

PROJEKT

„Książka Kucharska”

Temat: Baza przechowuje informacje dotyczące przepisów kulinarnych, które dodawane są przez użytkowników.

Zaprojektowana baza stanowi zbiór przepisów kulinarnych. Składa się z ośmiu tabel powiązanych ze sobą relacjami. Służy ona ułatwieniu dostępu do informacji na temat przygotowywanej potrawy. Wśród nich możemy odnaleźć dane dotyczące poszczególnego przepisu, listy składników, właściwości potrawy czy nawet osoby, która jest jej autorem. Baza w sposób stosunkowo łatwy i szybki pozwala poznać się z zamieszczonymi danymi, a także usprawnia proces wyszukiwania interesujących nas zagadnień. Dzięki temu określone kryteria pozwalają automatycznie wygenerować użytkownikom zindywidualizowane propozycje.

Lista Tabel:

- **Przepisy**
Tabela zawiera informacje dotyczące wprowadzanego przepisu (m.in. nazwę przepisu, czas jego przygotowania czy datę jego wpisu do bazy)
- **Produkty_spozywcze**
Posiada informacje o produktach dodanych do bazy, ich średnie koszty i kalorie wyrażone jednostkowo co do ustalonej dla produktu miary.
- **Miara**
Zawiera zbiór jednostek miary w jakich wyrażane są produkty spożywcze używane w przepisach.
- **Rodzaj_kuchni**
Zawiera zbiór rodzajów kuchni, które kategoryzują przepisy według ich pochodzenia – jak np. kuchnia włoska.
- **Składniki**
Tabela ta łączy w sobie dane dotyczące przepisu z niezbędnymi do jego realizacji produktami tworząc listę składników.
- **Autorzy**
Posiada informacje na temat autorów przepisów – ich dane osobowe.
- **Typ_dania**
Zawiera zbiór typów dań do jakich możemy przypisać dany przepis – jak np. śniadanie czy obiad.
- **Poziom_trudności**
Zawiera dane określające poziom trudności przepisu w oparciu o ilość dopuszczalnych składników.

DEFINICJE TABEL

TABELA - Przepisy			
Nazwa pola	Typ danych	Właściwości	Uwagi
ID_Przepis	int	IDENTITY	<PK>
Nazwa_przepisu	NVARCHAR (50)	UNIQUE, NOT NULL	<FK>
ID_Au	int	NOT NULL	<FK>
ID_Poz_trudnosci	int	NOT NULL	<FK>
ID_Danie	int	NOT NULL	<FK>
ID_Kuchnia	int	NOT NULL	<FK>
Ilosc_porcji	int	NOT NULL	
Data_zapisu	datetime	NOT NULL	DEFAULT Getdate()
Czas_przygotowania	time(7)	NOT NULL	
Okładka	image		

TABELA - Produkty_spozywcze			
Nazwa pola	Typ danych	Właściwości	Uwagi
ID_Produkt	int	IDENTITY	<PK>
Nazwa_produktu	NVARCHAR (50)	UNIQUE, NOT NULL	
Kcal_jednostkowe	FLOAT	NOT NULL	
Koszt_jednostkowy	FLOAT	NOT NULL	
ID_Jed_miary	int	NOT NULL	<FK>

TABELA - Miara			
Nazwa pola	Typ danych	Właściwości	Uwagi
ID_Miara	int	IDENTITY	<PK>
Jednostka_miary	VARCHAR (4)	UNIQUE, NOT NULL	

TABELA - Rodzaj_kuchni			
Nazwa pola	Typ danych	Właściwości	Uwagi
ID_Rodzaj_Kuchni	int	IDENTITY	<PK>
Nazwa_kuchni	NVARCHAR (25)	UNIQUE, NOT NULL	

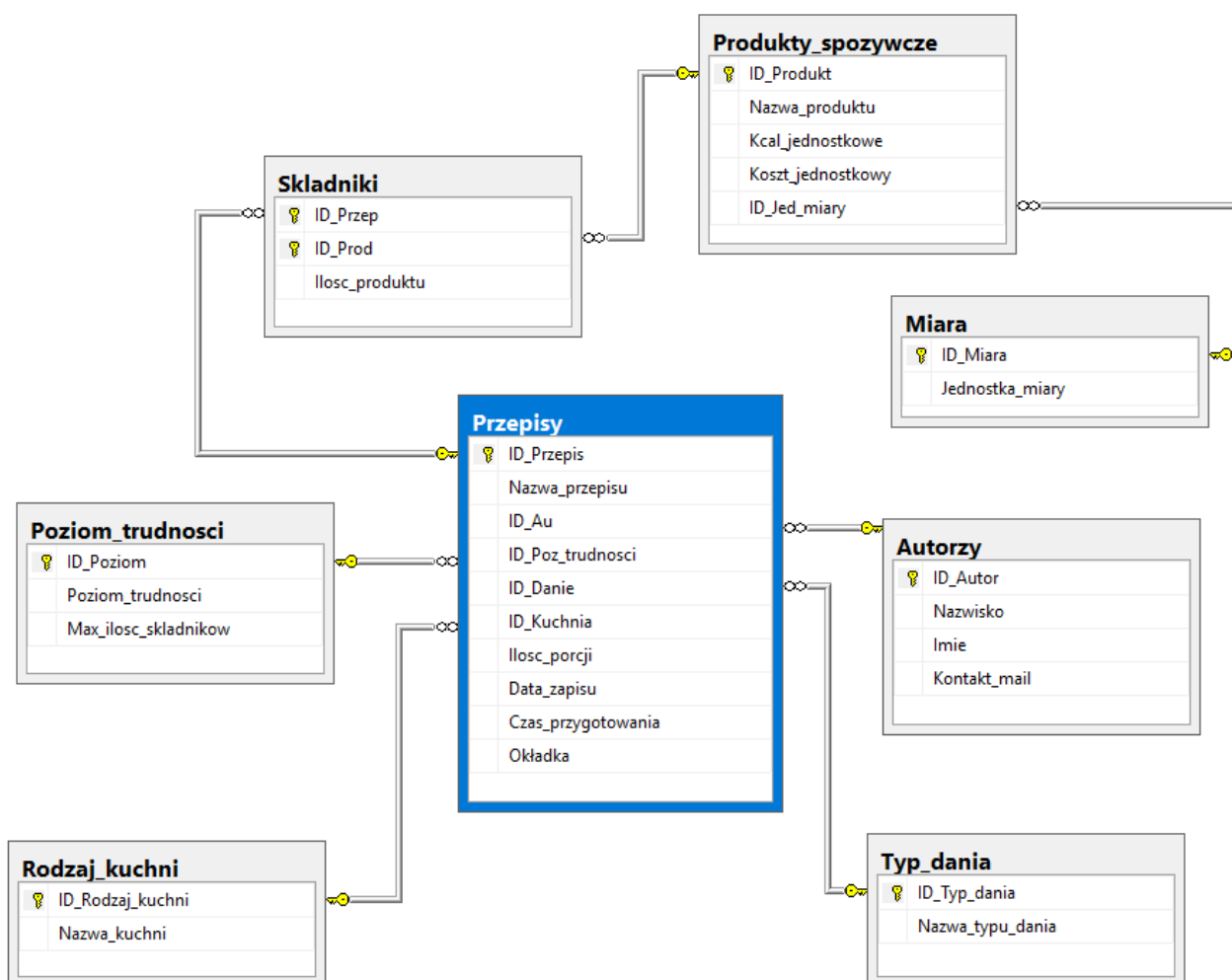
TABELA - Skladniki			
Nazwa pola	Typ danych	Właściwości	Uwagi
ID_Przep	int	NOT NULL	<PK>
ID_Prod	int	NOT NULL	<PK>
Ilosc_produktu	int	NOT NULL	

TABELA - Autorzy			
Nazwa pola	Typ danych	Właściwości	Uwagi
ID_Autor	int	IDENTITY	<PK>
Nazwisko	NVARCHAR (35)	NOT NULL	
Imie	NVARCHAR (25)	NOT NULL	
Kontakt_mail	NVARCHAR (45)		

TABELA - Typ_dania			
Nazwa pola	Typ danych	Właściwości	Uwagi
ID_Typ_dania	int	IDENTITY	<PK>
Nazwa_typu_dania	NVARCHAR (20)	UNIQUE, NOT NULL	

TABELA – Poziom_trudnosci			
Nazwa pola	Typ danych	Właściwości	Uwagi
ID_Poziom	int	IDENTITY	<PK>
Poziom_trudnosci	NVARCHAR (15)	UNIQUE, NOT NULL	
Max_ilosc_skladnikow	int	NOT NULL	

DIAGRAM



Równania relacji:

dbo.Produkty_spozywcze.ID_Jed_miary = dbo.Miara.ID_Miara

dbo.Produkty_spozywcze.ID_Produkt = dbo.Składniki.ID_Prod

dbo.Przepisy.ID_Przepis = dbo.Składniki.ID_Przep

dbo.Przepisy.ID_Au = dbo.Autorzy.ID_Autor

dbo.Przepisy.ID_Danie = dbo.Typ_dania.ID_Typ_dania

dbo.Przepisy.ID_Kuchnia = dbo.Rodzaj_kuchni.ID_Rodzaj_kuchni

dbo.Przepisy.ID_Poz_trudnosci = dbo.Poziom_trudnosci.ID_Poziom

Zadania bazy:**1. FUNKCJA - „Lista_składników”**

Po wpisaniu interesującego nas przepisu wyszukane zostają wszystkie składniki potrzebne do jego realizacji wraz z podaniem ilości oraz jednostki w jakiej są wyrażone produkty.

```

SQLQuery16.sql - D:\LKTF\Toshiba (55))*  SQLQuery15.sql - D:\LKTF\Toshiba (53))*  SQLQuery12.sql - D:\LKTF\Toshiba (54))*
use BD_Chamera_Projekt
GO
CREATE FUNCTION Lista_składników (@jaki_przepis nvarchar(50))
RETURNS TABLE
AS
RETURN
(
SELECT      Produkty_spozywcze.Nazwa_produktu, Składniki.Ilosc_produktu, Miara.Jednostka_miary
FROM        Produkty_spozywcze INNER JOIN
            Składniki ON Produkty_spozywcze.ID_Produkt = Składniki.ID_Prod INNER JOIN
            Miara ON Produkty_spozywcze.ID_Jed_miary = Miara.ID_Miara RIGHT OUTER JOIN
            Przepisy ON Składniki.ID_Przep = Przepisy.ID_Przepis
WHERE (Nazwa_przepisu = @jaki_przepis)
)

```

100 %

Messages

Command(s) completed successfully.

Realizacja:

```

SQLQuery20.sql - D:\LKTF\Toshiba (53))*  SQLQuery19.sql - D:\LKTF\Toshiba (53))*
use BD_Chamera_Projekt
SELECT* from Lista_składników ('Sałatka Grecka')

```

100 %

Results

	Nazwa_produktu	Ilosc_produktu	Jednostka_miary
1	Woda	15	ml
2	Sałata lodowa	1	szt.
3	Papryka	2	szt.
4	Czarna oliwki	20	szt.
5	Serfeta	100	g
6	Oliwa z oliwek	15	ml
7	Sos sałatkowy	1	szt.

Funkcja zwróciła w formie tabeli listę składników dla przepisu o nazwie „Sałatka Grecka”.

2. FUNKCJA – „przepis_z”

Służy do wyszukiwania przepisów zawierających dany produkt spożywczy na swojej liście składników.

```

use BD_Chamera_Projekt
go
CREATE FUNCTION przepis_z(@szukany_produknt nvarchar(50))
RETURNS TABLE
AS
RETURN
(
SELECT      Przepisy.Nazwa_przepisu
FROM        Produkty_spozywcze INNER JOIN
            Skladniki ON Produkty_spozywcze.ID_Produkt = Skladniki.ID_Prod INNER JOIN
            Przepisy ON Skladniki.ID_Przep = Przepisy.ID_Przepis
WHERE (Nazwa_produkntu = @szukany_produknt)
)

```

100 %

Messages

Command(s) completed successfully.

Realizacja:

```

use BD_Chamera_Projekt
SELECT* from przepis_z ('Cukier')

```

100 %

Results Messages

	Nazwa_przepisu
1	Magdalenki
2	Trifle

Wyszukanie przepisów zawierających jako składnik cukier.

3. FUNKCJA - „danie_z_kuchni”

Służy do wyszukiwania określonego typu dania ze wskazaniem preferencji co do pochodzenia potrawy – rodzaju kuchni.

```

use BD_Chamera_Projekt
go
CREATE FUNCTION danie_z_kuchni(@jakie_danie nvarchar(20), @jaka_kuchnia nvarchar(25))
RETURNS TABLE
AS
RETURN
(
SELECT      Przepisy.Nazwa_przepisu
FROM        Przepisy RIGHT OUTER JOIN
            Typ_dania ON Przepisy.ID_Danie = Typ_dania.ID_Typ_dania RIGHT OUTER JOIN
            Rodzaj_kuchni ON Przepisy.ID_Kuchnia = Rodzaj_kuchni.ID_Rodzaj_kuchni
WHERE (Nazwa_typu_dania = @jakie_danie)and (Nazwa_Kuchni = @jaka_kuchnia)
)

```

100 %

Messages

Command(s) completed successfully.

Realizacja:

```

SQLQuery4.sql - DE...LKTF\Toshiba (53))*  DESKTOP-EHFLKTF\S...ekt - dbo.Przepisy
use BD_Chamera_Projekt
select * from danie_z_kuchni('deser', 'francuska')
  
```

	Nazwa_przepisu
1	Magdalenki

Wyszukanie deserów kuchni francuskiej.

4. FUNKCJA – „jaki_czas_przygotowania”

Służy do wyszukiwania przepisów, których czas przygotowania jest mniejszy bądź równy temu wskazanemu przez nas.

```

SQLQuery9.sql - DE...LKTF\Toshiba (58))*  SQLQuery4.sql - DE...LKTF\Toshiba (53))*  SQLQuery5.sql - DE...LKTF\Toshiba (54))*
USE BD_Chamera_Projekt
GO
CREATE FUNCTION jaki_czas_przygotowania (@podaj_limit time(7))
RETURNS TABLE
AS
RETURN
(
SELECT      Przepisy.Nazwa_przepisu, Poziom_trudnosci.Poziom_trudnosci, Typ_dania.Nazwa_typu_dania, Przepisy.Czas_przygotowania
FROM        Przepisy LEFT OUTER JOIN
            Typ_dania ON Przepisy.ID_Danie = Typ_dania.ID_Typ_dania LEFT OUTER JOIN
            Poziom_trudnosci ON Przepisy.ID_Poz_trudnosci = Poziom_trudnosci.ID_Poziom

WHERE (Czas_przygotowania < @podaj_limit) OR (Czas_przygotowania = @podaj_limit)
)
  
```

Messages
Command(s) completed successfully.

Realizacja:

```

SQLQuery5.sql - DE...LKTF\Toshiba (54))*  SQLQuery7.sql - DE...LKTF\Toshiba (56))*  SQLQuery8.sql - DE...LKTF\Toshiba (57))*
use BD_Chamera_Projekt
select * from jaki_czas_przygotowania('00:25:00')
  
```

	Nazwa_przepisu	Poziom_trudnosci	Nazwa_typu_dania	Czas_przygotowania
1	Trifle	Umiarkowany	Deser	00:20:00.0000000
2	Naleśniki	Łatwy	Śniadanie	00:25:00.0000000
3	Salatka Grecka	Umiarkowany	Kolacja	00:15:00.0000000

Wyszukanie przepisów, których czas przygotowania nie przekracza 25 minut.

5. WIDOK – „Ostatnio_dodane”

Wyświetla przepisy, które zostały dodane do bazy w przeciągu ostatnich dwóch tygodni.

```
SQLQuery16.sql - D:\LKTF\Toshiba (55))*  SQLQuery21.sql - D:\LKTF\Toshiba (54))*  SQLQuery20.sql - D:\LKTF\Toshiba (53))*

use BD_Chamera_Projekt
GO
CREATE VIEW Ostatnio_dodane
AS SELECT      Przepisy.Nazwa_przepisu, Przepisy.Data_zapisu, Poziom_trudnosci.Poziom_trudnosci,
               Autorzy.Imie, Autorzy.Nazwisko
FROM          Przepisy INNER JOIN
               Autorzy ON Przepisy.ID_Au = Autorzy.ID_Autor INNER JOIN
               Poziom_trudnosci ON Przepisy.ID_Poz_trudnosci = Poziom_trudnosci.ID_Poziom
WHERE Data_zapisu > (GETDATE() - DATEPART(dd,14))

100 %
Messages
Command(s) completed successfully.
```

Realizacja:

```
SQLQuery16.sql - D:\LKTF\Toshiba (55))*  SQLQuery21.sql - D:\LKTF\Toshiba (54))*  SQLQuery20.sql - D:\LKTF\Toshiba (53))*

use BD_Chamera_Projekt
SELECT* from ostatnio_dodane
```

100 %

Results Messages

	Nazwa_przepisu	Data_zapisu	Poziom_trudnosci	Imie	Nazwisko
1	Nachos z guacamole	2018-01-25 16:34:20.000	Umiarkowany	Zygmunt	Lasowski
2	Kluski Śląskie	2018-01-23 07:02:02.000	Łatwy	Jolanta	Drozd
3	Coxinha	2018-01-19 06:02:27.000	Umiarkowany	Przemysław	Konieczny

6. WIDOK – „Aktywnosc_autorow”

Wyświetla ranking pięciu autorów, którzy dodali najwięcej przepisów do bazy.

```
SQLQuery3.sql - DE\...LKTF\Toshiba (53))*  SQLQuery2.sql - DE\...LKTF\Toshiba (55))*  SQLQuery1.sql - DE\...LKTF\Toshiba (52))*

use BD_Chamera_Projekt
GO
CREATE VIEW Aktywnosc_autorow
AS SELECT top 5 COUNT(ID_Au) as 'Ilość dodanych przepisów', Autorzy.Nazwisko, Autorzy.Imie
FROM          Autorzy LEFT OUTER JOIN
               Przepisy ON Autorzy.ID_Autor = Przepisy.ID_Au
group by      Autorzy.Nazwisko, Autorzy.Imie
order by      COUNT(ID_Au) desc

100 %
Messages
Command(s) completed successfully.
```

Realizacja:

SQLQuery3.sql - DE...LKTF\Toshiba (53))* SQLQuery2.sql

```
use BD_Chamera_Projekt
SELECT * from Aktywnosc_autorow
```

100 %

Results Messages

	Ilość dodanych przepisów	Nazwisko	Imie
1	2	Drozd	Jolanta
2	1	Filipiak	Bożena
3	1	Kaczmarek	Ilona
4	1	Maliniak	Małgorzata
5	1	Opolska	Paulina

7. WIDOK – „wykaz” (pomocniczy – użyty przy funkcji nr 8, 9 i 10)

Wyświetla informacje dotyczące składników w danym przepisie – ich kaloryczność i koszty jednostkowe jak i w przeliczeniu na potrzebną do przepisu ilość.

SQLQuery1.sql - DE...LKTF\Toshiba (52))* SQLQuery2.sql - DE...LKTF\Toshiba (53))*

```
use BD_Chamera_Projekt
GO
CREATE VIEW wykaz
AS SELECT      dbo.Przepisy.Nazwa_przepisu, dbo.Produkty_spozywcze.Nazwa_produkту, dbo.Skladniki.Ilosc_produkту, dbo.Produkty_spozywcze.Koszt_jednostkowy, dbo.Produkty_spozywcze.Kcal_jednostkowe,
               dbo.Produkty_spozywcze.Kcal_jednostkowe * dbo.Skladniki.Ilosc_produkту AS Kaloryczność,
               dbo.Produkty_spozywcze.Koszt_jednostkowy * dbo.Skladniki.Ilosc_produkту AS Koszt_skladnika, dbo.Przepisy.Ilosc_porcji
FROM           dbo.Produkty_spozywcze RIGHT OUTER JOIN
               dbo.Skladniki ON dbo.Produkty_spozywcze.ID_produkту = dbo.Skladniki.ID_Prod RIGHT OUTER JOIN
               dbo.Przepisy ON dbo.Skladniki.ID_Przep = dbo.Przepisy.ID_Przepis
```

100 %

Messages

Command(s) completed successfully.

Realizacja:

```
USE BD_Chamera_Projekt
SELECT * FROM wykaz
```

100 %

Results Messages

	Nazwa_przepisu	Nazwa_produkту	Ilosc_produkту	Koszt_jednostkowy	Kcal_jednostkowe	Kaloryczność	Koszt_skladnika	Ilosc_porcji
1	Magdalenki	Jajko	3	0,55	60	180	1,65	15
2	Magdalenki	Mąka pszenna	150	0,003	0,364	54,6	0,45	15
3	Magdalenki	Masło	100	0,0433	7,168	716,8	4,33	15
4	Magdalenki	Cukier	100	0,00246	3,867	386,7	0,246	15
5	Nachos z guacamole	Mąka pszenna	100	0,003	0,364	36,4	0,3	2
6	Nachos z guacamole	Woda	15	0,001	0	0	0,015	2
7	Nachos z guacamole	Awokado	2	2,99	227	474	7,98	2

8. FUNKCJA – „Koszty_porcji”

Funkcja służy do wyszukania kosztu jednej porcji podanego przepisu.

```
DESKTOP-EHFLKTF\S...ekt - dbo.Przepisy  SQLQuery2.sql - DE...LKTF\Toshiba (53))* X
use BD_Chamera_Projekt
GO
CREATE FUNCTION Koszty_porcji (@Szukany_przepis nvarchar(60))
RETURNS FLOAT
AS
BEGIN
DECLARE @x FLOAT , @y float
SELECT @x=sum(Koszt_skladnika)from wykaz where Nazwa_przepisu=@Szukany_przepis
SELECT @y=(Ilosc_porcji) from wykaz where Nazwa_przepisu=@Szukany_przepis
RETURN round(@x/@y, 2)
END

100 % <
Messages
Command(s) completed successfully.
```

Realizacja:

```
DESKTOP-EHFLKTF\S...ekt - dbo.Przepisy  SQLQuery12.sql - D...LKTF\Toshiba (54))* X
use BD_Chamera_Projekt
SELECT dbo.Koszty_porcji ('Pizza') as koszt

100 % <
Results Messages
koszt
1 8,75
```

Sprawdzono ile wynosi koszt porcji pizzy.

9. FUNKCJA – „Kaloryczność_na_porcje”

Umożliwia sprawdzenie kaloryczności potrawy w przeliczeniu na porcje po podaniu nazwy sprawdzanego przepisu.

```
SQLQuery1.sql - DE...LKTF\Toshiba (52))*  SQLQuery2.sql - DE...LKTF\Toshiba (53))* X
use BD_Chamera_Projekt
GO
CREATE FUNCTION Kaloryczność_na_porcje (@Sprawdzany_przepis nvarchar(60))
RETURNS FLOAT
AS
BEGIN
DECLARE @a FLOAT , @b float
SELECT @a=sum(Kaloryczność)from wykaz where Nazwa_przepisu=@Sprawdzany_przepis
SELECT @b=(Ilosc_porcji) from wykaz where Nazwa_przepisu=@Sprawdzany_przepis
RETURN @a/@b
END

100 % <
Messages
Command(s) completed successfully.
```

Realizacja:

```

DESKTOP-EHFLKTF\S...kt - dbo.Składniki  SQLQuery10.sql - D...\LKTF\Toshiba (55))*  SQLQuery5.sql - D...
use BD_Chamera_Projekt
SELECT dbo.Kaloryczność_na_porcje ('Naleśniki') as kcal

```

	kcal
1	121,32

Sprawdzono ile kalorii posiada porcja naleśników.

10. FUNKCJA – „Limity_ceny_i_kcal”

Służy do wyszukiwania tych przepisów, których cena i kaloryczność w przeliczeniu na porcję są niższe od wskazanych w zapytaniu limitów.

```

SQLQuery4.sql - DE...\LKTF\Toshiba (52))*  SQLQuery3.sql - DE...\LKTF\Toshiba (56))*  SQLQuery2.sql - DE...\LKTF\Toshiba (55))*
use BD_Chamera_Projekt
GO
CREATE FUNCTION Limity_ceny_i_kcal (@limit_ceny float, @limit_kcal float)
RETURNS TABLE
AS
RETURN
( SELECT
  Nazwa_przepisu, ilosc_porcji, ROUND(SUM(Kaloryczność)/ilosc_porcji,2) as Kcal_w_porcji,
  ROUND(SUM(Koszt_skladnika)/ilosc_porcji,2) as Cena_na_porcje
FROM      wykaz
group by  Nazwa_przepisu,ilosc_porcji
HAVING    (SUM(Koszt_skladnika)/ilosc_porcji < @limit_ceny) and (SUM (Kaloryczność)/ilosc_porcji < @limit_kcal)
)

```

100 % <

Messages

Command(s) completed successfully.

Realizacja:

```

SQLQuery4.sql - DE...\LKTF\Toshiba (52))*  SQLQuery3.sql - DE...\LKTF\Toshiba (56))*
use BD_Chamera_Projekt
select*from Limity_ceny_i_kcal (5, 200)

```

	Nazwa_przepisu	ilosc_porcji	Kcal_w_porcji	Cena_na_porcje
1	Magdalenki	15	89,21	0,45
2	Naleśniki	10	121,32	0,49
3	Salátka Grecka	3	195,73	3,24

Wyszukano przepisów o cenie mniejszej niż 5 zł i kaloryczności poniżej 200kcal na porcję.

11. WIDOK – „Maksimum” (pomocniczy – użyty w triggerze nr 12)

Zawiera informacje dotyczące ilości wprowadzonych składników do danego przepisu, a także ile można jeszcze wprowadzić (poz. Różnica) mając na uwadze maksymalną ilość składników dla danego poziomu trudności.

```

SQLQuery4.sql - DE...LKTF\Toshiba (52))* X
Use BD_Chamera_Projekt
go
CREATE VIEW Maksimum
AS SELECT      dbo.Przepisy.Nazwa_przepisu, dbo.Poziom_trudnosci.Max_ilosc_skladnikow,
COUNT(dbo.Skladniki.ID_Prod) AS Ilosc_skladnikow,
dbo.Poziom_trudnosci.Max_ilosc_skladnikow - COUNT(dbo.Skladniki.ID_Prod) AS Różnica,
      dbo.Skladniki.ID_Przep
FROM
      dbo.Skladniki RIGHT OUTER JOIN
      dbo.Przepisy ON dbo.Skladniki.ID_Przep = dbo.Przepisy.ID_Przepis LEFT OUTER JOIN
      dbo.Poziom_trudnosci ON dbo.Przepisy.ID_Poz_trudnosci = dbo.Poziom_trudnosci.ID_Poziom
GROUP BY dbo.Przepisy.Nazwa_przepisu, dbo.Poziom_trudnosci.Max_ilosc_skladnikow, dbo.Skladniki.ID_Przep

```

100 % <

Messages

Command(s) completed successfully.

Realizacja:

SQLQuery2.sql - DE...LKTF\Toshiba (55))* X SQLQuery1.sql - DE...LKTF\Toshiba (54))*

```

USE BD_Chamera_Projekt
SELECT * FROM Maksimum

```

100 % <

Results Messages

	Nazwa_przepisu	Max_ilosc_skladnikow	Ilosc_skladnikow	Różnica	ID_Przep
1	Coxinha	12	9	3	11
2	Kluski Śląskie	5	3	2	7
3	Magdalenki	5	4	1	1
4	Nachos z guacamole	12	7	5	3
5	Naleśniki	5	5	0	6
6	Pizza	25	13	12	12
7	Salatka Grecka	12	7	5	9
8	Trifle	12	8	4	5

12. TRIGGER - „dodawanie_skladnikow”

Sprawdzanie czy kolejny wpisywany składnik nie przekracza limitu dozwolonego na dany poziom trudności. W przypadku jego przekroczenia blokuje możliwość zapisania danych i zwraca komunikat o tym informujący.

```

SQLQuery5.sql - DE...LKTF\Toshiba (53))*
USE BD_Chamera_Projekt
GO
CREATE TRIGGER [dbo].[dodawanie_skladnikow] ON skladniki
AFTER INSERT, UPDATE AS
DECLARE @c int, @d int
SELECT @c=inserted.ID_Przep FROM INSERTED
SELECT @d=Różnica from Maksimum where ID_Przep=@c
IF @d > 0 or @d = 0 PRINT 'Składnik wprowadzono poprawnie. Pozostała liczba składników do wprowadzenia = ' + cast(@d as varchar)
ELSE
BEGIN
PRINT 'Składnik nie może zostać dodany! Przekroczono w tym przepisie limit produktów na dany poziom trudności.'
ROLLBACK
END

```

100 %

Messages
Command(s) completed successfully.

Realizacja:

ID_Przep	ID_Prod	Ilosc_produktu
12	2	250
12	3	150
12	6	5
12	8	1
12	9	10
12	11	125
12	1018	1
12	1019	25
12	1028	1
12	1029	50
12	1030	75
12	1031	6
12	1032	75
6	16	20
NULL	NULL	NULL

Microsoft SQL Server Management Studio

No row was updated.

The data in row 57 was not committed.
Error Source: .Net SqlClient Data Provider.
Error Message: The transaction ended in the trigger. The batch has been aborted.
Składnik nie może zostać dodany! Przekroczono w tym przepisie limit produktów na dany poziom trudności.

Correct the errors and retry or press ESC to cancel the change(s).

OK Pomoc

Po próbie dodania nowego składnika do listy przepisu „Naleśniki” (ID_Przep=6) zablokowano wprowadzanie rekordu i wyświetlono komunikat.