

Celem ćwiczeń jest zaprojektowanie oraz utworzenie na serwerze bazy danych przechowującej informacje na temat danych kontaktowych. Celem jest również zapoznanie z podstawowymi zapytaniami języka SQL służącymi do definiowania struktur bazy danych oraz manipulacji na danych.

Projektowana baza ma przechowywać następujące informacje:

grupa kontaktu, imię, nazwisko, adres, telefon, mobile, email

Przykładowy kontakt w naszej bazie miałby postać:

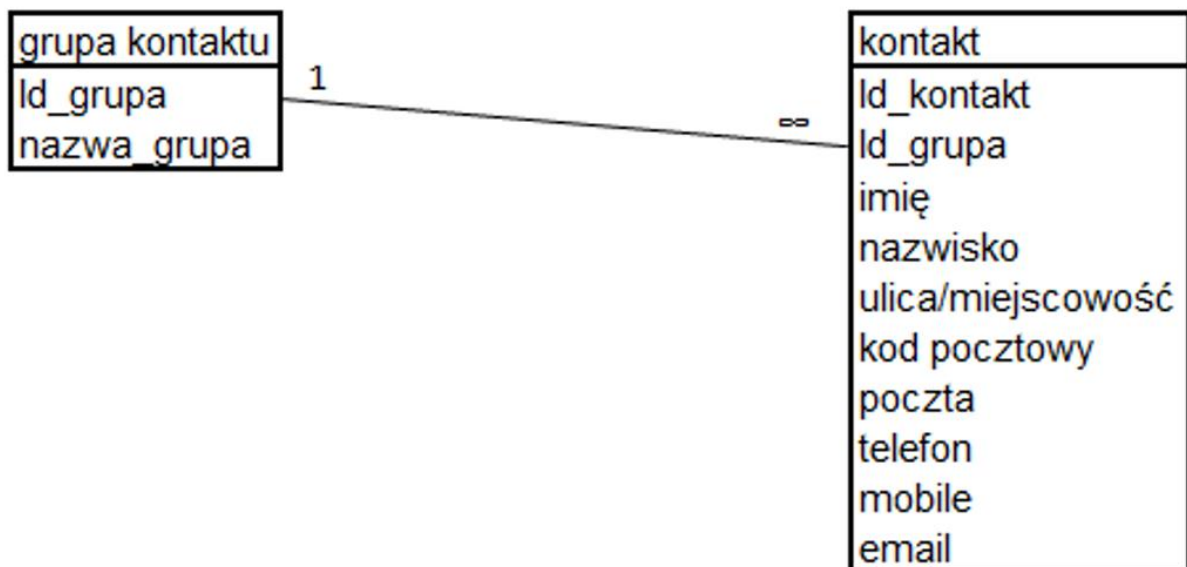
koledzy, Jan, Nowak, ul. Niecała 8/23, 52-128, Wrocław, 713285926, +48602532152, jan.nowak@o2.pl

Ze względu na to, że w ramach kolumny **grupa kontaktu** dochodziło by do redundancji czyli nadmiarowości danych w celu jej wyeliminowania dane będą przechowywane w dwóch tabelach:

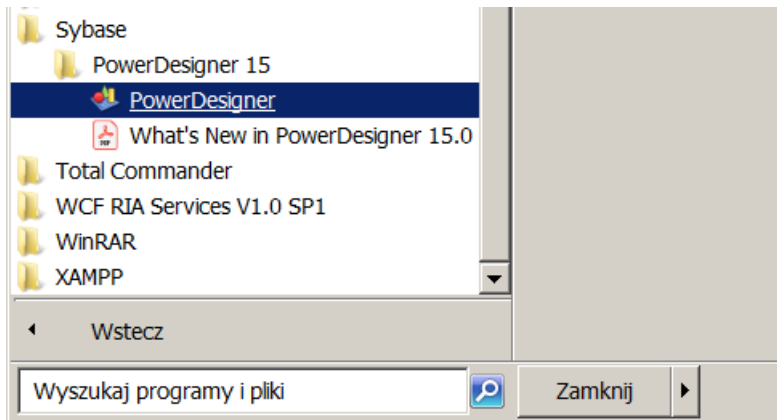
grupa kontaktu
koledzy
rodzina
szkoła
studia
wrogowie
VIP

imię	nazwisko	adres	telefon	mobile	email
Jan	Kowalski	ul. Niecała 8/25 52-128 Wrocław	713285956	+48603564789	jan.kowalski@o2.pl
Maurycy	Niebylski	Miękinia 58	712354566	+48608564236	niebylski@interia.pl

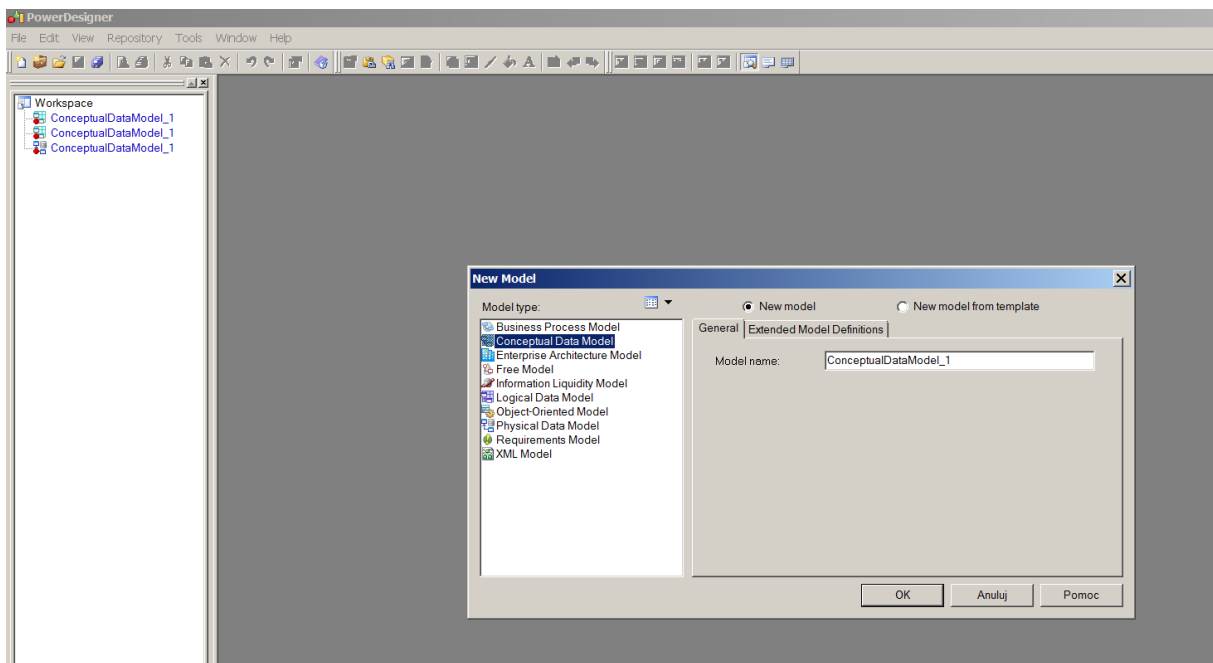
W celu zachowania informacji do jakiej grupy przypisany jest określony kontakt tabele te połączone zostaną relacją 1-n jak poniżej



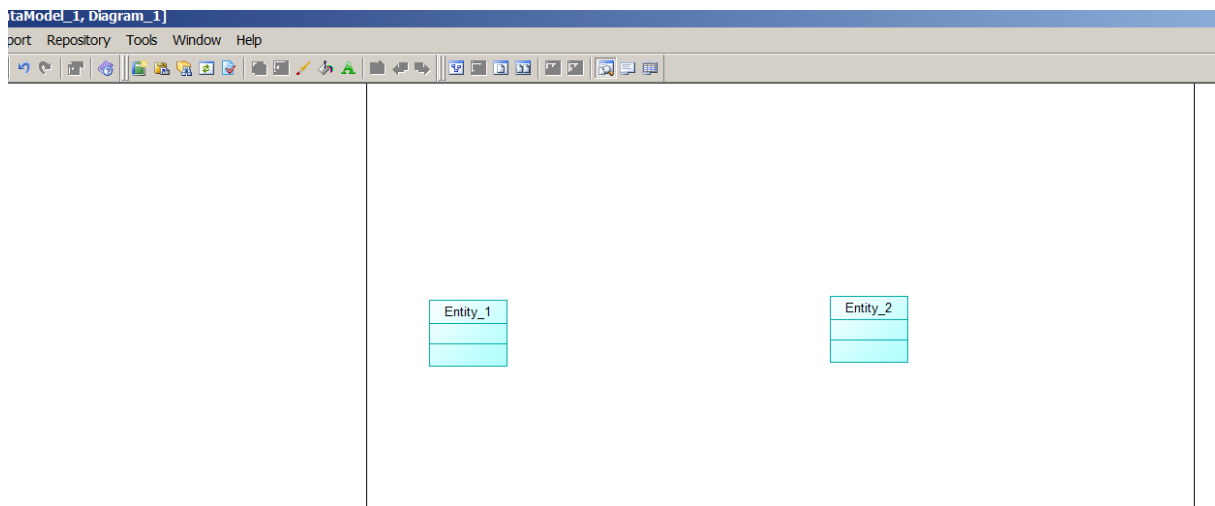
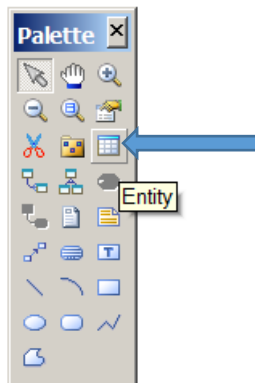
Rozpoczniemy od zaprojektowania diagramu bazy danych zgodnie z przyjętymi założeniami w programie **PowerDesigner**.



Wykonamy diagram konceptualny tej bazy.



W diagramie umieszczamy dwie **Encje** (tabele)



Następnie **PKM** (prawy klawisz myszy) by anulować wstawianie kolejnych encji.

Klikamy dwukrotnie **LKM** (lewy klawisz myszy) na **encji_1** by zdefiniować jej właściwości.

Wprowadzamy jej nazwę.

Entity Properties - Entity_1 (Entity_1)

General | Attributes | Identifiers | Mapping | Notes | Rules | Related Diagrams | Dependencies | Extended Dependencies | Version Info

Name: grupa

Code: grupa

Comment:

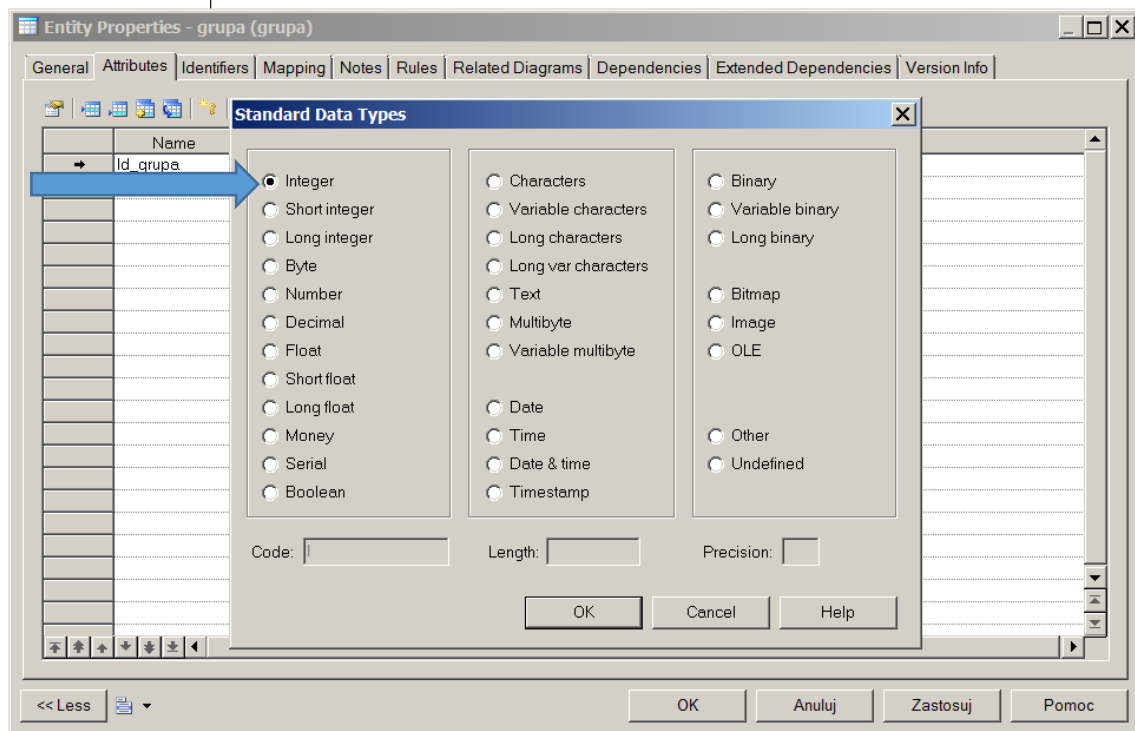
Stereotype:

Number: Generate

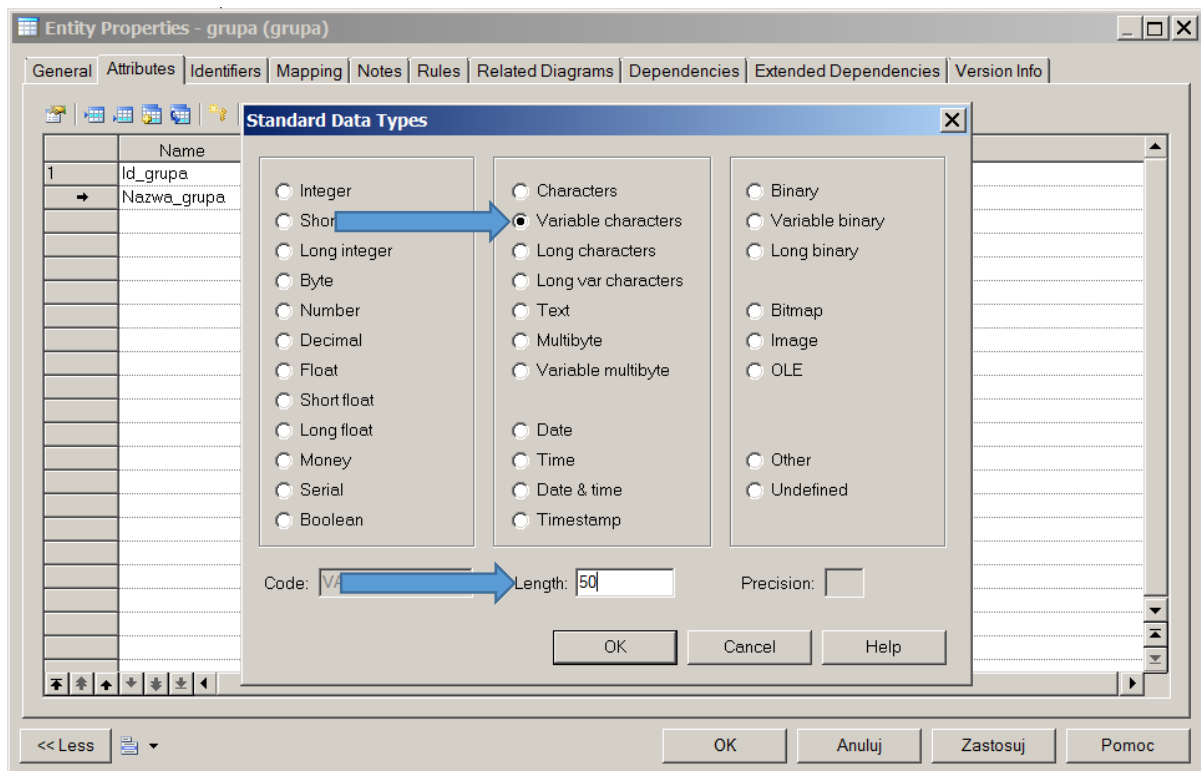
Parent Entity: <None>

<< Less OK Anuluj Zastosuj Pomoc

W przypadku klucza głównego będzie to liczba całkowita, zatem typ danych: **Integer**.



Definiujemy typ danych dla tego atrybutu. W przypadku nazw grup będą to ciągi tekstowe, zatem optymalnym będzie tu typ danych: **Variable characters** o maksymalnej długości ciągu równej **50**.



Zamykamy edycję tej tabeli, uzyskamy efekt jak poniżej.

grupa	
<u>Id_grupa</u>	<pi> Integer <M>
Nazwa_grupa	Variable characters (50) <M>
Id_grupa	<pi>

Entity_2

Przechodzimy do edycji drugiej tabeli.

Wprowadzamy jej nazwę.

Entity Properties - Entity_2 (Entity_2)

General | Attributes | Identifiers | Mapping | Notes | Rules | Related Diagrams | Dependencies | Extended Dependencies | Version Info

Name: kontakt =

Code: kontakt =

Comment:

Stereotype:

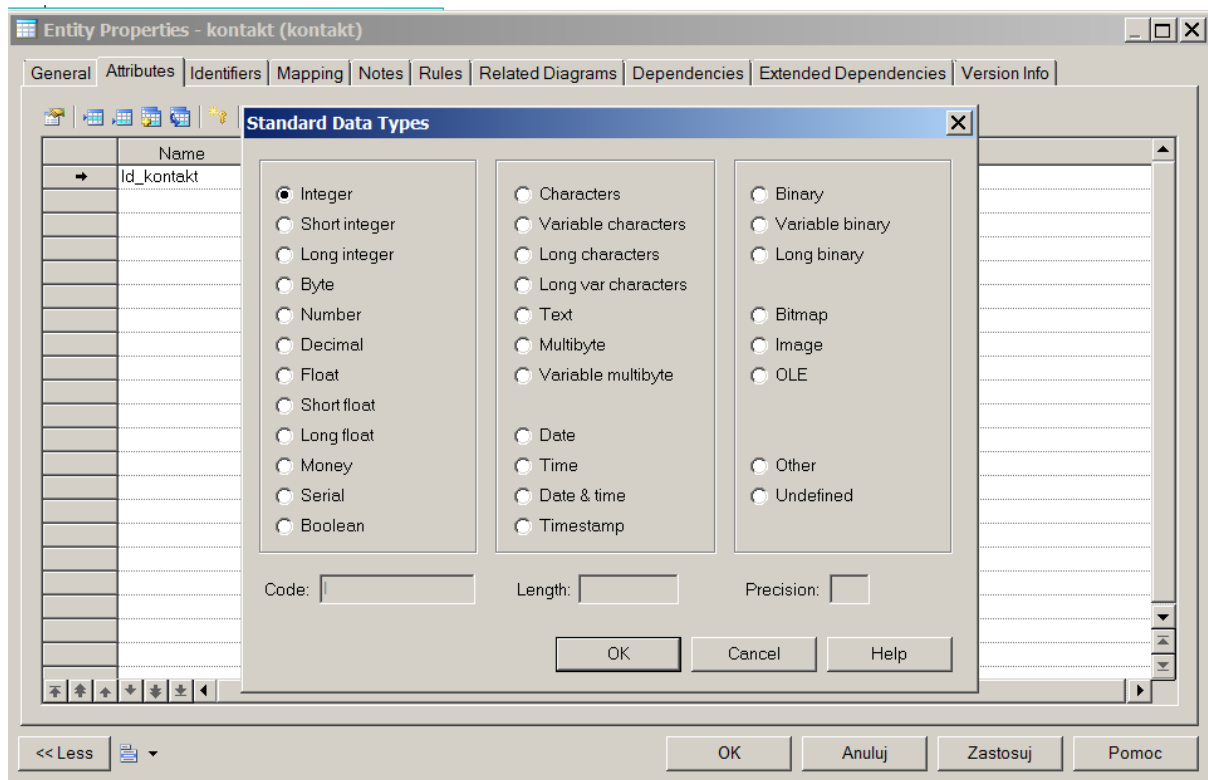
Number: Generate

Parent Entity: <None>

<< Less ▼

OK Anuluj Zastosuj Pomoc

Definiujemy typ danych dla tego atrybutu.



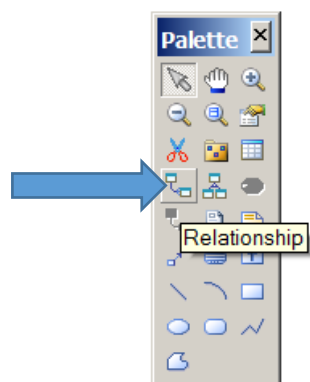
Zamykamy edycję tej tabeli, uzyskamy efekt jak poniżej.

grupa			
<u>Id_grupa</u>	<pi>	Integer	<M>
Nazwa_grupa		Variable characters (50)	<M>
Id_grupa	<pi>		

kontakt			
<u>Id_kontakt</u>	<pi>	Integer	<M>
Imie		Variable characters (50)	<M>
Nazwisko		Variable characters (50)	<M>
Adres_1		Variable characters (150)	<M>
Adres_2		Characters (6)	<M>
Adres_3		Variable characters (100)	<M>
Telefon		Variable characters (20)	<M>
Mobile		Variable characters (20)	<M>
Email		Variable characters (100)	<M>
Id_kontakt	<pi>		

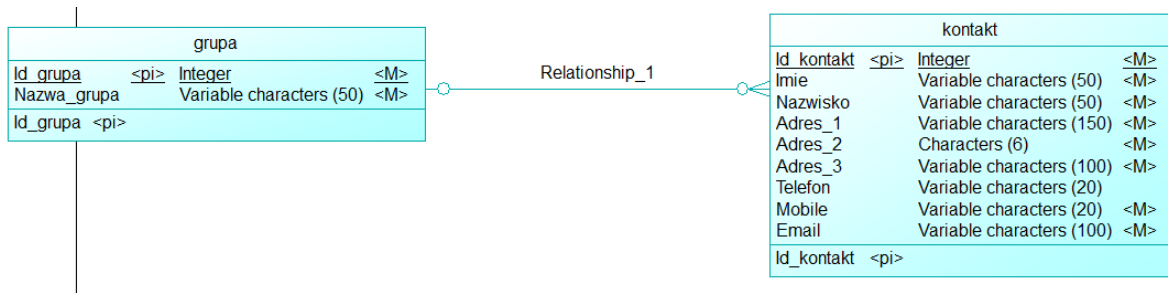
Następnie utworzymy pomiędzy zaprojektowanymi tabelami relację.

Wybieramy opcję jak poniżej.



Trzymając LKM rysujemy linię od wnętrza jednej tabeli do wnętrza drugiej, nie ma znaczenia, od której do której tabeli ją utworzymy, relację można dowolnie przedefiniować.

U Państwa może wyglądać to inaczej, ważne jedynie by relacja została utworzona.



Klikamy dwukrotnie LKW na utworzoną relację oraz przechodzimy na zakładkę: **Cardinalities**.

Relationship Properties - Relationship_1 (Relationship_1)

Entity 1 Entity 2

grupa kontakt

General Cardinalities Notes Rules

One - One One - Many Many - One Many - Many

Dominant role: <None>

grupa to kontakt

Role name:

Dependent Mandatory Cardinality: 0,n

kontakt to grupa

Role name:

Dependent Mandatory Cardinality: 0,1

More >>

Definiujemy końce naszej relacji.

W ramach określonej grupy, może znajdować się zero lub wiele kontaktów rozwijamy zatem listę i wybieramy jak poniżej.

grupa to kontakt

Role name:

Dependent Mandatory Cardinality: **0,n**

kontakt to grupa

Role name:

Dependent Mandatory Cardinality: **0,1**

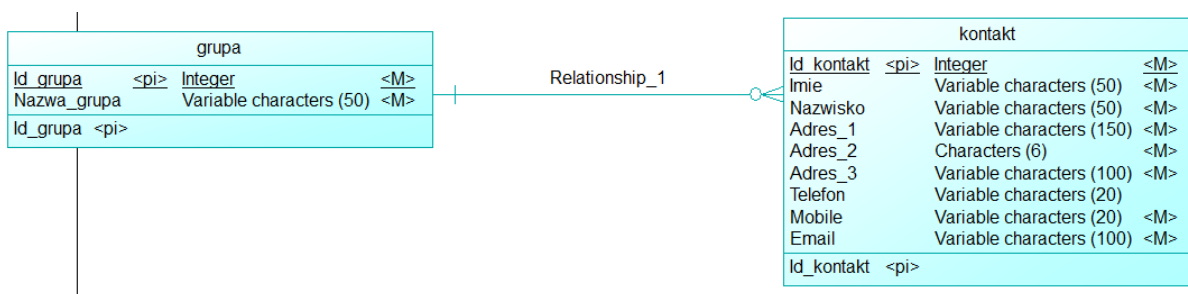
Określony kontakt w naszym przypadku będzie natomiast należał dokładnie do jednej grupy.

kontakt to grupa

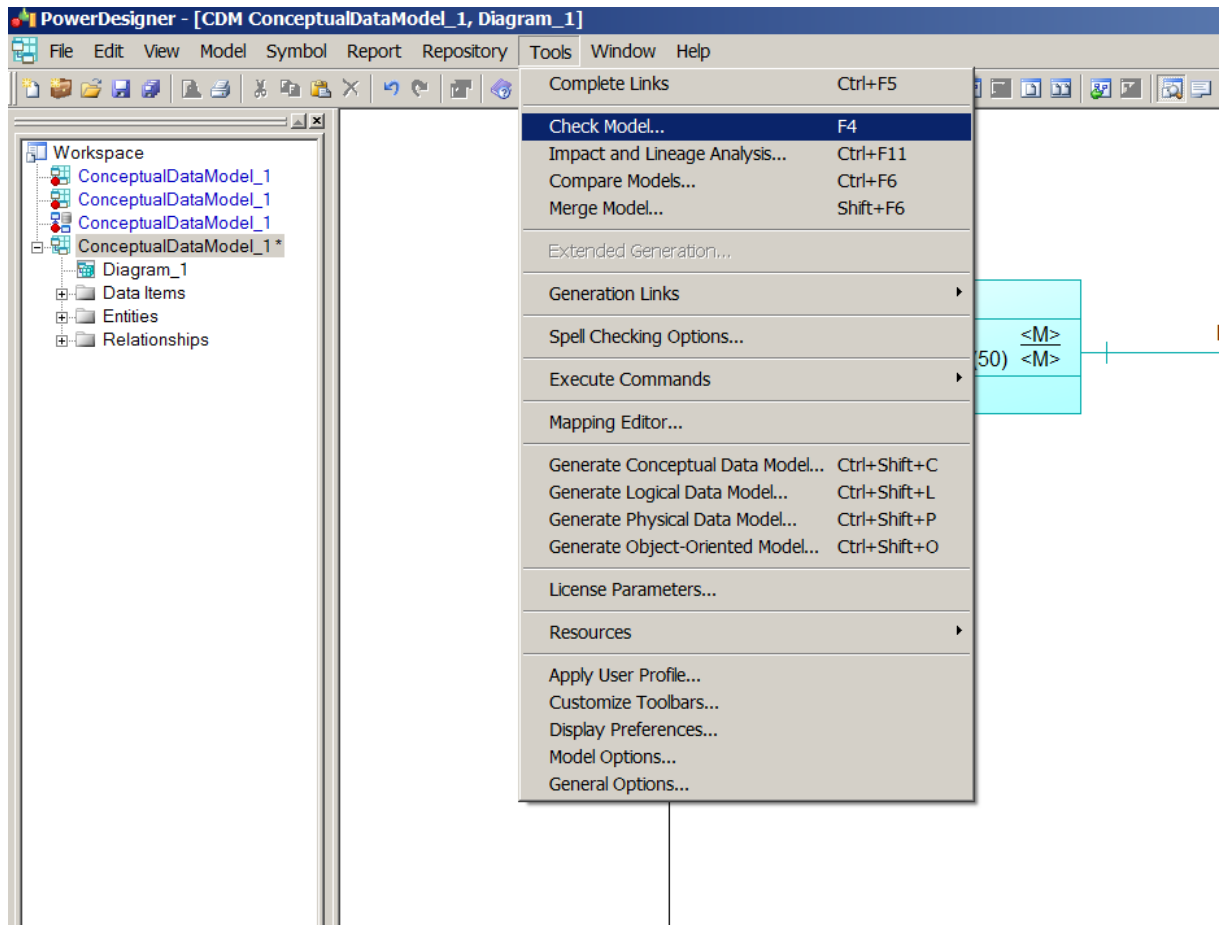
Role name:

Dependent Mandatory Cardinality: **1,1**

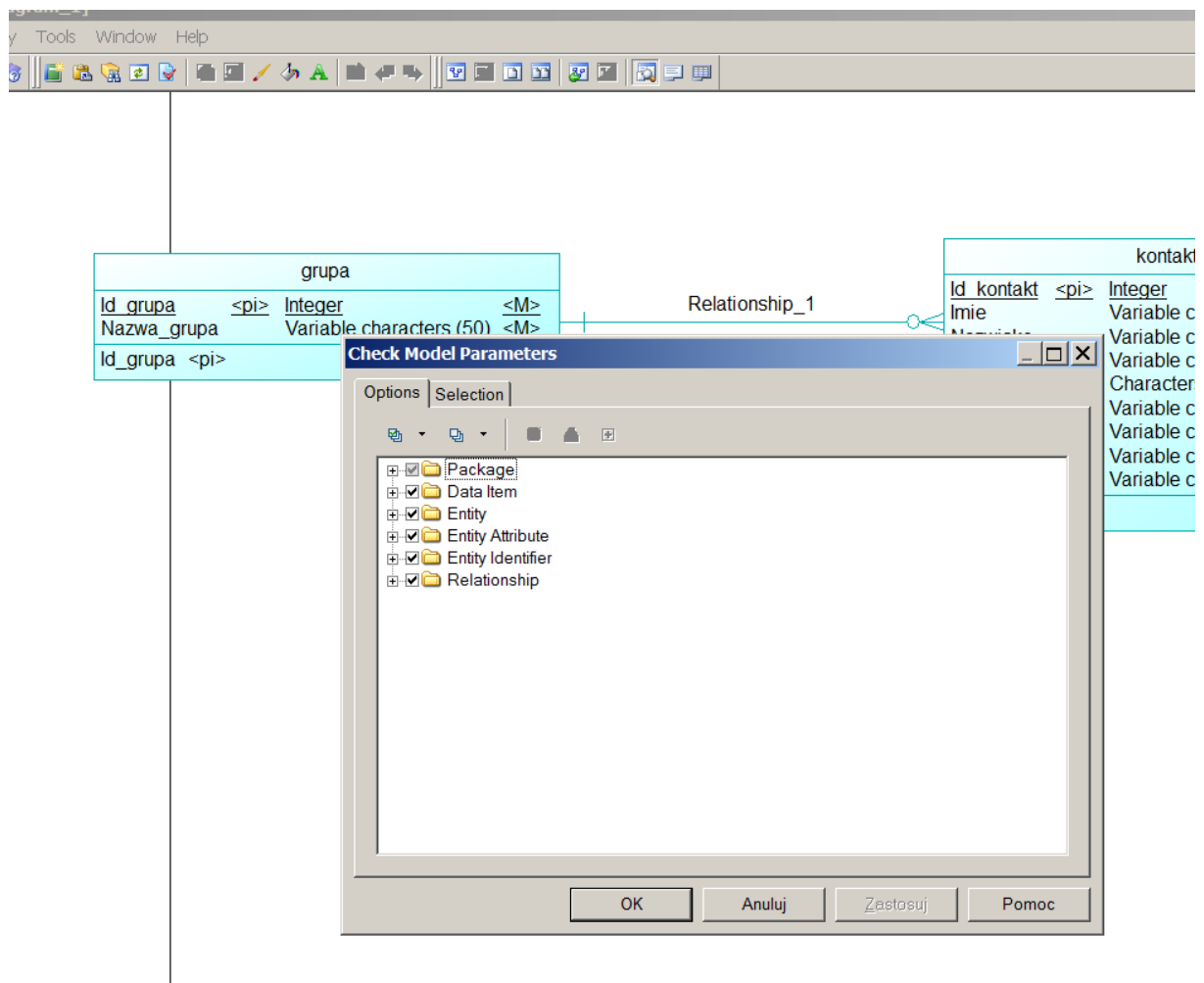
Po zakończeniu edycji relacji uzyskamy efekt jak poniżej.



Sprawdzimy teraz poprawność diagramu.

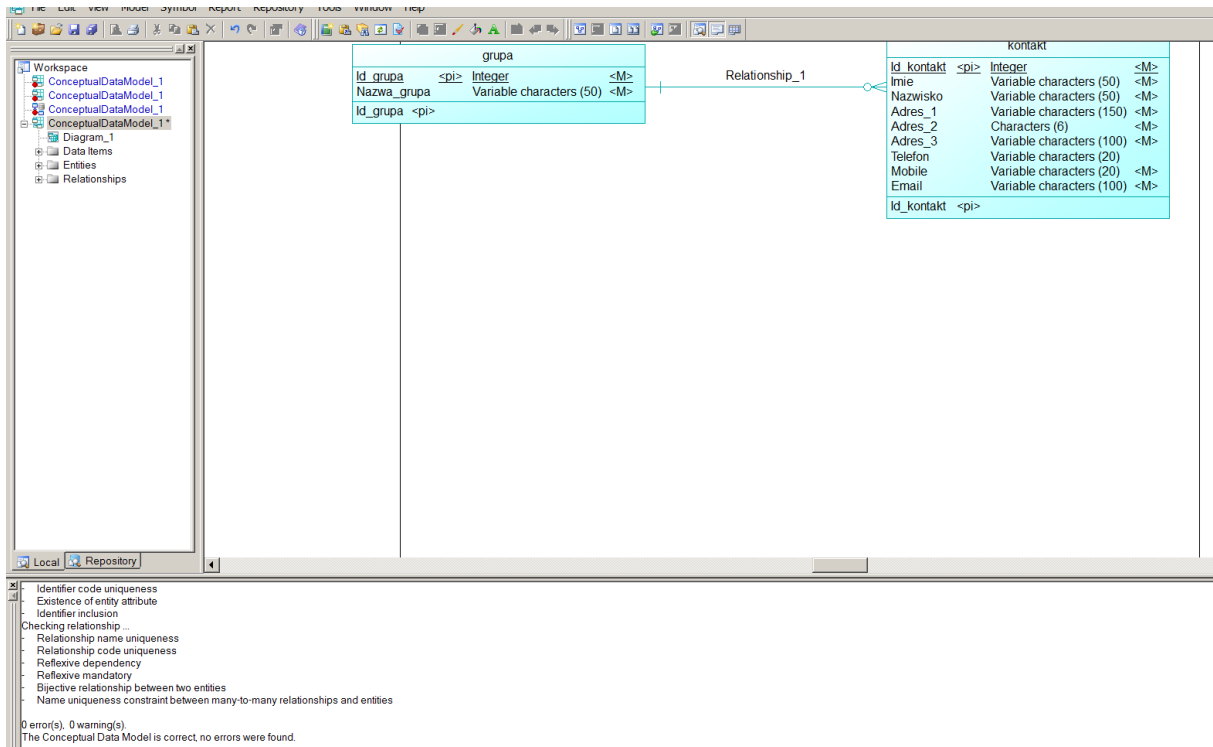


Dla domyślnych ustawień opcji sprawdzenia wybieramy **OK**.



Po sprawdzeniu powinno być 0 ostrzeżeń oraz 0 błędów.

Jeżeli u Państwa pojawią się jakieś ostrzeżenia lub błędy proszę to bezwzględnie zgłosić prowadzącemu!!!!



Na podstawie zaprojektowanego diagramu koncepcyjnego wygenerujemy teraz diagram fizyczny tej bazy.

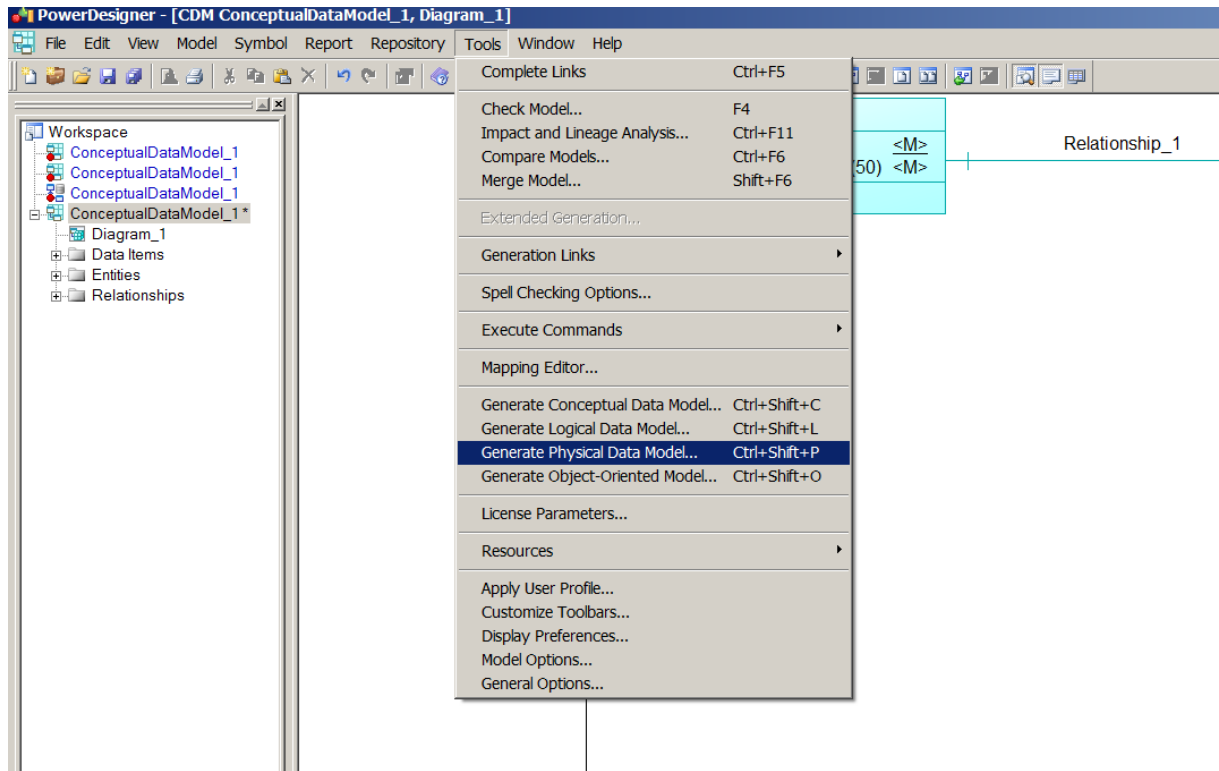
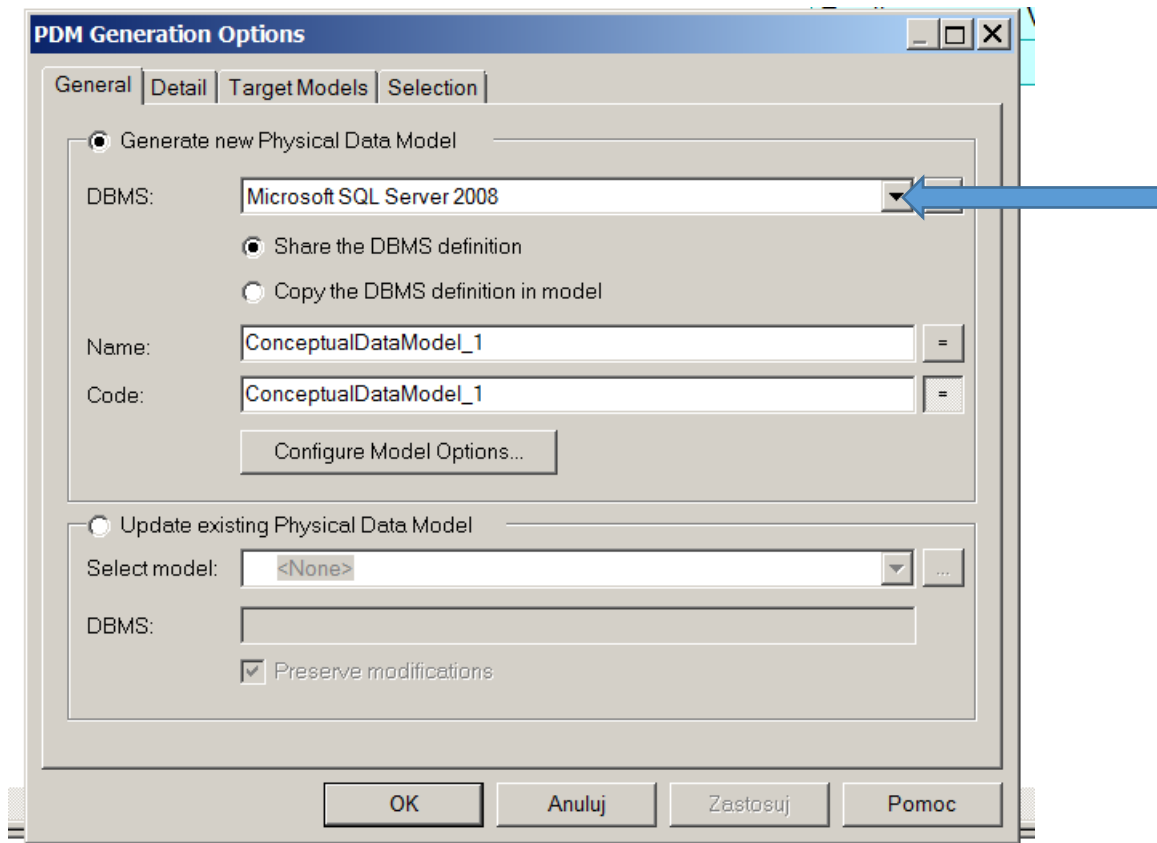


Diagram wygenerujemy dla wersji DBMS **Microsoft SQL Server 2008** (Data Base Management System – system zarządzania relacyjną bazą danych).

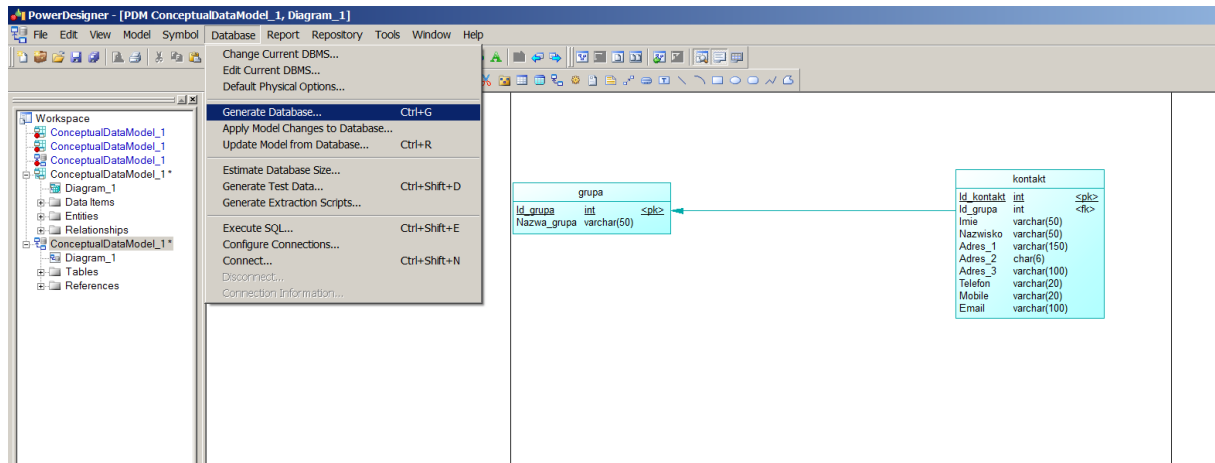


Wygenerowany diagram będzie wyglądał jak poniżej.

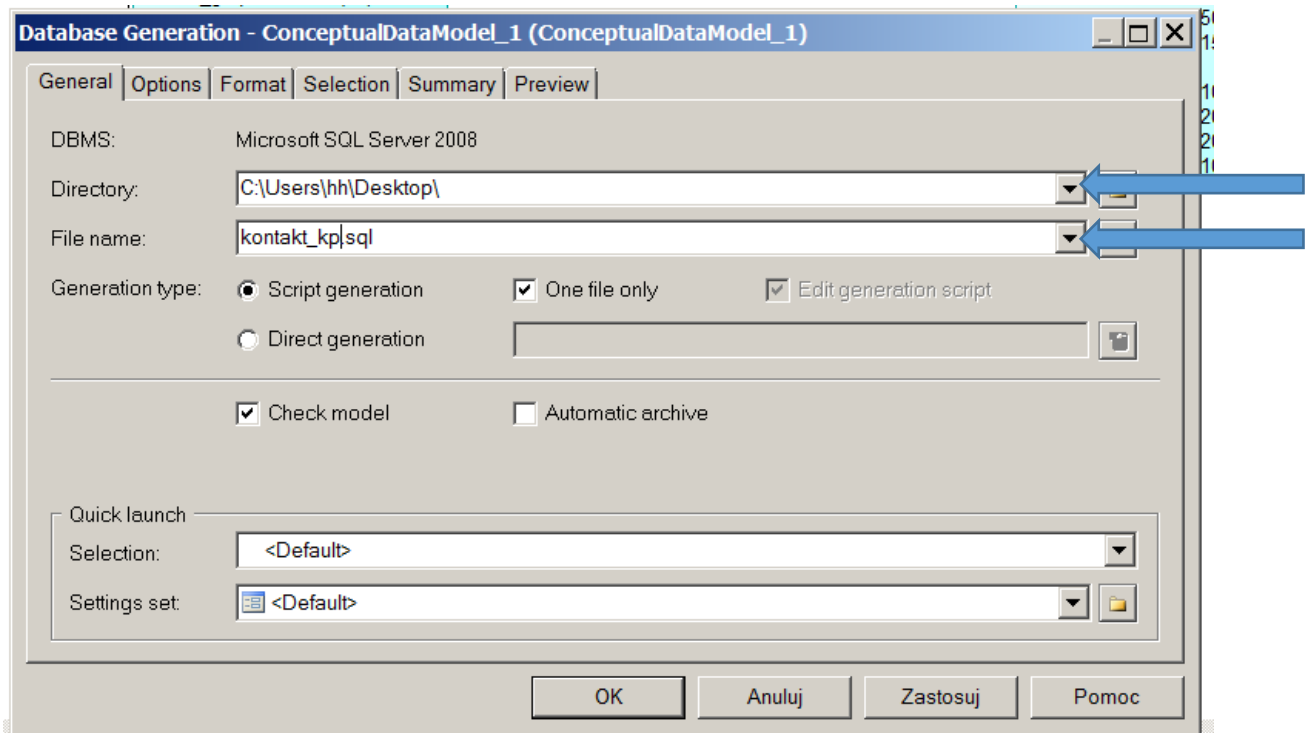
Do tabeli dodany został klucz obcy **Id_grupa** będąca wynikiem utworzonej relacji.



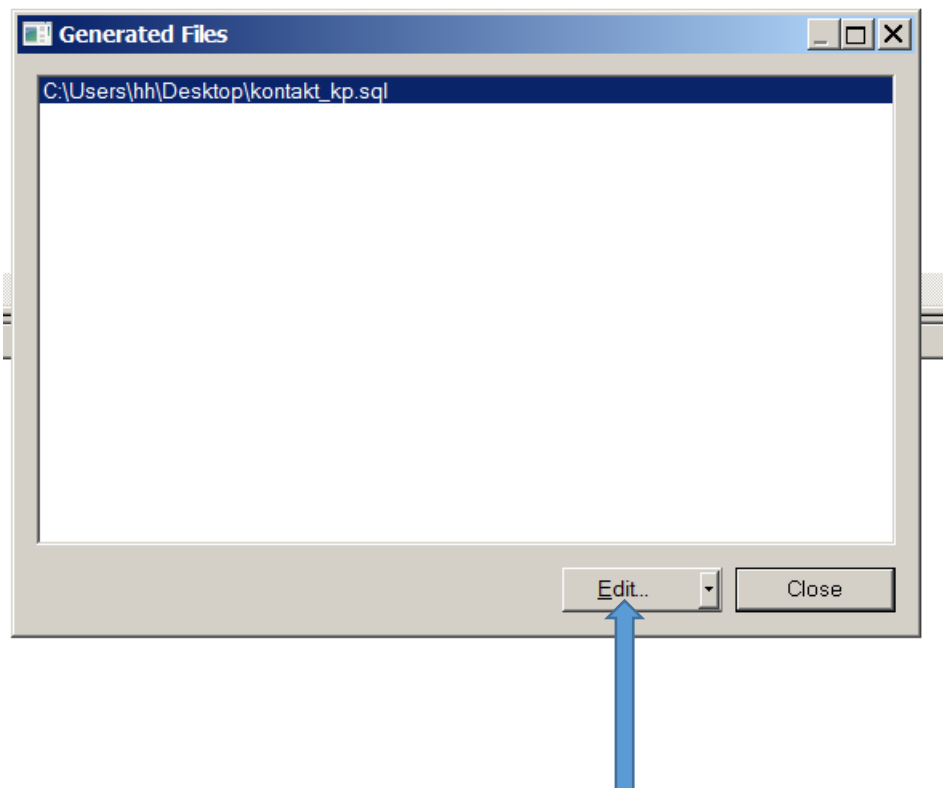
Na podstawie wygenerowanego modelu fizycznego utworzymy teraz skrypt SQL, który utworzy na serwerze strukturę bazy danych – table oraz atrybuty.



Proszę ustawić ścieżkę zapisu tak by wiadomo było gdzie plik zostanie zapisany oraz podanie jego nazwy.



Można otworzyć bezpośrednio plik wybierając jak poniżej.



Proszę zostawić otwarty ten plik będzie potrzebny na kolejnym etapie.

```
kontakt_kp — Notatnik
Plik Edycja Format Widok Pomoc
/*=====*/
/* DBMS name:      Microsoft SQL Server 2008      */
/* Created on:     2018-08-08 11:44:19           */
/*=====*/

if exists (select 1
           from sysobjects
           where id = object_id('grupa')
           and type = 'U')
drop table grupa
go

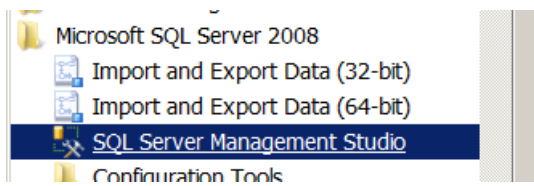
if exists (select 1
           from sysindexes
           where id = object_id('kontakt')
           and name = 'Relationship_1_FK'
           and indid > 0
           and indid < 255)
drop index kontakt.Relationship_1_FK
go

if exists (select 1
           from sysobjects
           where id = object_id('kontakt')
           and type = 'U')
drop table kontakt
go

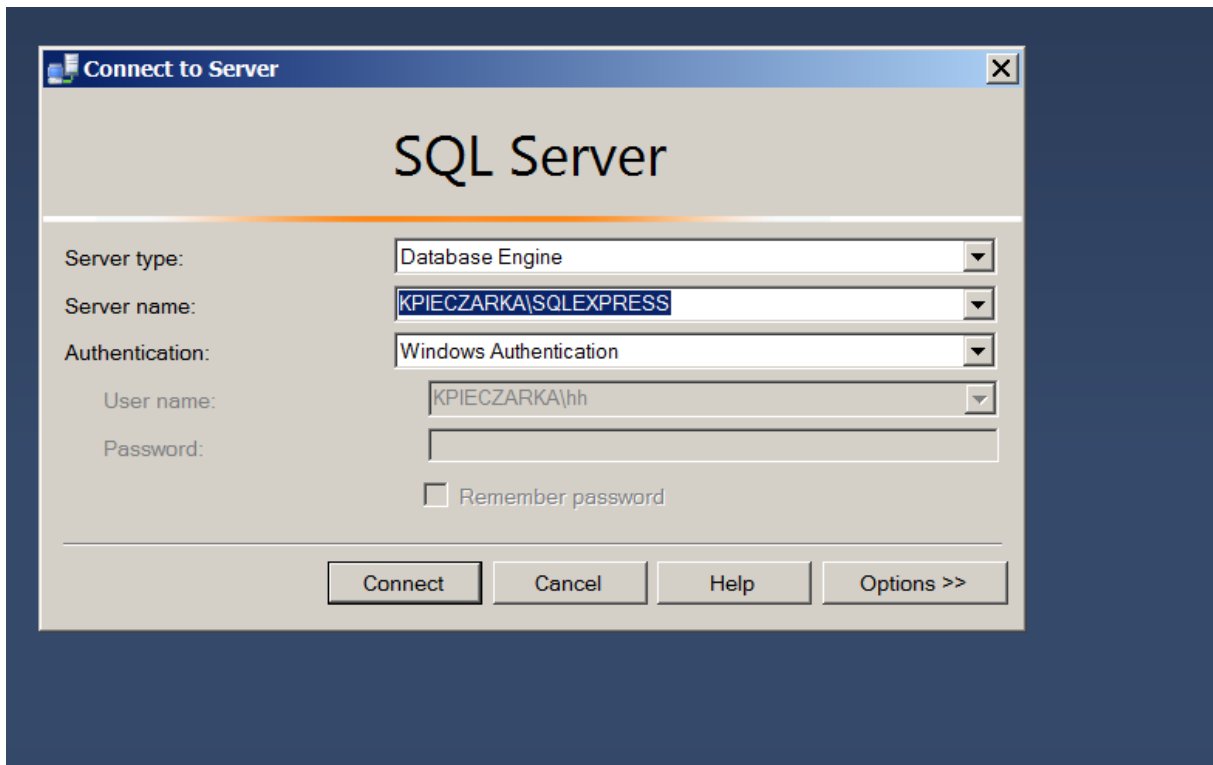
/*=====*/
/* Table: grupa                                     */
/*=====*/
create table grupa (
  Id_grupa          int          not null,
  Nazwa_grupa      varchar(50)  not null,
  constraint PK_GRUPA primary key nonclustered (Id_grupa)
)
go

/*=====*/
/* Table: kontakt                                 */
/*=====*/
create table kontakt (
  Id_kontakt       int          not null,
  Id_grupa          int          not null,
  Imie              varchar(50)  not null,
  Nazwisko          varchar(50)  not null,
  Adres_1           varchar(150) not null,
  Adres_2           char(6)      not null,
  Adres_3           varchar(100) not null,
  Telefon           varchar(20)  null,
  Mobile            varchar(20)  not null,
  Email             varchar(100) not null,
  constraint PK_KONTAKT primary key nonclustered (Id_kontakt)
)
go
```

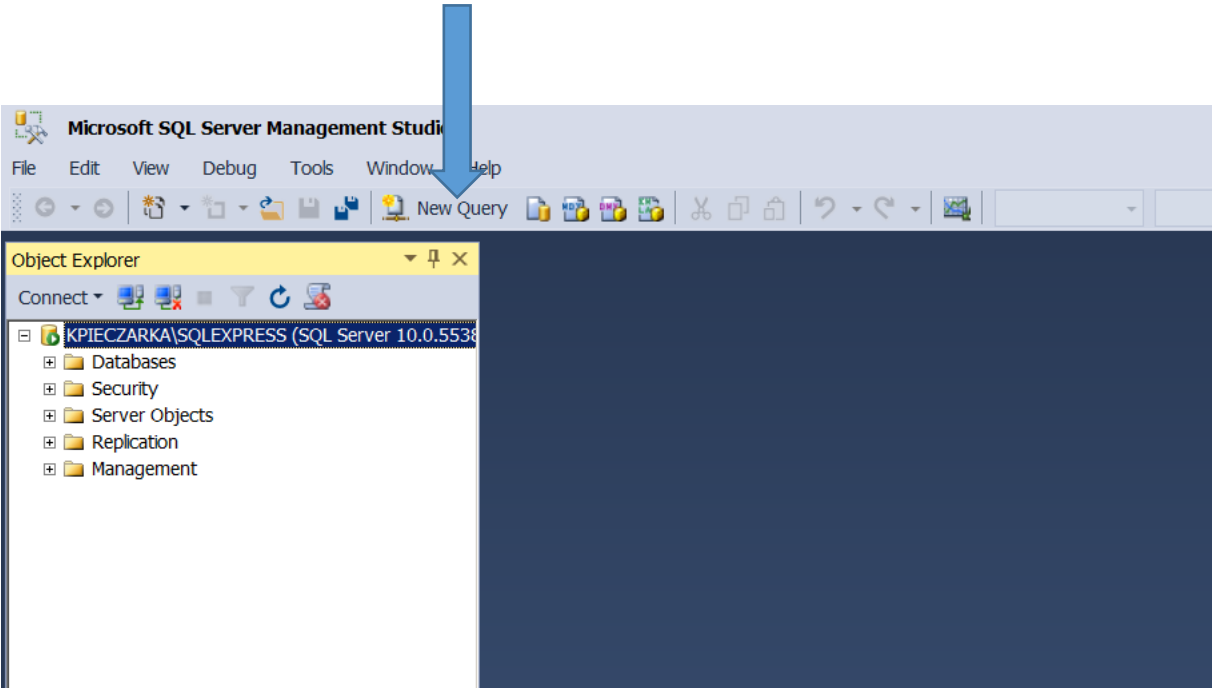
Uruchamiamy program za pomocą, którego możliwy będzie dostęp do serwera bazy danych.



Wybieramy **Connect**, nazwa serwera w Państwa przypadku będzie oczywiście inna, serwer jest zainstalowany lokalnie na komputerze, przy którym Państwo siedzicie.



Wywołujemy okno nowego zapytania do DBMS.

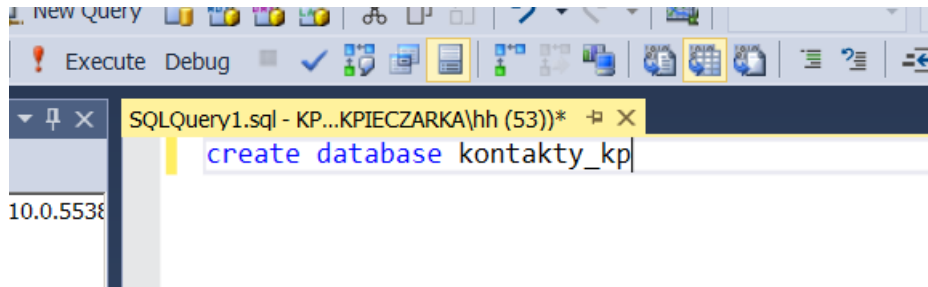


Utworzymy teraz „pojemnik” bazy danych, do tworzenia bazy danych czy tabeli wykorzystujemy klauzulę **CREATE**.

Wprowadzamy zatem zapytanie jak poniżej.

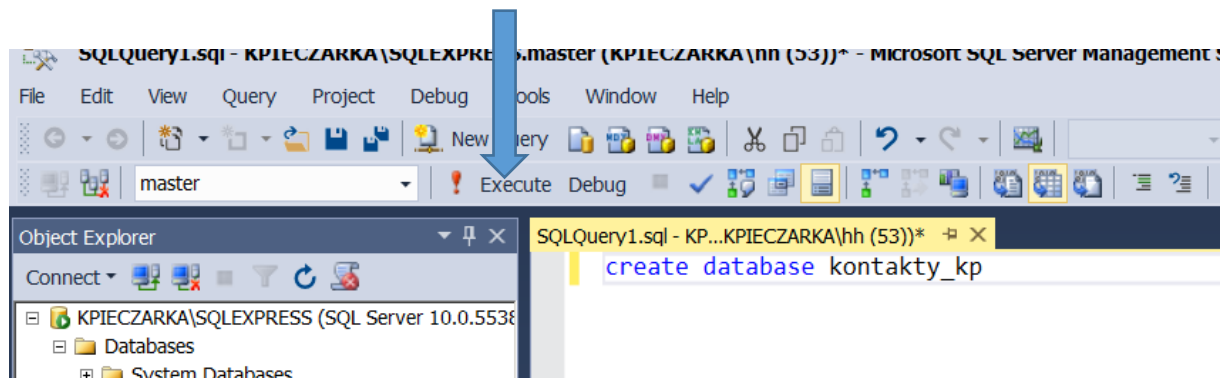
PROSZĘ ZASTOSOWAĆ UNIKATOWĄ NAZWĘ TWORZONEJ BAZY

NP *kontakty_Państwa inicjały*

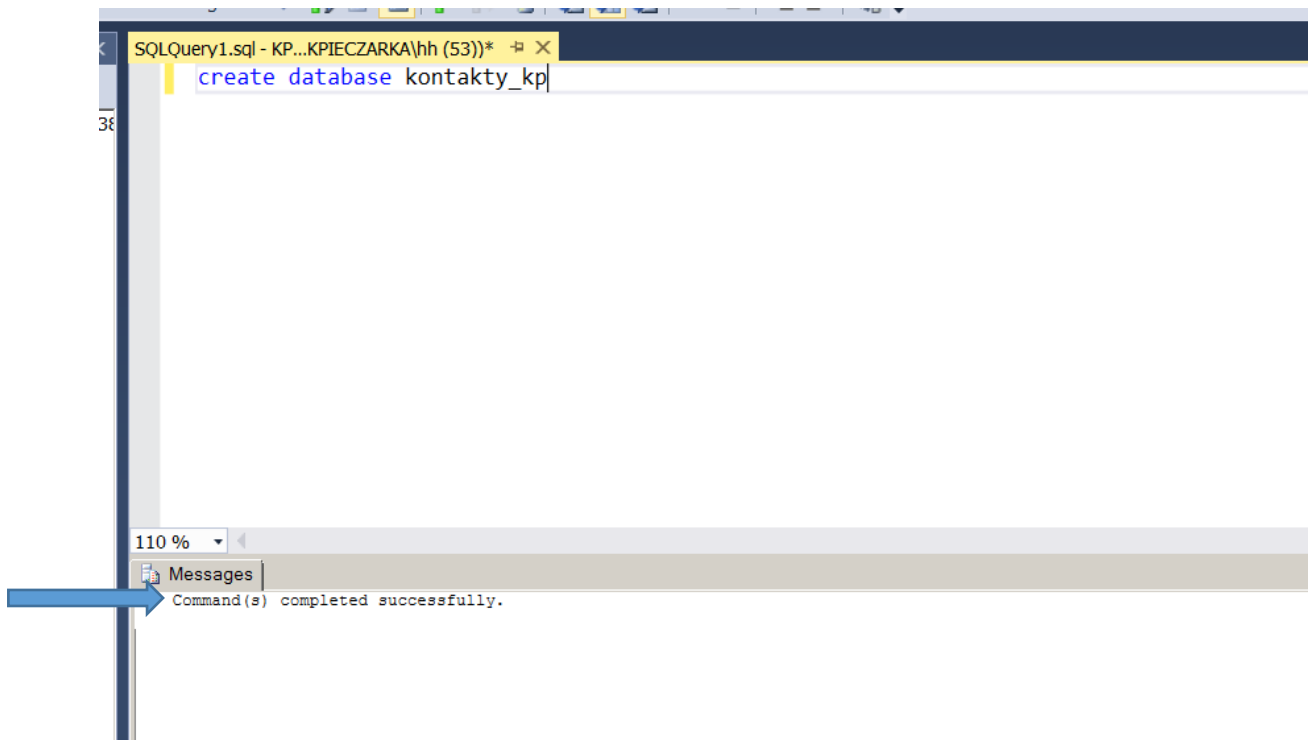


Następnie, wywołamy wykonanie tego zapytania

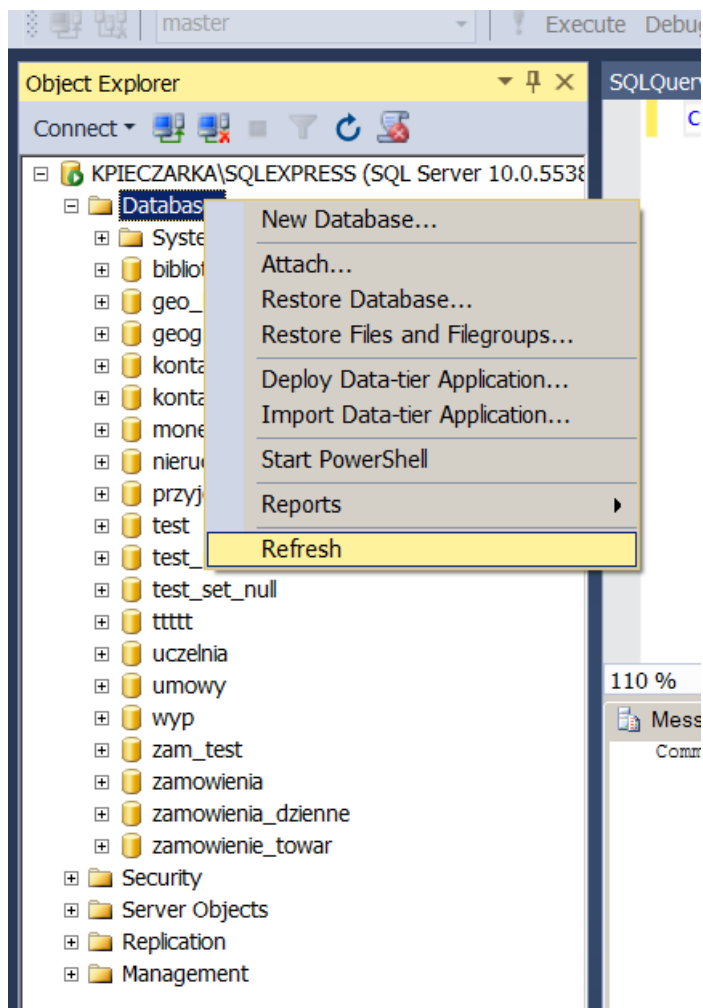
Skrót klawiszowy to **F5**



Jeżeli zapytanie zostanie wykonane otrzymamy potwierdzenie jak poniżej.

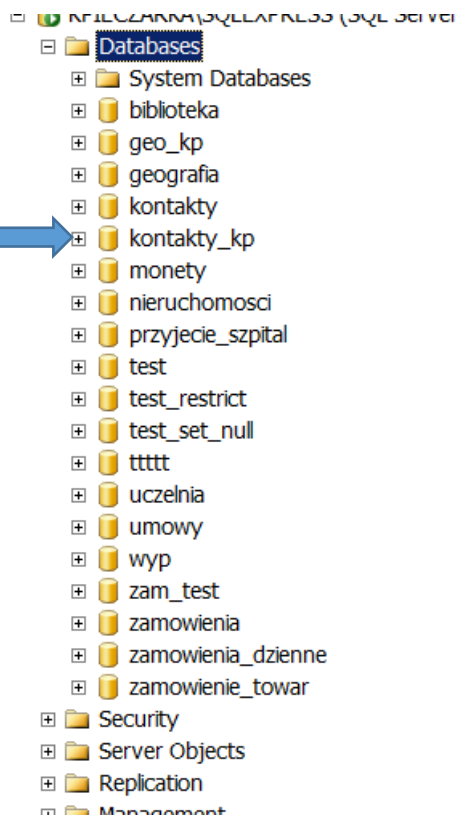


Po rozwinięciu listy baz danych utworzonych na serwerze i odświeżeniu jego zawartości



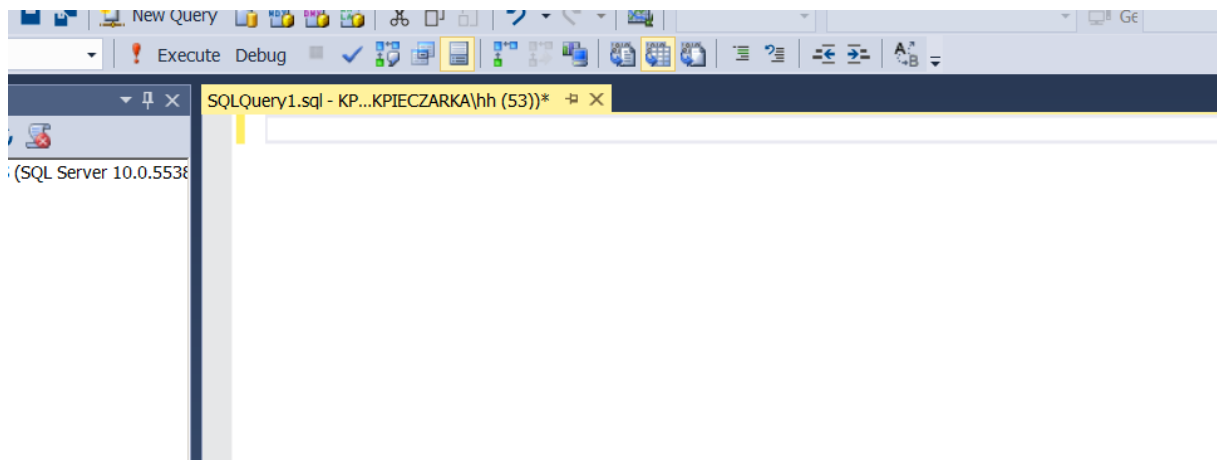
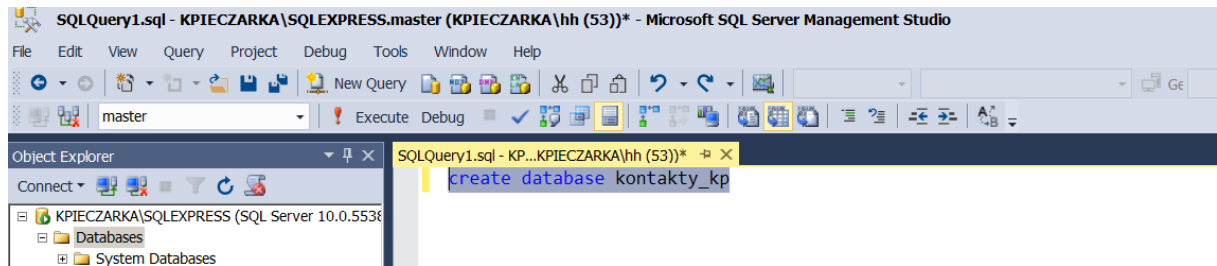
Znajdować się tam będzie utworzona baza.

Ilość baz danych może być u Państwa oczywiście inna niż poniżej.

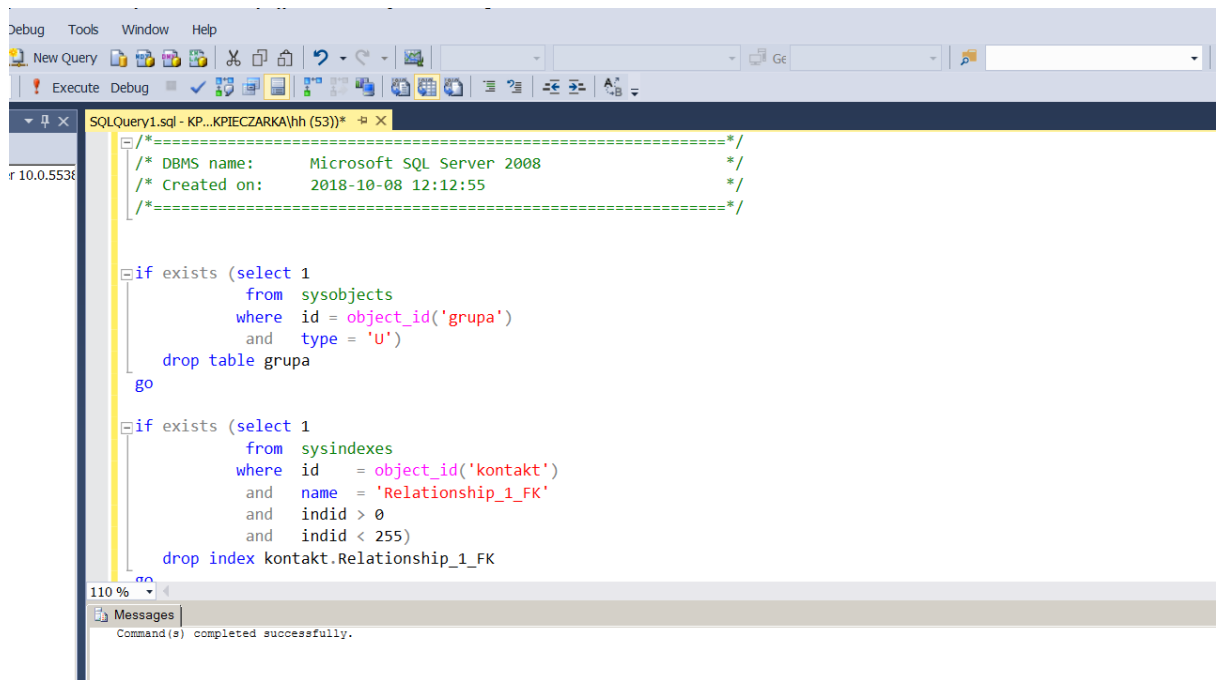


Za pomocą skryptu, który wygenerował program **PowerDesigner**, utworzymy teraz wewnątrz tej bazy table.

Usuamy poprzednie zapytanie



I wklejamy w całości zapytanie, które zostało wygenerowane przez program **PowerDesigner**.



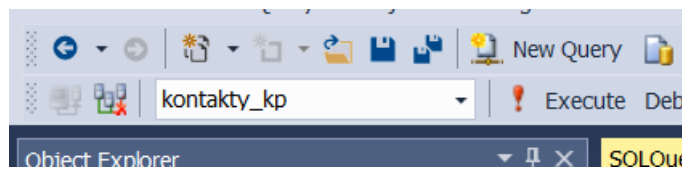
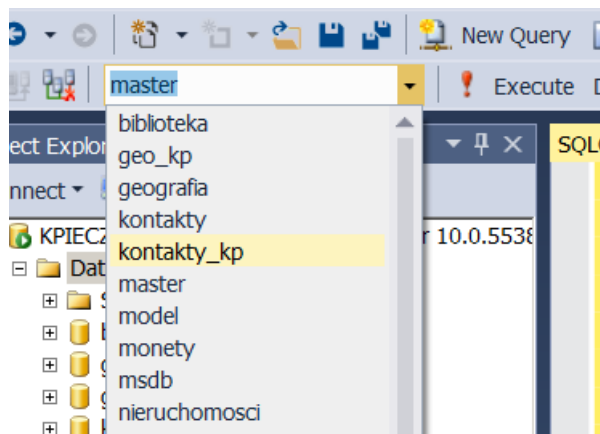
```
/*-----*/
/* DBMS name:      Microsoft SQL Server 2008      */
/* Created on:     2018-10-08 12:12:55            */
/*-----*/

if exists (select 1
           from sysobjects
           where id = object_id('grupa')
                 and type = 'U')
    drop table grupa
go

if exists (select 1
           from sysindexes
           where id = object_id('kontakt')
                 and name = 'Relationship_1_FK'
                 and indid > 0
                 and indid < 255)
    drop index kontakt.Relationship_1_FK
```

Messages
Command(s) completed successfully.

Musimy jeszcze wybrać, wewnątrz której bazy zapytanie to ma być wykonane.

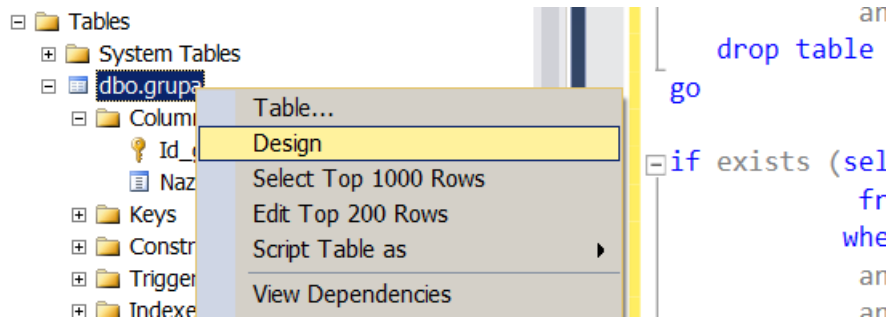


Wykonujemy zapytanie.

Po wykonaniu zapytania zostaną utworzone tabele oraz wewnątrz nich atrybuty jak poniżej.

- [-] kontakty_kp
 - [-] Database Diagrams
 - [-] Tables
 - [-] System Tables
 - [-] dbo.grupa
 - [-] Columns
 - Id_grupa (PK, int, not null)
 - Nazwa_grupa (varchar(50), not null)
 - [-] Keys
 - [-] Constraints
 - [-] Triggers
 - [-] Indexes
 - [-] Statistics
 - [-] dbo.kontakt
 - [-] Columns
 - Id_kontakt (PK, int, not null)
 - Id_grupa (int, not null)
 - Imie (varchar(50), not null)
 - Nazwisko (varchar(50), not null)
 - Adres_1 (varchar(150), not null)
 - Adres_2 (char(6), not null)
 - Adres_3 (varchar(150), not null)
 - Telefon (varchar(20), null)
 - Mobile (varchar(20), not null)
 - Email (varchar(150), not null)
 - [-] Keys

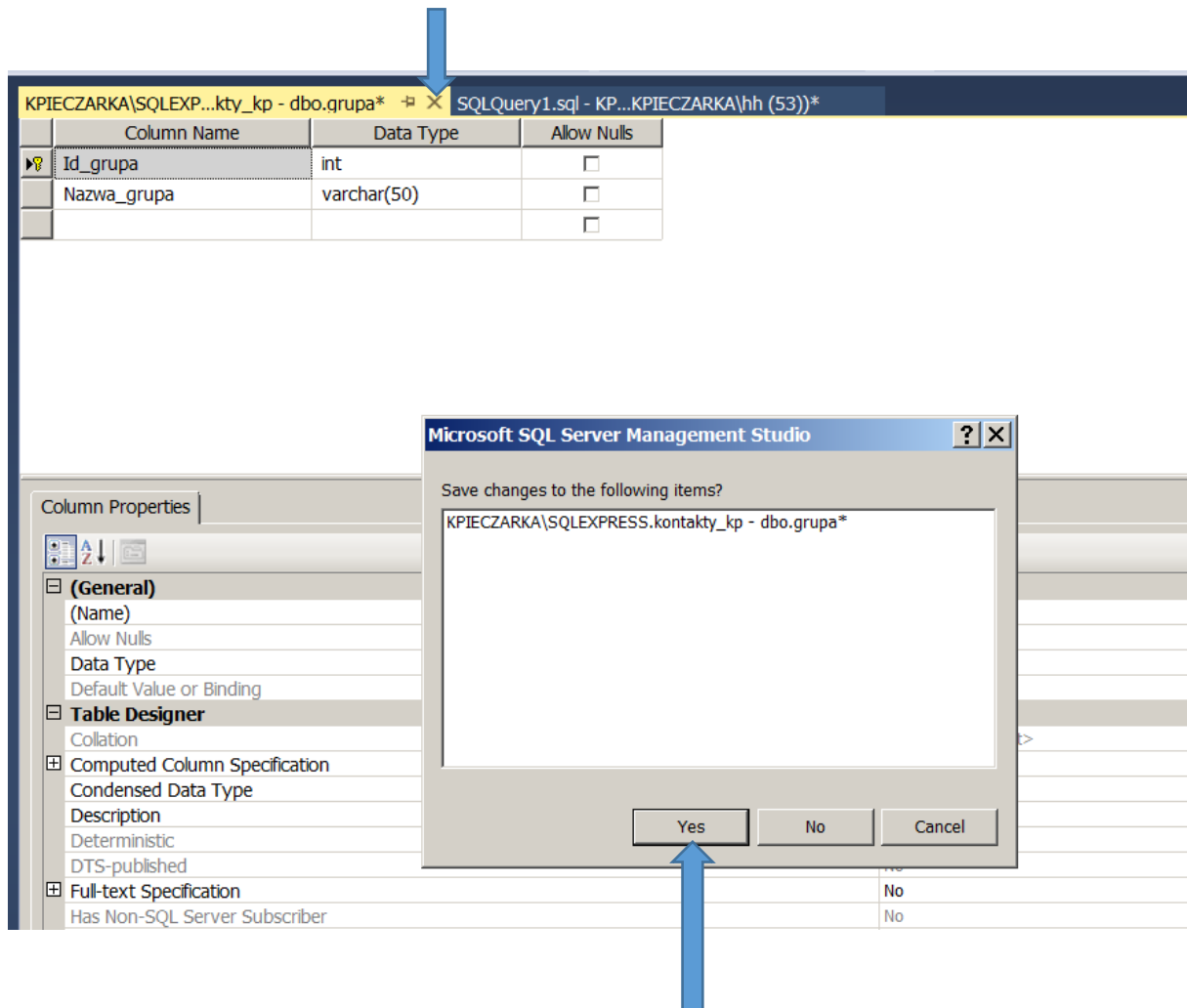
Aby baza działała w pełni funkcjonalnie wymagane jest jeszcze włączenie opcji autouzupelniania dla kluczy głównych tabel.



Column Name	Data Type	Allow Nulls
Id_grupa	int	<input type="checkbox"/>
Nazwa_grupa	varchar(50)	<input type="checkbox"/>

Column Properties	
(General)	
(Name)	Id_grupa
Allow Nulls	No
Data Type	int
Default Value or Binding	
Table Designer	
Collation	<database default>
Computed Column Specification	
Condensed Data Type	int
Description	
Deterministic	Yes
DTS-published	No
Full-text Specification	
Full-text Specification	No
Has Non-SQL Server Subscriber	No
Identity Specification	
(Is Identity)	Yes
Identity Increment	Yes
Identity Seed	No
Indexable	Yes
Is Columnset	No

Zamykamy to okno dialogowe i zapisujemy zmiany.



Analogicznie dla tabeli kontakt.

The screenshot displays the SQL Server Enterprise Manager interface. At the top, a table structure for 'kontakt' is shown with columns: id_kontakt (int), id_grupa (int), imie (varchar(50)), nazwisko (varchar(50)), adres_1 (varchar(150)), adres_2 (char(6)), adres_3 (varchar(150)), telefon (varchar(20)), mobile (varchar(20)), and email (varchar(150)). The 'telefon' column has the 'Allow Nulls' checkbox checked.

Below the table structure, the 'Column Properties' window is open for the 'id_kontakt' column. The 'Data Type' is 'int'. Under the 'Identity Specification' section, the '(Is Identity)' checkbox is checked, and the 'Identity Increment' is set to 1.

Column Name	Data Type	Allow Nulls
id_kontakt	int	<input type="checkbox"/>
id_grupa	int	<input type="checkbox"/>
imie	varchar(50)	<input type="checkbox"/>
Nazwisko	varchar(50)	<input type="checkbox"/>
Adres_1	varchar(150)	<input type="checkbox"/>
Adres_2	char(6)	<input type="checkbox"/>
Adres_3	varchar(150)	<input type="checkbox"/>
Telefon	varchar(20)	<input checked="" type="checkbox"/>
Mobile	varchar(20)	<input type="checkbox"/>
Email	varchar(150)	<input type="checkbox"/>

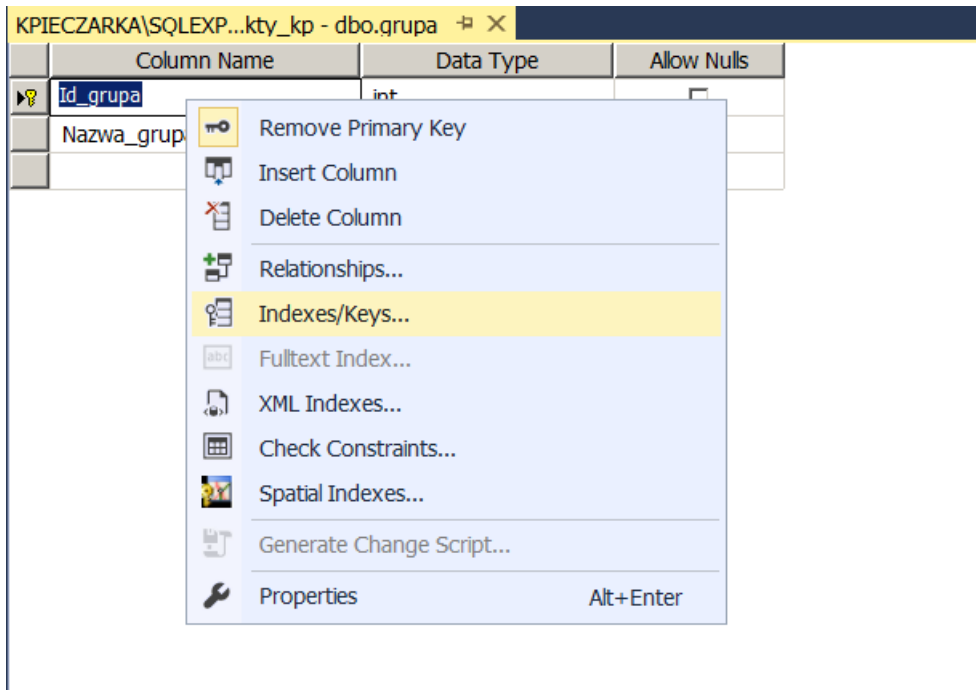
Column Properties

Data Type	int
Default Value or Binding	
Table Designer	
Colation	<database default>
Computed Column Specification	
Condensed Data Type	int
Description	
Deterministic	Yes
DTS-published	No
Full-text Specification	No
Has Non-SQL Server Subscriber	No
Identity Specification	Yes
(Is Identity)	Yes
Identity Increment	1
Identity Seed	1
Indexable	Yes
Is Columnset	No
Is Sparse	No

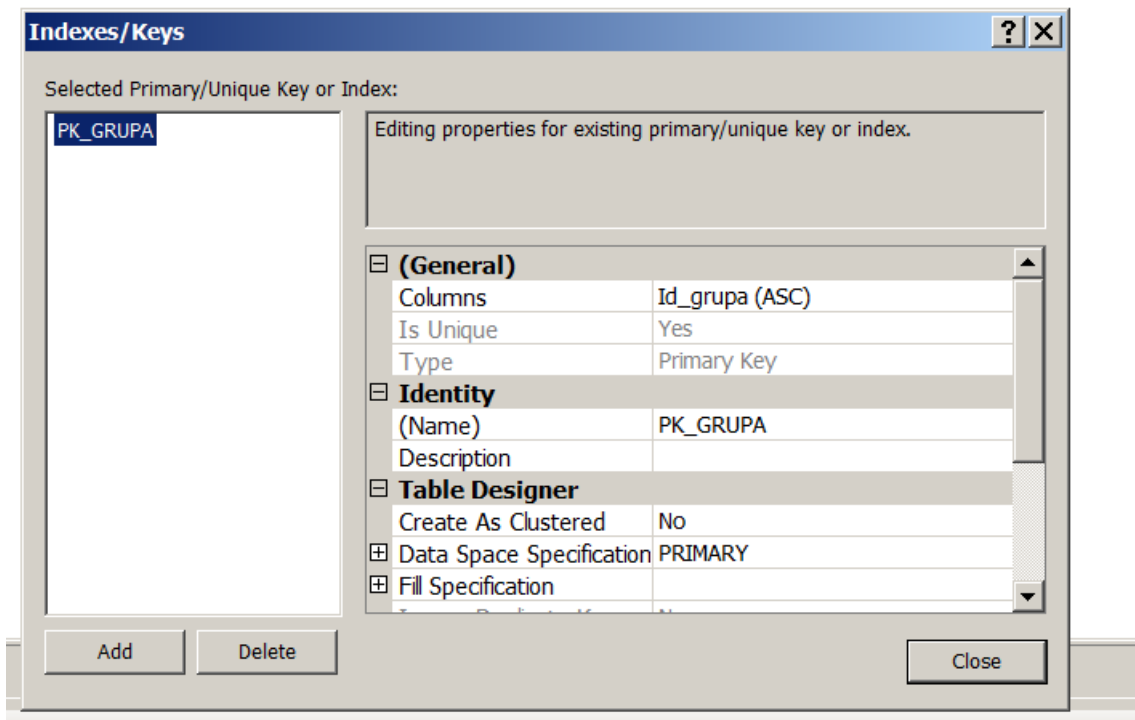
Z uwagi na fakt, że tabela grupa jest tabelą słownikową ważne jest by wartości w ramach atrybutu **Nazwa_grupa** nie powtarzały się. Zabronimy zatem na duplikaty w ramach tego atrybutu.

Klikamy zatem na nazwę tabeli **PP** i wybieramy **Design**.

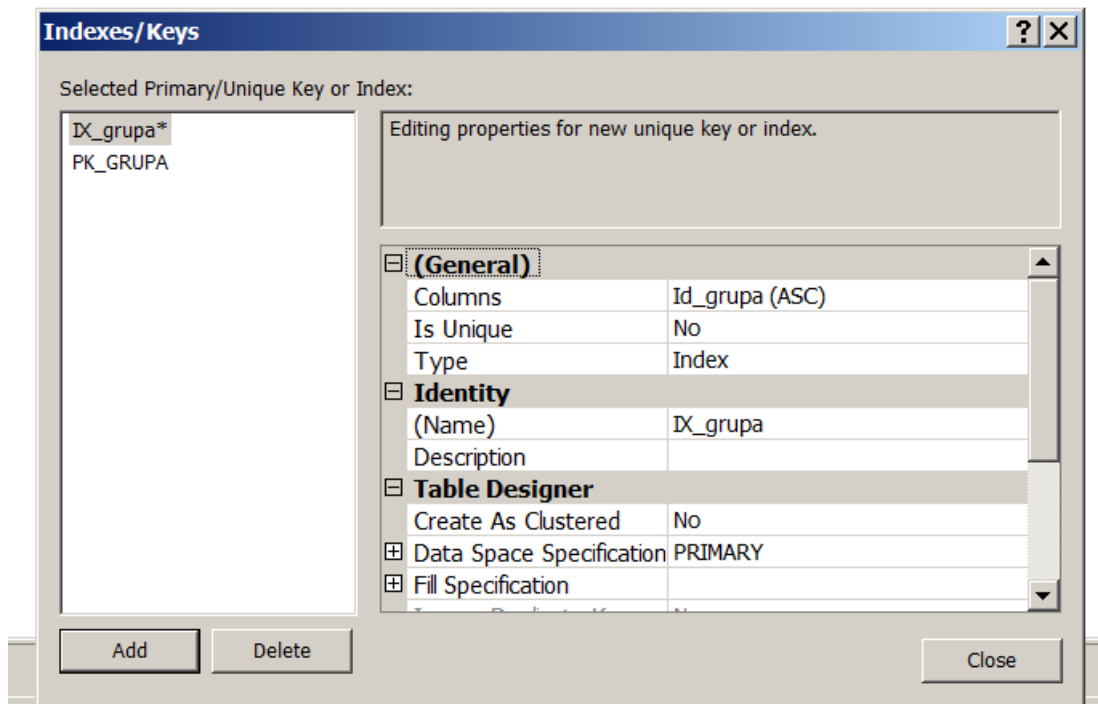
Następnie w dowolną nazwę atrybutu **PP** i wybieramy jak poniżej.



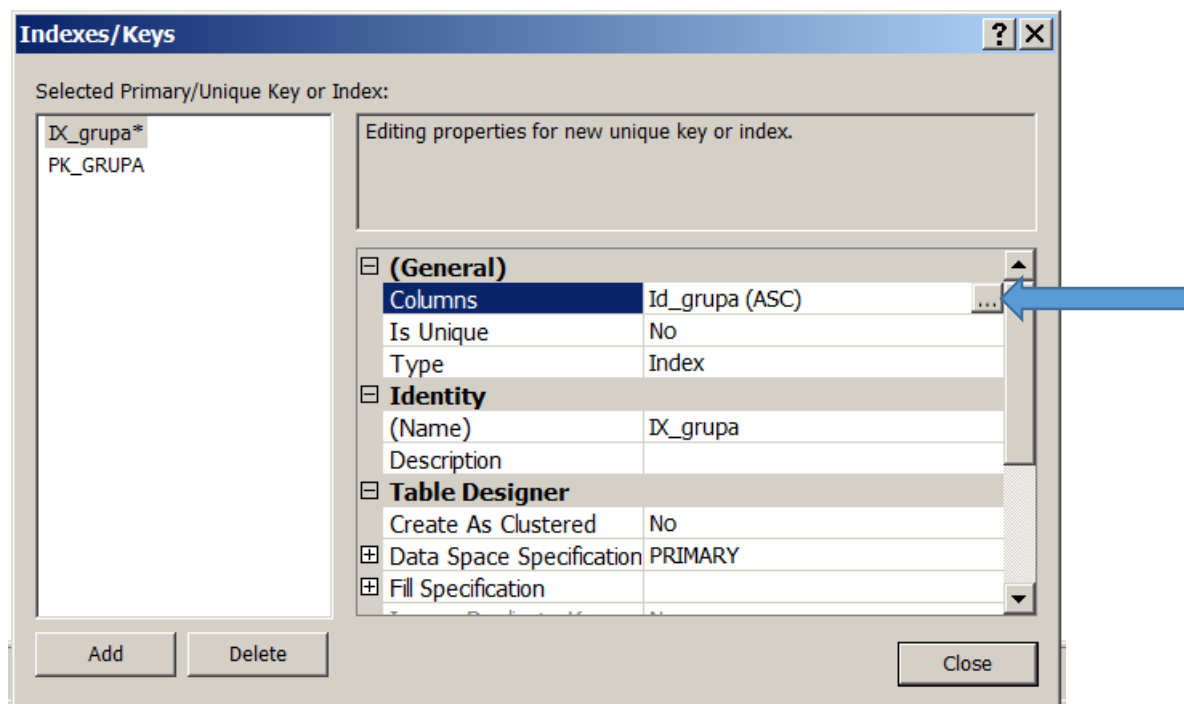
Następnie wybieramy **Add**



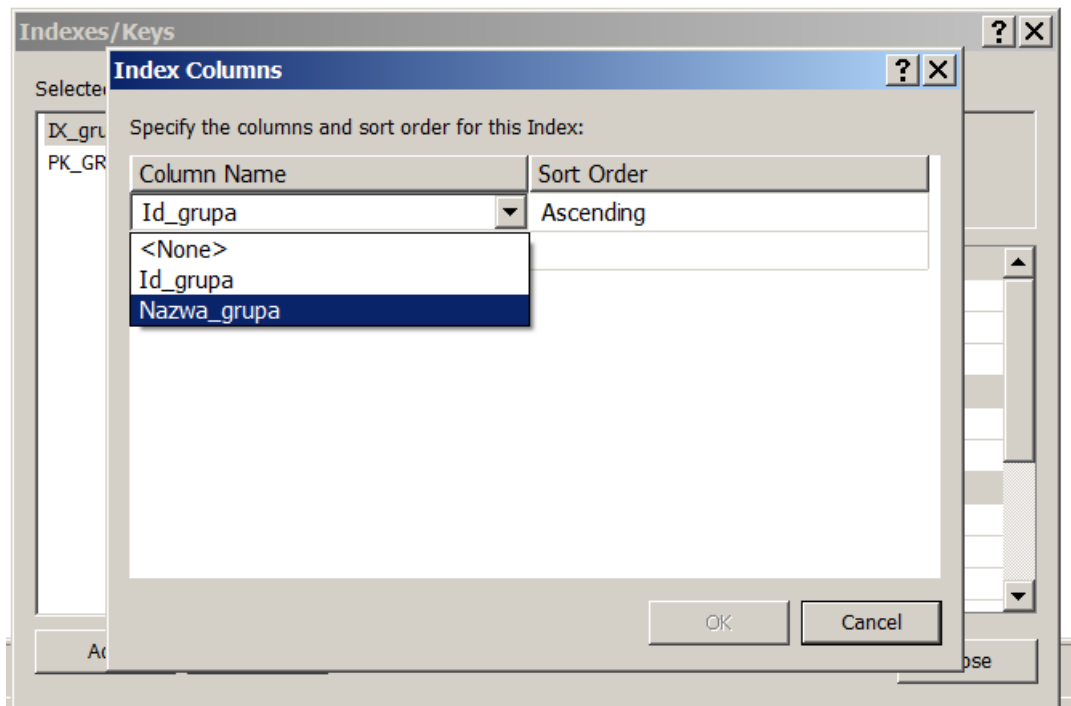
Zostanie dodany nowy Index



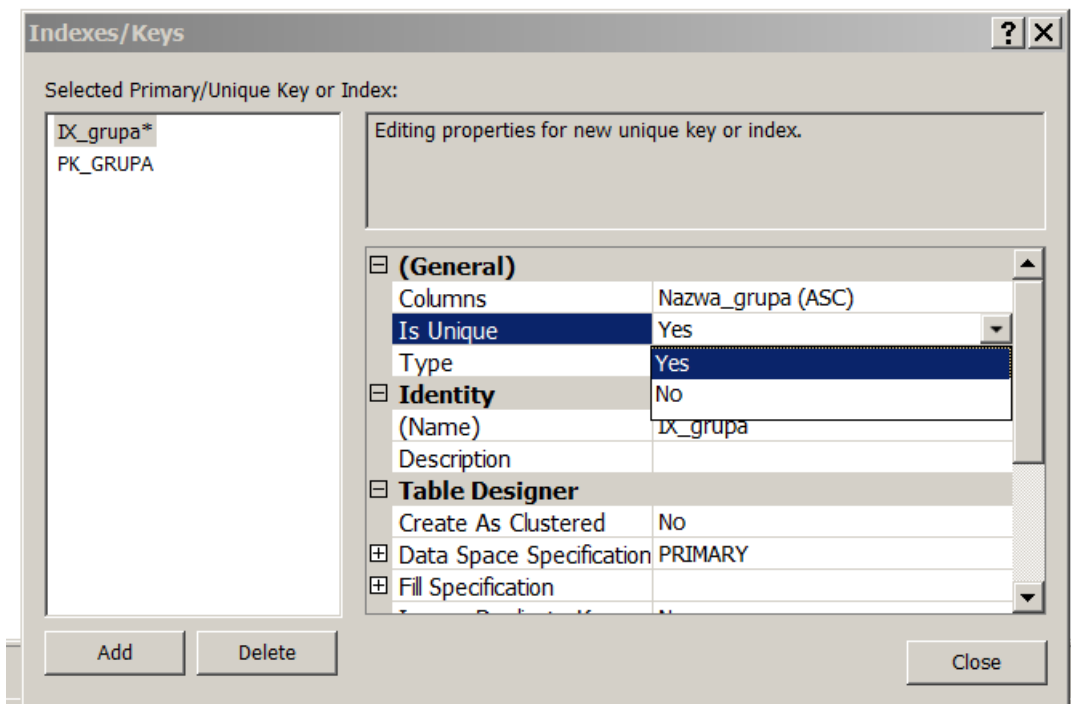
Wybieramy dla którego atrybutu tabeli ma być ustanowiony



Wybieramy zatem jak poniżej i akceptujemy **OK**



Następnie opcję **Is Unique** ustawiamy na **Yes**



Zamykamy okno dialogowe i zapisujemy zmiany w tabeli.

Od tego momentu nie będzie możliwe dodanie dwóch grup kontaktu o tej samej nazwie.

Analogicznie proszę postąpić w tabeli kontakt dla atrybutów mobile oraz email.

Baza jest już w pełni funkcjonalna.

Rozpoczniemy od wprowadzenia danych do tabeli grupa.

W języku SQL do wprowadzania danych wykorzystujemy klauzulę **INSERT**.

Zapytanie do wprowadzania danych ma następującą postać:

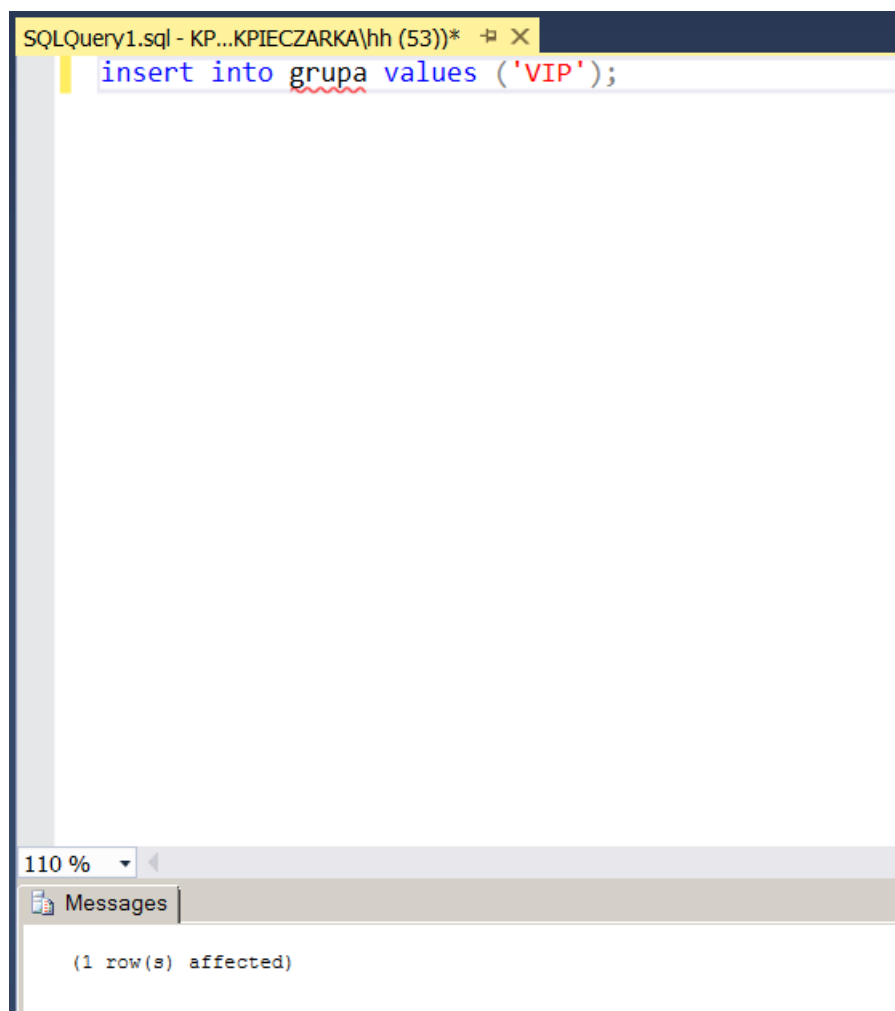
```
insert into nazwa_tabeli values ('atrybut_1', 'atrybut_2', 'atrybut_3', 'atrybut_4');
```

Tabela **grupa** posiada dwa atrybuty **Id_grupa** oraz **Nazwa_grupa**.

Atrybut **Id_grupa** jest atrybutem kluczowym i ma ustawioną opcję autouzupelniania, zatem będzie wypełniany automatycznie przez DBMS.

Powiedzmy, że chcemy dodać grupę VIP, zapytanie zatem będzie miało postać jak poniżej:

Wykonujemy to zapytanie.



The screenshot shows a SQL query editor window titled "SQLQuery1.sql - KP...KPIECZARKA\hh (53))*". The query text is "insert into grupa values ('VIP');". Below the query editor, there is a "Messages" pane showing the result "(1 row(s) affected)".

```
SQLQuery1.sql - KP...KPIECZARKA\hh (53))* X
```

```
insert into grupa values ('VIP');
```

110 %

Messages

(1 row(s) affected)

Modyfikując to zapytanie proszę dodać następujące grupy:

szkoła, rodzina, znajomi, praca.

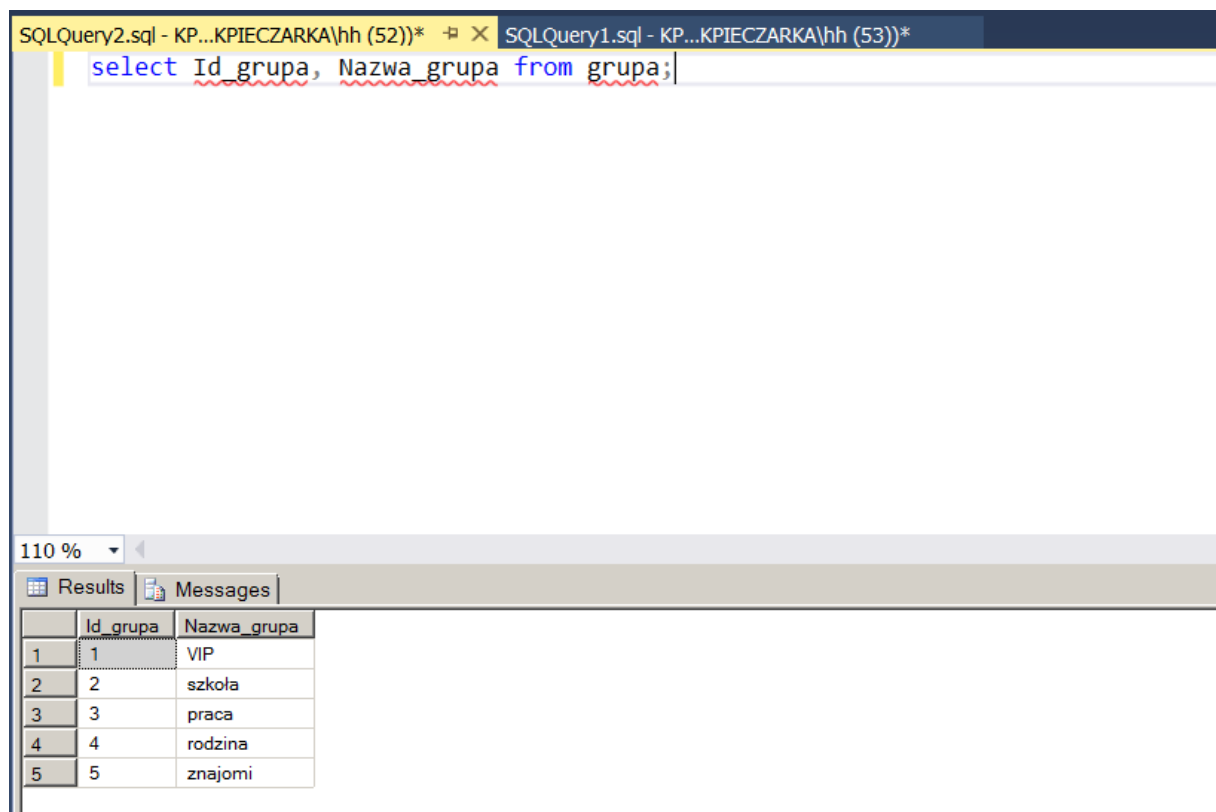
Zobaczmy czy dane zostały dodane.

Do pobierania danych z bazy wykorzystujemy zapytania **SELECT**.

Struktura zapytania pobierającego dane jest następująca:

select atrybut_1, atrybut_2, atrybut_3,.... from nazwa_tabeli

Zatem w naszym przypadku



The screenshot shows a SQL Server Enterprise Manager interface. At the top, there are two tabs: 'SQLQuery2.sql - KP...KPIECZARKA\hh (52))*' and 'SQLQuery1.sql - KP...KPIECZARKA\hh (53))*'. The active window displays the following SQL query:

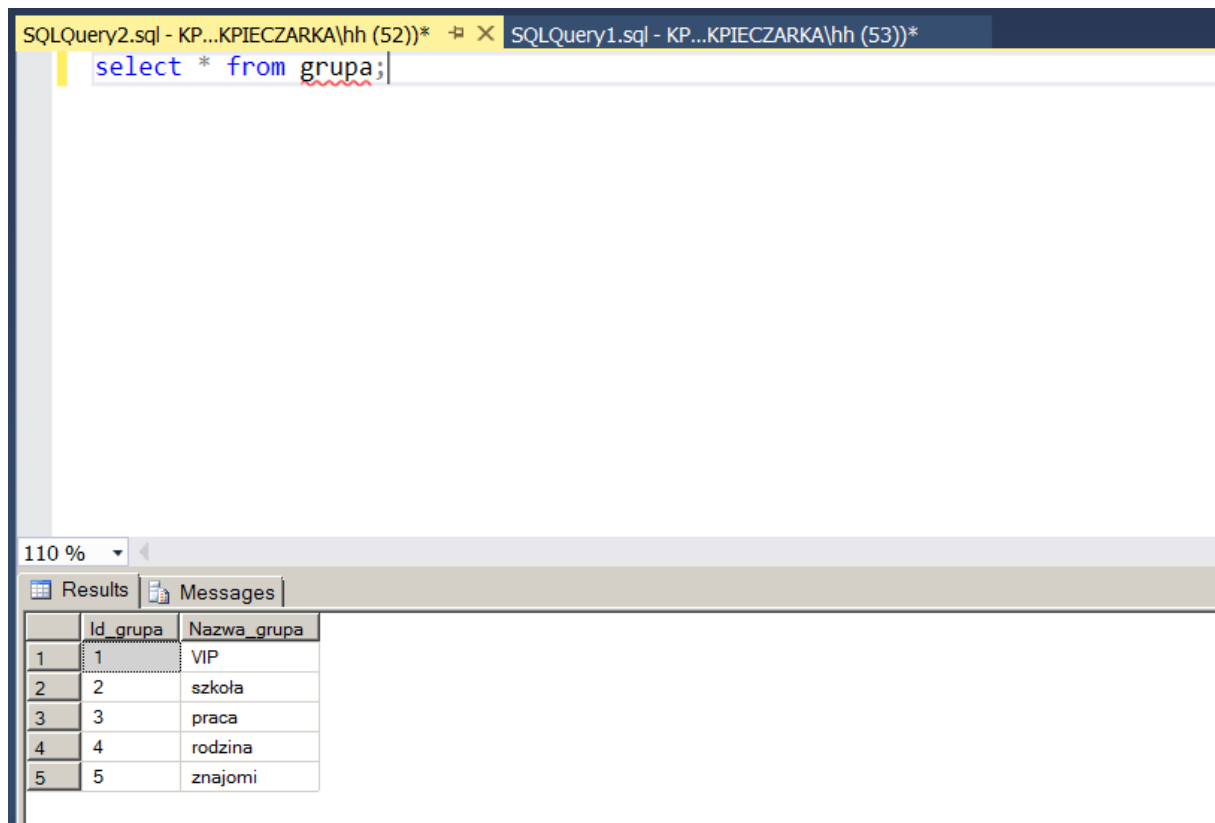
```
select Id_grupa, Nazwa_grupa from grupa;
```

Below the query window, there is a 'Results' pane showing the output of the query. The results are displayed in a table with two columns: 'Id_grupa' and 'Nazwa_grupa'. The data is as follows:

	Id_grupa	Nazwa_grupa
1	1	VIP
2	2	szkoła
3	3	praca
4	4	rodzina
5	5	znajomi

Kolejność poszczególnych grup w tabeli jak również wartości klucza jakie są przydzielone do poszczególnych grup nie mają wpływu na prawidłowe działanie bazy – jeżeli w Państwa przypadku wygląda to inaczej niż powyżej również jest prawidłowo.

Jeżeli pobieramy wszystkie atrybuty tabeli zamiast je wymieniać możemy użyć symbolu: *



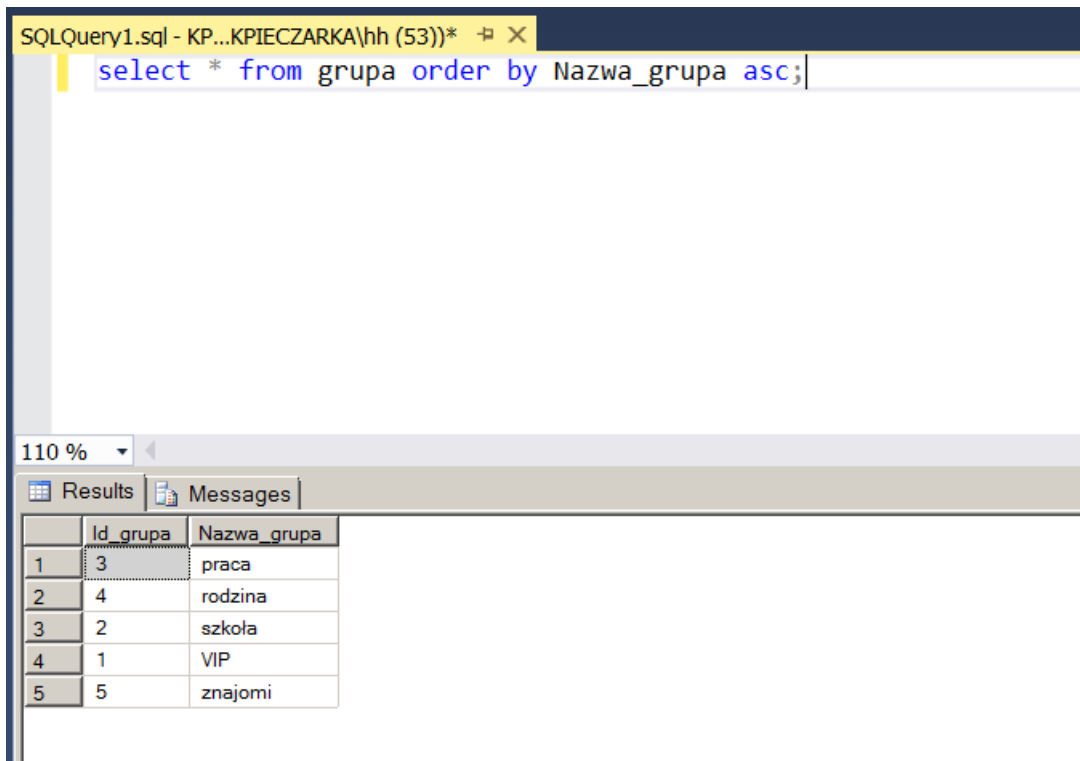
The screenshot shows a SQL Server Enterprise Manager interface. At the top, there are two tabs: 'SQLQuery2.sql - KP...KPIECZARKA\hh (52))*' and 'SQLQuery1.sql - KP...KPIECZARKA\hh (53))*'. The active query window contains the SQL statement: `select * from grupa;`. Below the query window, the 'Results' pane is visible, showing a table with 5 rows and 3 columns: 'Id_grupa', 'Nazwa_grupa', and an unlabeled column. The data is as follows:

	Id_grupa	Nazwa_grupa
1	1	VIP
2	2	szkola
3	3	praca
4	4	rodzina
5	5	znajomi

Kolejność danych odpowiada kolejności wprowadzania.

Jeżeli chcemy posortować dane do zapytania **select** należy dodać klauzulę **order by**.

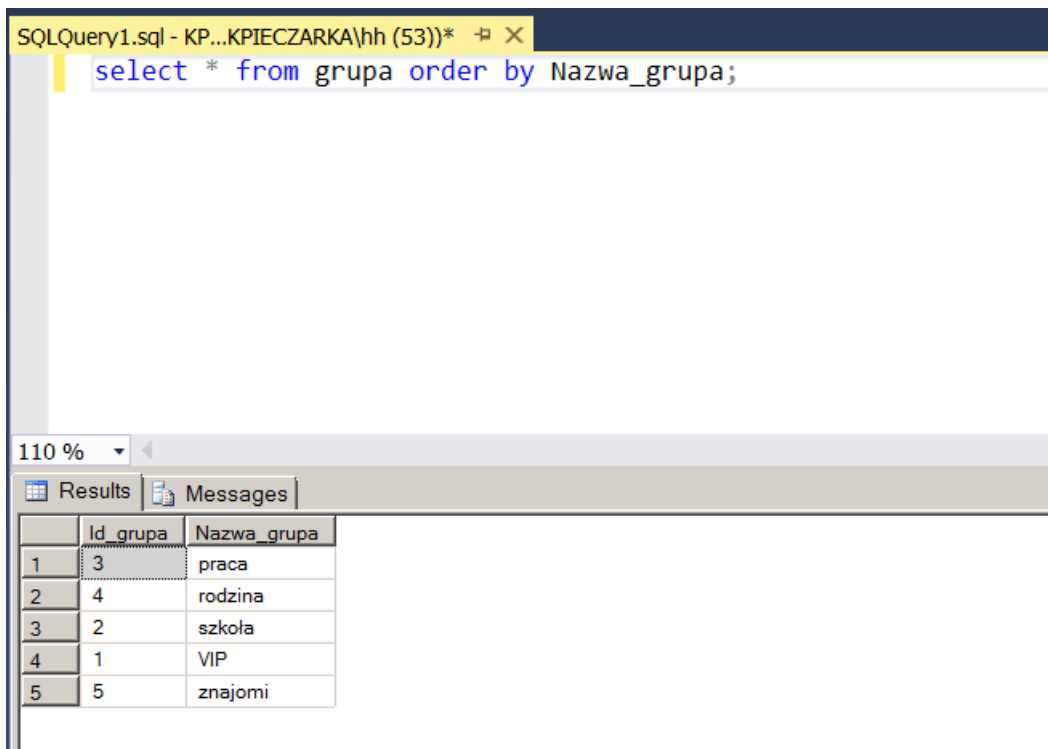
Sortowanie rosnąco A-Z, dodajemy klauzulę **asc**.



The screenshot shows a SQL query editor window titled "SQLQuery1.sql - KP...KPIECZARKA\hh (53))*". The query text is `select * from grupa order by Nazwa_grupa asc;`. Below the editor, the "Results" pane shows a table with 5 rows and 3 columns: "Id_grupa", "Nazwa_grupa", and an unlabeled column. The data is as follows:

	Id_grupa	Nazwa_grupa
1	3	praca
2	4	rodzina
3	2	szkoła
4	1	VIP
5	5	znajomi

Jeżeli sortujemy rosnąco nie jest wymagane dodawanie **asc**, jest to wartość domyślana.



The screenshot shows a SQL query editor window titled "SQLQuery1.sql - KP...KPIECZARKA\hh (53))*". The query text is `select * from grupa order by Nazwa_grupa;`. Below the editor, the "Results" pane shows a table with 5 rows and 3 columns: "Id_grupa", "Nazwa_grupa", and an unlabeled column. The data is as follows:

	Id_grupa	Nazwa_grupa
1	3	praca
2	4	rodzina
3	2	szkoła
4	1	VIP
5	5	znajomi

Sortowanie malejąco Z-A, dodajemy klauzulę **desc**.

The screenshot shows a SQL query window with the following text:

```
SQLQuery1.sql - KP...KPIECZARKA\hh (53))* [X]  
select * from grupa order by Nazwa_grupa desc;
```

Below the query editor, the 'Results' tab is active, displaying a table with 5 rows and 3 columns: 'Id_grupa', 'Nazwa_grupa', and an unlabeled column. The data is sorted in descending order by 'Nazwa_grupa'.

	Id_grupa	Nazwa_grupa
1	5	znajomi
2	1	VIP
3	2	szkoła
4	4	rodzina
5	3	praca

Kolejna operacja, którą możemy wykonać na danych to ich modyfikowanie, do tego wykorzystujemy zapytania **UPDATE**.

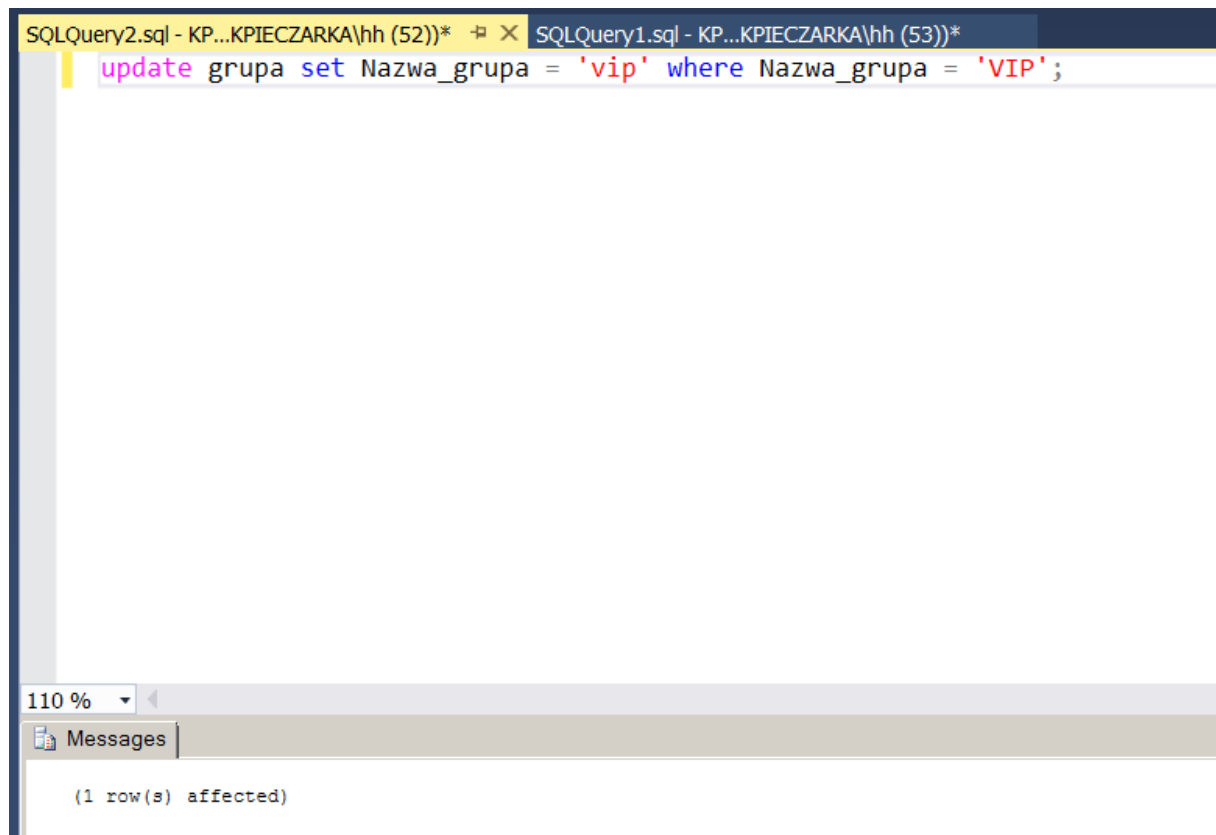
Zapytanie modyfikujące dane ma następującą strukturę:

update nazwa_tabeli set nazwa_atrybutu = 'wartość na którą zmieniamy'

Jednak, jeżeli zostało by wykonane zapytanie jak powyżej w całej tabeli zmienione by były wszystkie wartości dla tego atrybut.

Jeżeli chcemy zmienić wybraną określoną wartość, to wymagane jest zdefiniowanie warunku by wskazać ten konkretny za pomocą klauzuli **WHERE**.

Załóżmy, że w naszym przypadku chcemy zmienić nazwę grupy z **VIP** na **vip**, zapytanie takie miało by wówczas postać:



```
SQLQuery2.sql - KP...KPIECZARKA\hh (52))*  SQLQuery1.sql - KP...KPIECZARKA\hh (53))*
update grupa set Nazwa_grupa = 'vip' where Nazwa_grupa = 'VIP';
```

110 %

Messages

(1 row(s) affected)

Wóczas

The screenshot shows a SQL query execution window with two tabs: 'SQLQuery2.sql - KP...KPIECZARKA\hh (52))*' and 'SQLQuery1.sql - KP...KPIECZARKA\hh (53))*'. The active tab contains the query: `select * from grupa order by Nazwa_grupa;`. Below the query editor, the 'Results' tab is selected, displaying a table with 5 rows and 3 columns: 'Id_grupa', 'Nazwa_grupa', and an unlabeled column. The results are as follows:

	Id_grupa	Nazwa_grupa
1	3	praca
2	4	rodzina
3	2	szkoła
4	1	vip
5	5	znajomi

Kolejna operacja którą można wykonać na danych to ich usuwanie, do tego służą zapytania **DELETE**.

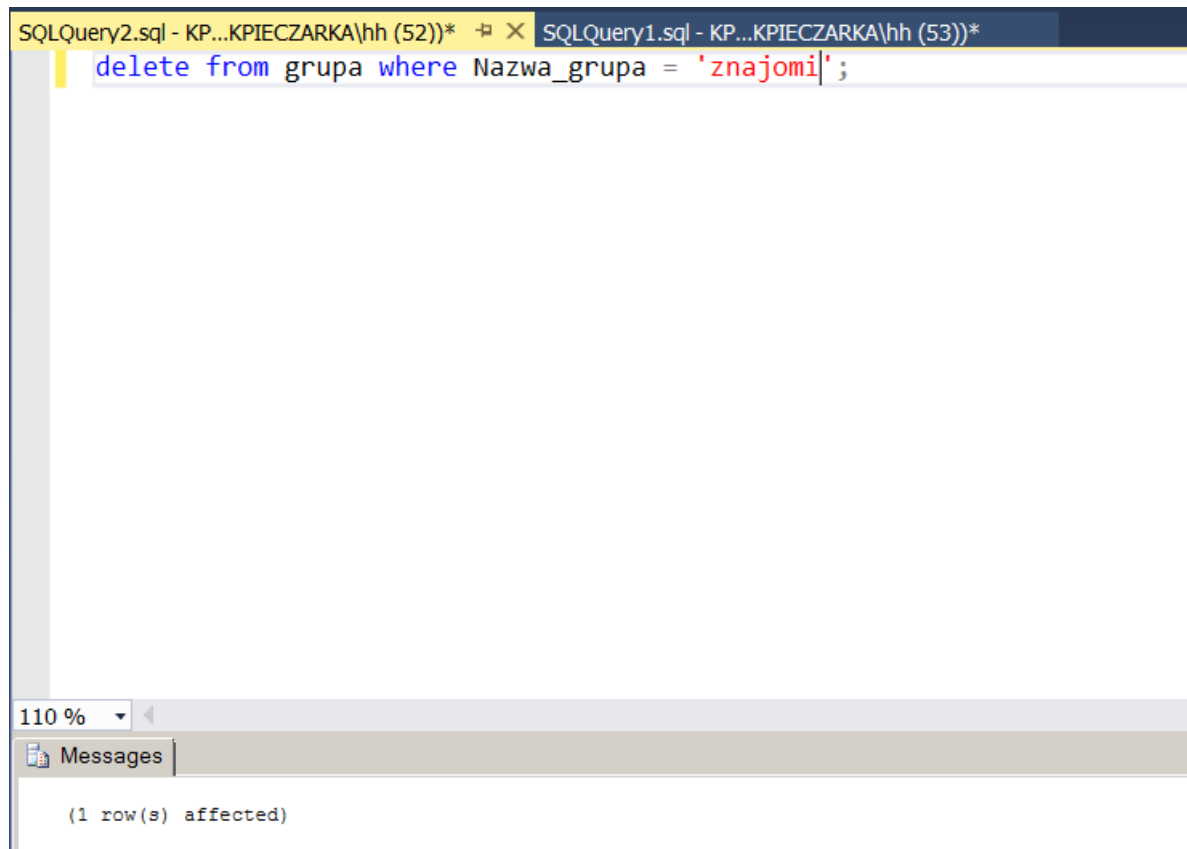
Struktura zapytania **DELETE** jest następująca:

delete from nazwa_tabeli

jednak wykonanie takiego zapytania jak powyżej, spowodowałoby usunięcie wszystkich danych z tabeli, zazwyczaj chcemy usunąć wybrane dane dlatego też wymagane jest dodanie określonego warunku który te dane zdefiniuje

pamiętać należy jednak, że usuwana jest cała krotka (cały wiersz tabeli)

załóżmy, że chcemy usunąć grupę **znajomi**, postać takiego zapytania jest następująca:



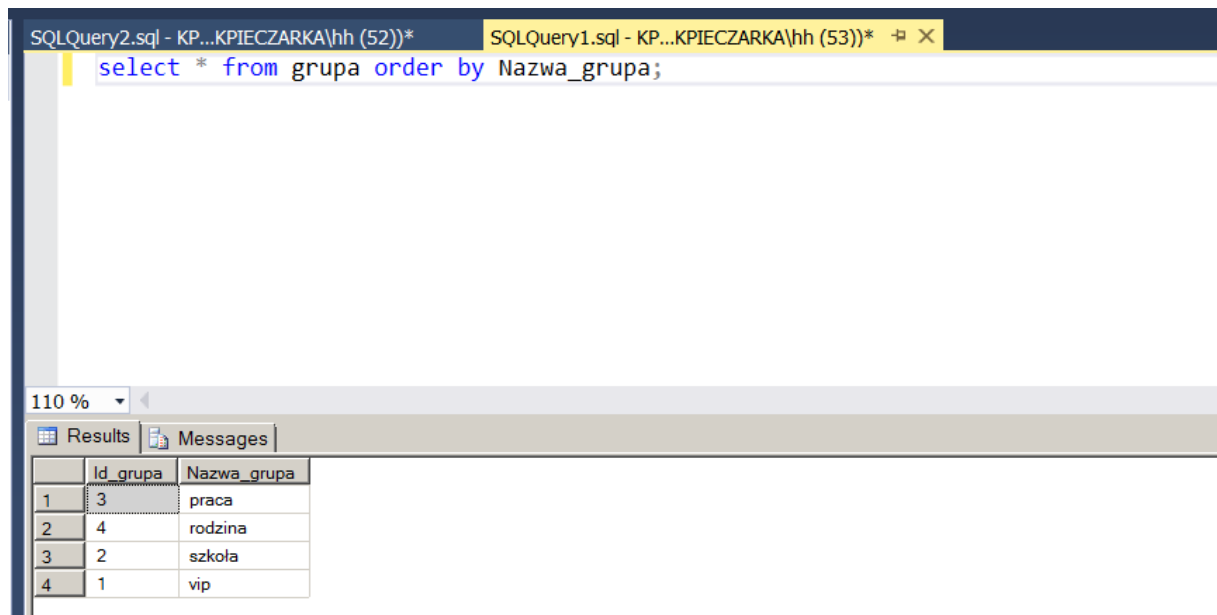
The screenshot shows a SQL query editor window with two tabs: 'SQLQuery2.sql - KP...KPIECZARKA\hh (52)*' and 'SQLQuery1.sql - KP...KPIECZARKA\hh (53)*'. The active tab contains the following SQL statement:

```
delete from grupa where Nazwa_grupa = 'znajomi';
```

Below the editor, there is a 'Messages' pane showing the result of the query execution:

```
(1 row(s) affected)
```


Wówczas



The screenshot shows a SQL query editor with two tabs: "SQLQuery2.sql - KP...KPIECZARKA\hh (52))*" and "SQLQuery1.sql - KP...KPIECZARKA\hh (53))*". The active tab contains the query: `select * from grupa order by Nazwa_grupa;`. Below the query editor, there is a "Results" tab showing a table with 4 rows and 3 columns: "Id_grupa", "Nazwa_grupa", and an unlabeled column. The data is as follows:

	Id_grupa	Nazwa_grupa
1	3	praca
2	4	rodzina
3	2	szkoła
4	1	vip

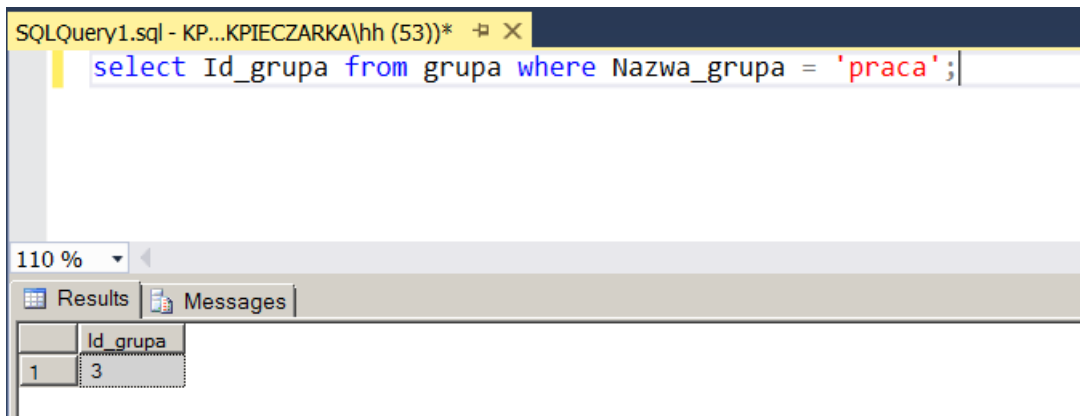
Wprowadzimy teraz dane do tabeli kontakt.

Kolejność argumentów w tabeli tej jest następująca:

Id_kontakt	Id_grupa	Imie	Nazwisko	Adres_1	Adres_2	Adres_3	Telefon	Mobile	Email
------------	----------	------	----------	---------	---------	---------	---------	--------	-------

Klucz **Id_kontakt** będzie wypełniany automatycznie przez DBMS zatem w zapytaniu **INSERT** zostanie pominięty.

Założmy, że chcemy dodać kontakt do grupy **praca**, wówczas dla atrybutu **Id_grupa** w zapytaniu **INSERT** wymagane będzie podanie wartości klucza obcego, który grupie tej odpowiada, zatem najpierw wymagane by było wykonanie zapytania:



```
SQLQuery1.sql - KP...KPIECZARKA\hh (53))* -# X
select Id_grupa from grupa where Nazwa_grupa = 'praca';
```

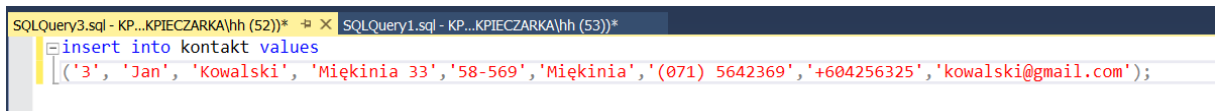
110 %

Results Messages

	Id_grupa
1	3

Otrzymamy wartość tego klucza

Następnie:



```
SQLQuery3.sql - KP...KPIECZARKA\hh (52))* -# X SQLQuery1.sql - KP...KPIECZARKA\hh (53))*
insert into kontakt values
('3', 'Jan', 'Kowalski', 'Miękinia 33', '58-569', 'Miękinia', '(071) 5642369', '+604256325', 'kowalski@gmail.com');
```

Jednak zamiast wykonywać to za pomocą dwóch zapytań można to zrobić w jednym podstawiając zapytanie, które pobiera wartość klucza do zapytania **INSERT**, miało by ono postać:

```
insert into kontakt values
(
(select Id_grupa from grupa where Nazwa_grupa = 'praca'),
'Jan', 'Kowalski', 'Miękinia 33', '58-569', 'Miękinia', '(071)
5642369', '+604256325', 'kowalski@gmail.com'
);
```

Proszę odpowiednio modyfikując to zapytanie dodać kilka przykładowych kontaktów do różnych grup.

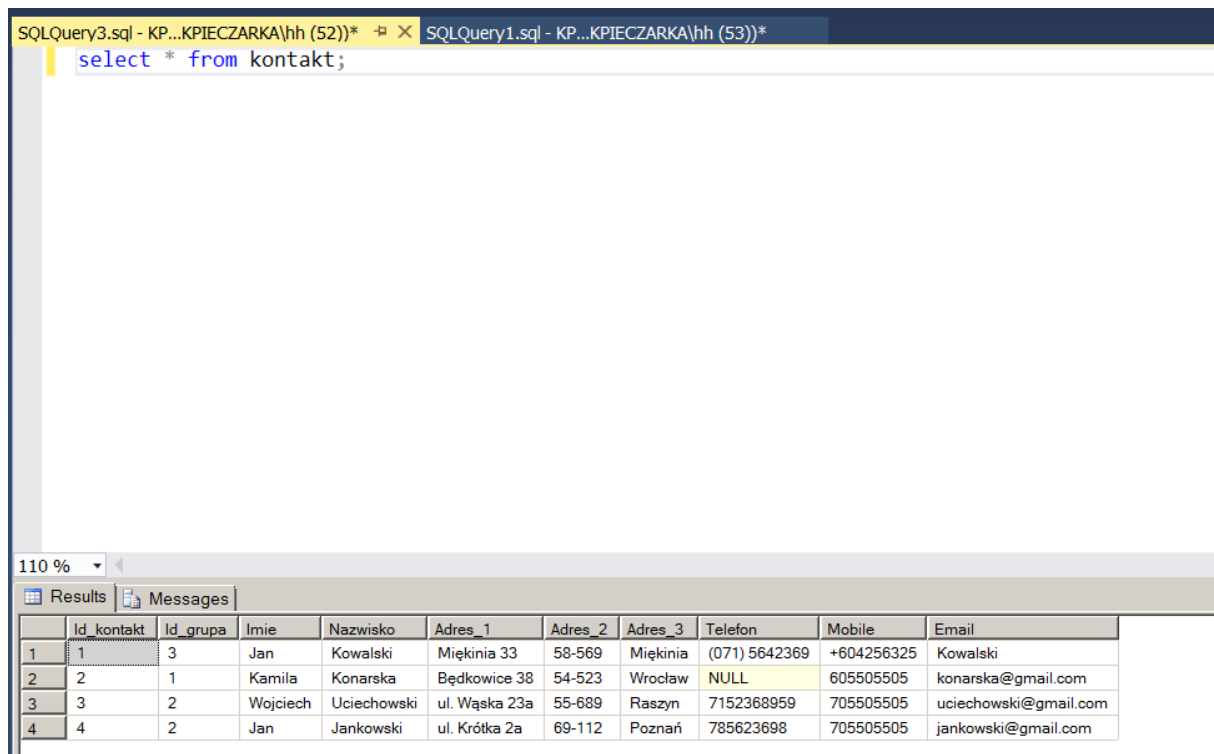
Ponieważ telefon stacjonarny w naszej bazie nie jest wymagany możemy dodać kontakt bez jego podawania.

Wówczas w zapytaniu wymagane jest podanie wartości : NULL

Przykład takiego zapytania:

```
insert into kontakt values ((select id_grupa from grupa where nazwa_grupa = 'VIP'),
'Kamila', 'Konarska', 'Będkowice 38', '54-523', 'Wrocław', NULL, '605505505',
'konarska@gmail.com');
```

Wyświetlmy zatem te dodane kontakty.



The screenshot shows a SQL query execution window with two tabs: 'SQLQuery3.sql - KP...KPIECZARKA\hh (52))*' and 'SQLQuery1.sql - KP...KPIECZARKA\hh (53))*'. The query entered is 'select * from kontakt;'. Below the query editor, the results are displayed in a table with 11 columns: 'Id_kontakt', 'Id_grupa', 'Imie', 'Nazwisko', 'Adres_1', 'Adres_2', 'Adres_3', 'Telefon', 'Mobile', and 'Email'. The table contains 4 rows of data.

	Id_kontakt	Id_grupa	Imie	Nazwisko	Adres_1	Adres_2	Adres_3	Telefon	Mobile	Email
1	1	3	Jan	Kowalski	Miękinia 33	58-569	Miękinia	(071) 5642369	+604256325	Kowalski
2	2	1	Kamila	Konarska	Będkowice 38	54-523	Wrocław	NULL	605505505	konarska@gmail.com
3	3	2	Wojciech	Uciechowski	ul. Wąska 23a	55-689	Raszyn	7152368959	705505505	uciechowski@gmail.com
4	4	2	Jan	Jankowski	ul. Krótka 2a	69-112	Poznań	785623698	705505505	jankowski@gmail.com

Jednak w tabeli **kontakt** nie są przechowywane **nazwy grup** a jedynie wartości klucza obcego, które odpowiadają konkretnym nazwą grup w tabeli grupa.

By zatem wyświetlić nazwę grupy i pełne dane określonego kontaktu wymagane jest pobranie danych z dwóch tabel.

Postać takiego zapytania byłaby następująca:

select

tabela_1.atrybut_1, tabela_1.atrybut_2, tabela_2.atrybut_1, tabela_2.atrybut_2

from

tabela_1, tabela_2;

Zatem w naszym przypadku:

```
SQLQuery3.sql - KP...KPIECZARKA\hh (52))* SQLQuery1.sql - KP...KPIECZARKA\hh (53))*
select
  grupa.Nazwa_grupa, kontakt.Imie, kontakt.Nazwisko, kontakt.Adres_1, kontakt.Adres_2, kontakt.Adres_3,
  kontakt.Telefon, kontakt.Mobile, kontakt.Email
from
  grupa, kontakt
```

110 %

Results Messages

	Nazwa_grupa	Imie	Nazwisko	Adres_1	Adres_2	Adres_3	Telefon	Mobile	Email
1	vip	Jan	Kowalski	Miękinia 33	58-569	Miękinia	(071) 5642369	+604256325	Kowalski
2	szkoła	Jan	Kowalski	Miękinia 33	58-569	Miękinia	(071) 5642369	+604256325	Kowalski
3	praca	Jan	Kowalski	Miękinia 33	58-569	Miękinia	(071) 5642369	+604256325	Kowalski
4	rodzina	Jan	Kowalski	Miękinia 33	58-569	Miękinia	(071) 5642369	+604256325	Kowalski
5	vip	Kamila	Konarska	Będkowice 38	54-523	Wrocław	NULL	605505505	konarska@gmail.com
6	szkoła	Kamila	Konarska	Będkowice 38	54-523	Wrocław	NULL	605505505	konarska@gmail.com
7	praca	Kamila	Konarska	Będkowice 38	54-523	Wrocław	NULL	605505505	konarska@gmail.com
8	rodzina	Kamila	Konarska	Będkowice 38	54-523	Wrocław	NULL	605505505	konarska@gmail.com
9	vip	Wojciech	Uciechowski	ul. Wąska 23a	55-689	Raszyn	7152368959	705505505	uciechowski@gmail.com
10	szkoła	Wojciech	Uciechowski	ul. Wąska 23a	55-689	Raszyn	7152368959	705505505	uciechowski@gmail.com
11	praca	Wojciech	Uciechowski	ul. Wąska 23a	55-689	Raszyn	7152368959	705505505	uciechowski@gmail.com
12	rodzina	Wojciech	Uciechowski	ul. Wąska 23a	55-689	Raszyn	7152368959	705505505	uciechowski@gmail.com
13	vip	Jan	Jankowski	ul. Krótka 2a	69-112	Poznań	785623698	705505505	jankowski@gmail.com
14	szkoła	Jan	Jankowski	ul. Krótka 2a	69-112	Poznań	785623698	705505505	jankowski@gmail.com
15	praca	Jan	Jankowski	ul. Krótka 2a	69-112	Poznań	785623698	705505505	jankowski@gmail.com
16	rodzina	Jan	Jankowski	ul. Krótka 2a	69-112	Poznań	785623698	705505505	jankowski@gmail.com

Jednak wyświetlone dane nie są prawidłowe, został wykonany iloczyn na zbiorach dwóch tabel.

Ponieważ tabele połączone są relacją wymagane jest zdefiniowanie warunku złączenia można to zrobić w klauzuli **WHERE**, a warunek ten jest następujący:

Wartość klucza głównego **Id_grupa** w tabeli **grupa** ma być równe wartości klucza obcego **Id_grupa** w tabeli **kontakt**, zatem:

```
SQLQuery3.sql - KP...KPIECZARKA\hh (52)*  SQLQuery1.sql - KP...KPIECZARKA\hh (53)*
select
  grupa.Nazwa_grupa, kontakt.Imie, kontakt.Nazwisko, kontakt.Adres_1, kontakt.Adres_2, kontakt.Adres_3,
  kontakt.Telefon, kontakt.Mobile, kontakt.Email
from
  grupa, kontakt
where
  grupa.Id_grupa = kontakt.Id_grupa;
```

110 %

Results Messages

	Nazwa_grupa	Imie	Nazwisko	Adres_1	Adres_2	Adres_3	Telefon	Mobile	Email
1	vip	Kamila	Konarska	Będkowice 38	54-523	Wrocław	NULL	605505505	konarska@gmail.com
2	szkola	Wojciech	Uciechowski	ul. Wąska 23a	55-689	Raszyn	7152368959	705505505	uciechowski@gmail.com
3	szkola	Jan	Jankowski	ul. Krótka 2a	69-112	Poznań	785623698	705505505	jankowski@gmail.com
4	praca	Jan	Kowalski	Miękinia 33	58-569	Miękinia	(071) 5642369	+604256325	Kowalski

W programie PowerDesigner zaprojektuj diagram konceptualny bazy dla danych jak poniżej.

W przypadku atrybutów (kolumn) w których dane powtarzają się wymagane będzie zaprojektowanie osobnych encji (tabel). Zaprojektowane tabele połącz odpowiednimi relacjami.

	Nazwa_marka	Nazwa_model	rok_prod	Nazwa_kolor	nr_rej	przebieg	cena
1	Alfa Romeo	145	1986	niebieski	SK 15864	243000	12990
2	Fiat	Panda	2010	biały	DW 52887	112000	14000
3	Fiat	Panda	2010	biały	ZK 99775	274000	7900
4	Ford	Mustang	2019	czerwony	NULL	3	258810
5	BMW	M6	2019	niebieski	NULL	5	973000
6	Audi	Q5	2014	granat	DW 1254H	86000	119000

Wygeneruj diagram fizyczny bazy danych.

Wygeneruj skrypt CREATE bazy danych.

Na serwerze MSSQL utwórz nową bazę danych.

Przy pomocy skryptu CREATE utwórz strukturę bazy danych.

Dla każdej z tabel ustaw opcję auto-upełniania dla klucza głównego.

Przy pomocy odpowiednich zapytań INSERT wprowadź dane do poszczególnych tabel.

Utwórz zapytanie SELECT, które zwróci dane jak powyżej.

Utwórz zapytanie SELECT w wyniku, którego dane będą posortowane malejąco wg. ceny.

	Nazwa_marka	Nazwa_model	rok_prod	Nazwa_kolor	nr_rej	przebieg	cena
1	BMW	M6	2019	niebieski	NULL	5	973000
2	Ford	Mustang	2019	czerwony	NULL	3	258810
3	Audi	Q5	2014	granat	DW 1254H	86000	119000
4	Fiat	Panda	2010	biały	DW 52887	112000	14000
5	Alfa Romeo	145	1986	niebieski	SK 15864	243000	12990
6	Fiat	Panda	2010	biały	ZK 99775	274000	7900

Utwórz zapytanie SELECT w wyniku, którego dane będą posortowane rosnąco wg. marki.

	Nazwa_marka	Nazwa_model	rok_prod	Nazwa_kolor	nr_rej	przebieg	cena
1	Alfa Romeo	145	1986	niebieski	SK 15864	243000	12990
2	Audi	Q5	2014	granat	DW 1254H	86000	119000
3	BMW	M6	2019	niebieski	NULL	5	973000
4	Fiat	Panda	2010	biały	DW 52887	112000	14000
5	Fiat	Panda	2010	biały	ZK 99775	274000	7900
6	Ford	Mustang	2019	czerwony	NULL	3	258810