystać tabele Sales.SalesOrderDetail, Sales.SpecialOffer, Production.Product i Production.ProductSubcategory, aby uzyskać informacje o rabatach i podkategoriach produktów. Następnie wykorzystuje funkcję MIN i MAX, aby obliczyć minimalną i maksymalną kwotę rabatu dla każdej podkategorii. Warunek WHERE eliminuje produkty, które nie mają rabatu. Wyniki są grupowane po nazwie podkategorii i sortowane alfabetycznie.

8. Podaj produkty, których cena jest wyższa od średniej ceny produktów w sklepie.

SQL:

```
SELECT Name AS "Nazwa Produktu", ListPrice AS "Cena"
FROM Production.Product
WHERE ListPrice > (SELECT AVG(ListPrice)
                   FROM Production.Product);
```

WYNIK:

Nazwa Produktu	Cena
HL Road Frame - Black, 58	1431.50
HL Road Frame - Red, 58	1431.50
...	..
Road-750 Black, 48	539.99
Road-750 Black, 52	539.99

INTERPRETACJA:

Aby podać produkty których cena jest wyższa od średniej produktów w sklepie należy obliczyć średnią ceny produktów z tabeli produkt a później korzystając z zapytania w ramach warunku WHERE wypisać te na których cena jest wyższa niż obliczona średnia.

9. Ile średnio produktów w każdej kategorii sprzedaje się w poszczególnych miesiącach?

SQL:

```
DECLARE @cols NVARCHAR (MAX);
SELECT @cols = COALESCE (@cols + ',[' + [Name] + ']',
          '[' + [Name] + ']')
          FROM    (SELECT DISTINCT [Name] FROM [Production].[ProductCategory]) PV
          ORDER BY [Name]

DECLARE @query NVARCHAR(MAX)
SET @query ='SELECT
   OrderYear,
   OrderMonth,
   '+@cols +'
FROM
   (SELECT
      YEAR(SalesOrderHeader.OrderDate) AS OrderYear,
      MONTH(SalesOrderHeader.OrderDate) AS OrderMonth,
      ProductCategory.Name AS CategoryName,
                  AVG(CAST(SalesOrderDetail.OrderQty AS float)) AS AvgProductsSold
   FROM
      Sales.SalesOrderDetail
      JOIN Production.Product
         ON SalesOrderDetail.ProductID = Product.ProductID
      JOIN Production.ProductSubcategory
         ON Product.ProductSubcategoryID = ProductSubcategory.ProductSubcategoryID
      JOIN Production.ProductCategory
         ON ProductSubcategory.ProductCategoryID = ProductCategory.ProductCategoryID
      JOIN Sales.SalesOrderHeader
         ON SalesOrderDetail.SalesOrderID = SalesOrderHeader.SalesOrderID
   GROUP BY
      YEAR(SalesOrderHeader.OrderDate),
      MONTH(SalesOrderHeader.OrderDate),
      ProductCategory.Name) AS SourceTable
PIVOT
   (MIN(AvgProductsSold) FOR CategoryName IN ('+ @cols +')) AS PivotTable
ORDER BY
   OrderYear,
   OrderMonth;'

EXEC SP_EXECUTESQL @query;
```

WYNIK:

OrderYear	OrderMonth	Accesories	Bikes	Clothing	Components

2011	5	2.27027027027027	2.26600985221675	2.98214285714286	1.86885245901639
2011	6	NULL	1	NULL	NULL
...					
2014	4	1.00029515938607	1	1.01088270858525	2
2014	5	1.34695512820513	1.85849462365591	2.66241610738255	2.39268978444236
2014	6	1	NULL	1	NULL

INTERPRETACJA:

W celu klarownej reprezentacji przedstawienia średniej sprzedaży produktu w każdej kategorii zdecydowano się na użycie dynamicznego SQL wraz z operacją PIVOT. Na samym początku inicjujemy zmienną @cols typu NVARCHAR. Późniejsze zapytanie przypisuje do zmiennej @cols wartości kolumny [Name] z tabeli [Production].[ProductCategory]. Kolumny te zostaną wykorzystane później do operacji PIVOT. Wykorzystywana jest tutaj funkcja COALESCE, aby wartość @cols została ustawiona na null, jeśli nie istnieją wartości z tabeli [Production].[ProductCategory]. Zapytanie zapisuje wartość kolumny @cols w zapytaniu wykorzystując konkatenację łańcuchów, a także zawiera operację PIVOT, która grupuje wartości AVG(CAST(SalesOrderDetail.OrderQty AS float)) dla każdej kategorii produktów, w danym miesiącu i roku. Wynik końcowy jest sortowany według roku i miesiąca. EXEC SP_EXECUTESQL @query; Uruchamia zapytanie zapisane w zmiennej @query.

10. Ile średnio czasu klient czeka na dostawę zamówionych produktów? Przygotuj zestawienie w zależności od kodu regionu (SalesTerritory.CountryRegionCode).

SQL:

```
SELECT st.CountryRegionCode AS "Kod Regionu" , AVG(CAST(DATEDIFF(day, soh.OrderDate, soh.ShipDate)AS float)) AS "Srednia Czasu Dostawy"
FROM Sales.SalesOrderHeader soh
JOIN Sales.SalesOrderDetail sod ON soh.SalesOrderID = sod.SalesOrderID
JOIN Sales.SalesTerritory st ON soh.TerritoryID = st.TerritoryID
GROUP BY st.CountryRegionCode;
```

WYNIK:

Kod Regionu	Srednia Czasu Dostawy
AU	7
CA	7
DE	7.00371944739639
FR	7
GB	7
US	7

INTERPRETACJA:

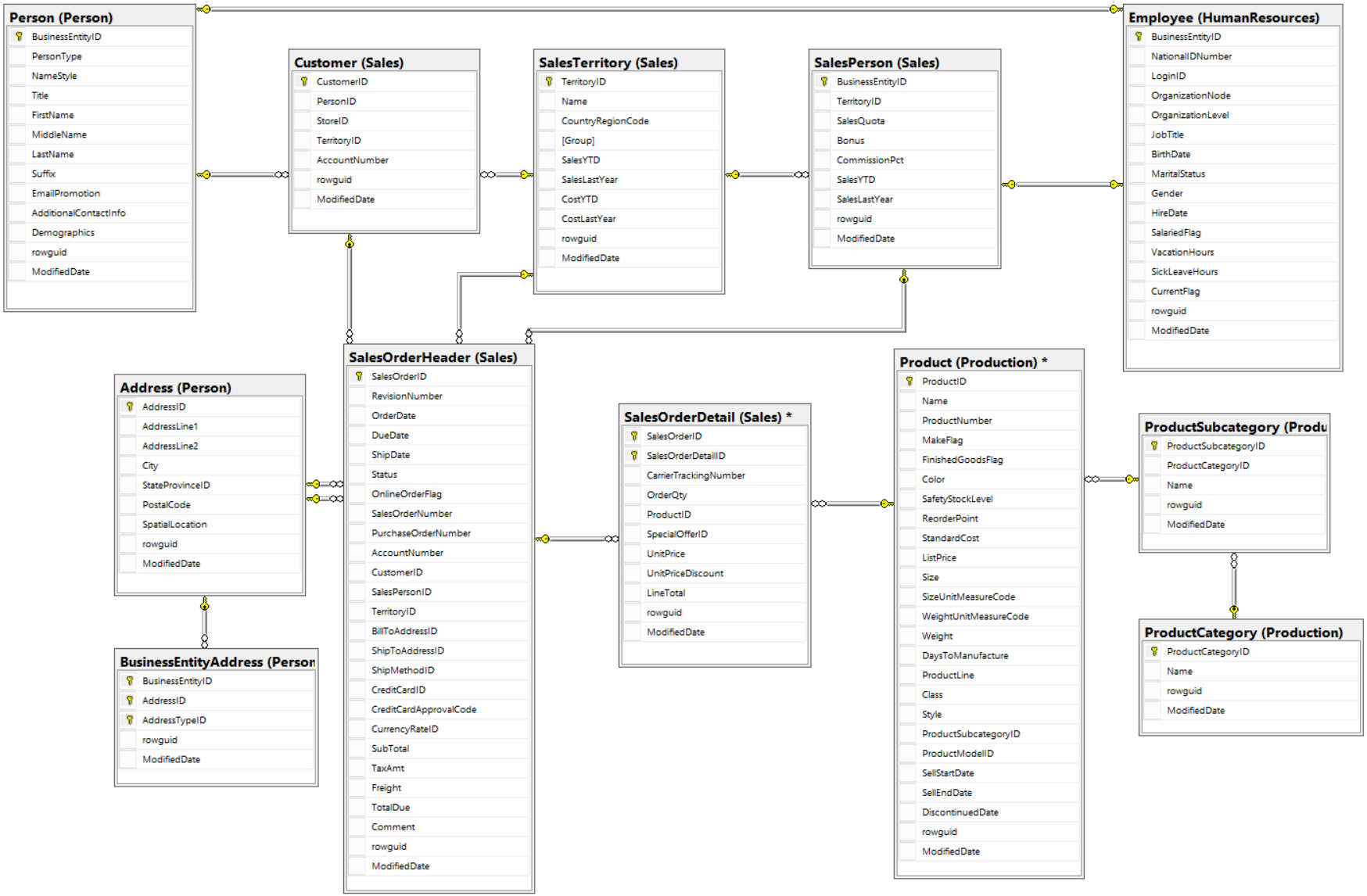
To zapytanie wybiera kod regionu kraju oraz średni czas dostawy dla zamówień złożonych w tym regionie. Wybiera kolumny z tabeli Sales.SalesOrderHeader, które zawierają kod regionu kraju oraz daty zamówienia i wysyłki. Następnie wykorzystując funkcję DATEDIFF oblicza różnicę w dniach między datą zamówienia i wysyłki, a funkcja CAST konwertuje tę wartość na liczbę zmiennoprzecinkową. Ostatecznie obliczana jest średnia czasu dostawy (AvgDeliveryTime) dla każdego regionu. Do zapytania dołączane są tabele Sales.SalesOrderHeader, Sales.SalesOrderDetail oraz Sales.SalesTerritory, aby uzyskać informacje o kodzie kraju, zamówieniach i terytoriach. Grupuje wyniki po kodzie regionu kraju, aby uzyskać średni czas dostawy dla każdego regionu.

Wnioski:

1. Stworzenie diagramu ERD byłoby lepszym wyborem dla modelu konceptualnego bazy danych niż diagram klas, ponieważ jest on bardziej przenośny na fizyczny model bazy danych. W diagramie klas brakuje np. pola Id, co utrudnia definiowanie ograniczeń.
2. Można zauważyć, że język używany w Systemie MS SQL 2017 (czyli TRANSACT lub T-SQL) różni się od czystego SQL. Niektóre typy danych wymagały zmiany.
3. System zarządzania bazą danych (SZBD) z dobrze napisanymi ograniczeniami skutecznie uniemożliwia użytkownikowi wprowadzenie niewłaściwych danych.
4. Każda odmiana SQL na konretnej bazy danych ma inaczej zapisywane zmienne, procedury i funkcje, w porównaniu do bazy Oracle inaczej się deklaruje zmienne, inne są nazwy procedur itd. ISNULL() – NVL()
5. Należy pamiętać o wyłączeniu i ponownym włączeniu ograniczeń dotyczących tabel po wprowadzeniu danych.
6. Przy diagramie klas trzeba analizować końce asocjacji by poprawnie określić w której tabeli ma się znajdować klucz obcy wskazujący na klucz główny innej tabeli.
7. Ciekawym zjawiskiem jest oznaczenie jednego końca asocjacji {bag} co odnosi się do kolekcji nieuporządkowanej posiadającej duplikaty. Jest to dość mylące biorąc pod uwagę klasę asocjacyjną oraz fakt trudności zrozumienia jak to przełożyć na związki relacji dla bazy danych.

Uwaga!

- Sprawozdanie, bez wniosków podsumowujących aspekt zagadnień analizowanych na zajęciach laboratoryjnych i zawartych w sprawozdaniu, jest automatycznie oceniane negatywnie!



Rysunek 2. Schemat relacji dla wybranych tabel bazy AdventureWorks