

<b>Projekt Technologie obiektowe</b> <b>Wydział Elektrotechniki Automatyki i Informatyki</b> <b>Politechnika Świętokrzyska</b>	
<b>Studia: Stacjonarne II stopnia</b>	<b>Kierunek: Informatyka</b>
<b>Grupa: 1ID22A</b>	<b>Autor:</b> <b>Mateusz Skrok</b>
<b>Generator kodu SQL dla utworzonego schematu bazy danych</b>	

## 1. Cel projektu

Celem projektu było stworzenie aplikacji umożliwiającej na podstawie utworzonego schematu bazy danych wygenerowanie kodu SQL. Schemat ten powinien obsługiwać wszystkie trzy rodzaje relacji czyli: jeden do jednego, jeden do wielu oraz wielu do wielu, jak i również dziedziczenie.

## 2. Wstęp teoretyczny

- Java jest obiektowy językiem programistycznym stworzony z myślą o ogólnej dostępności, jak i również prostocie w użyciu. Została przedstawiona publicznie pod koniec 1995 roku przez firmę Sun Microsystems jako Java 1.0. Składnia Javy opiera się na stylu pisania oprogramowania, który został przedstawiony w języku programowania C++, zabieg ten służył do tego, aby programiści zaznajomieni z tym językiem programowania mogli łatwiej korzystać z funkcjonalności, jakie oferuje Java, dodatkowo Java uznają się za mniej skomplikowany język programistyczny od C++, poprzez zastąpienie lub usunięcie niewygodnych operacji takich jak wskaźniki, struktury, przeciążenia operatorów oraz kilka innych operacji, które są ważnymi elementami programowania w C++, który również umożliwia programowanie obiektowe. W przypadku procesu kompilacji programów w Javie został, rozpoczyna on działania od przekazania kodu źródłowego aplikacji do programu `javac`, proces ten służy do utworzenia plików klasy zawierających kod bajtowy, jest to konieczne, ponieważ plik klasy jest najmniejszą jednostką funkcjonalności, która jest wykorzystywana przez platformę, oraz jedyną możliwością przesłania kodu do działającego programu. Programowanie obiektowe jest rodzajem oprogramowania działającego na podstawie obiektów, które przedstawiają dane nazywane inaczej polami oraz działania, jakie można z tymi danymi podjąć, tak zwane metody. Ważnym elementem programowania obiektowego są klasy, które służą do tworzenia obiektów, według ustalonego schematu danej klasy, oznacza to, że każdy obiekt stworzonej klasy posiada własną wersję pól oraz metod przedstawionych w danej klasie. Program utworzone przy użyciu programowania obiektowego wykonuje operacje, korzystając ze zbioru obiektów, które do wykonania określonego zadania, komunikują się ze sobą. Sposób programowania obiektowego

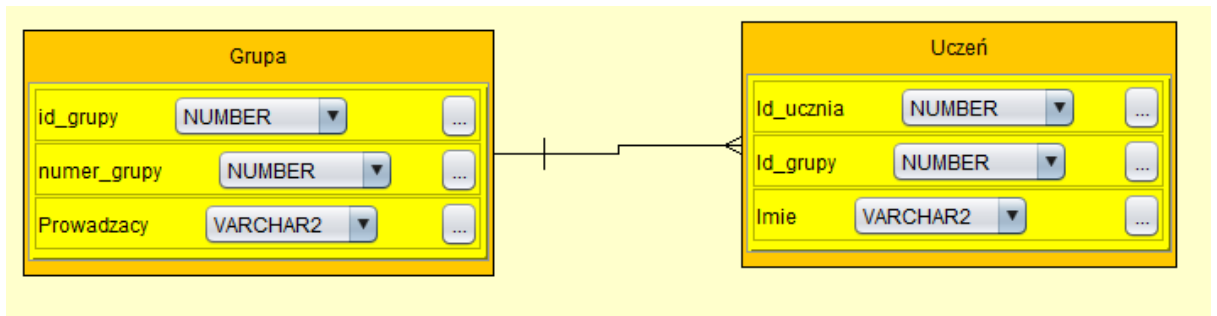
został spopularyzowany przez wprowadzenie C++ który był znaczącym rozszerzeniem do języka programowania C. Do najważniejszych atrybutów programowania obiektowego zalicza się: dziedziczenie, hermetyzacja oraz polimorfizm. Dziedziczenie jest jedną z ważniejszych cech programowania obiektowego, ponieważ poprzez dziedziczenie programista ma możliwość stworzenia klasy, która może odwołać się do nie prywatnych pól i metod klas bazowych, co pozwala na tworzenie różnych klas posiadających wspólne cechy, realizowane jest to poprzez stworzenie klasy zawierającej pola oraz metody wspólne dla kilku klas, a następnie utworzenie klas, które dziedziczą po tej klasie.

- Java Swing jest to interfejs programistyczny umożliwiający tworzenie graficznego interfejsu użytkownika, aplikacje utworzone w ten sposób nazywane są okienkowymi lub desktopowymi. Wykorzystywane są do graficznego przedstawienia elementów, które ułatwiają komunikację użytkownika z komputerem, poprzez różnego rodzaju tablice, przyciski czy pola do wprowadzania tekstu. Swing został wypuszczony w 1997 roku, jako ulepszona wersja AWT, przy współpracy dwóch firm Sun Microsystems i Netscape Communications Corporation. Początkowo był on dostępny jako oddzielna biblioteka do pobrania i zaimplementowania przez użytkownika, został później zaimplementowany do Javy jak część Java Standard Edition wersja 1.2 i jest dostępny dla każdego użytkownika korzystającego z Javy pod pakietem javax.swing. Głównymi zaletami pakietu Swing jest duża ilość elementów interfejsu graficznego, jak i również wygoda ich użytkowania, jest on mniej podatny na problemy związane z różnymi systemami dodatkowo każdy element oraz sposób jego działania jest taki sam na każdej platformie. Dodatkową zaletą Java Swing są elementy graficzne, które wyświetlana są poprzez rysowanie się na ekranie, z tego powodu większość komponentów pakietu Swing jest mniej obciążającą dla systemu. Mniej ważną cechą, ale przydatną w tworzeniu aplikacji okienkowych jest również możliwość przypisywania poszczególnych klawiszy klawiatury do odpowiednich komponentów.
- SQL został zaprojektowany głównie do pracy z relacyjnymi bazami danych, opiera się on na prostych zdaniach posiadających instrukcję kluczową oraz informację zależną od stworzonej bazy danych. Najczęściej używaną instrukcją SQL jest SELECT, który służy do zwrócenia wierszy z określonych tabel zakładając przy tym określone przez użytkownika warunki. Do modyfikacji informacji w tabelach

wykorzystywane są instrukcje INSERT, UPDATE oraz DELETE, natomiast do tworzenia struktury bazy danych, jak i modyfikacji jej tabel należy skorzystać z instrukcji CREATE, ALTER, DROP oraz RENAME.

- Relacyjne bazy danych służą do przechowywania powiązanych ze sobą informacji w tabelach, każda informacja jest przechowywana w oddzielnym wierszu, gdzie kolumny określają przechowywaną informację. Tabele w tym typie bazy danych są powiązane ze sobą relacjami, wyróżniamy trzy rodzaje powiązań relacji: jeden do jeden, jeden do wielu oraz wiele do wielu, do określenia tych powiązań wykorzystuje się klucze główne oraz obce. Powiązania te ułatwiają użytkownikowi określenie, w jaki sposób informacje są przechowywane w bazie danych oraz zabezpieczają tą bazę przed niepoprawnym wprowadzaniem danych.
- Relacja jeden-do-wielu jest najczęstszym rodzajem relacji. W tego rodzaju relacji wiersz w tabeli A może mieć wiele pasujących wierszy w tabeli B. Jednak wiersz w tabeli B może mieć tylko jeden pasujący wiersz w tabeli A. Na przykład tabele „Wydawcy” i „Tytuły” mają relację jeden do wielu. Oznacza to, że każdy wydawca produkuje wiele tytułów. Ale każdy tytuł pochodzi tylko od jednego wydawcy. Relacja jeden-do-wielu jest tworzona, jeśli tylko jedna z powiązanych kolumn jest kluczem podstawowym lub ma unikatowe ograniczenie.

```
CREATE TABLE Uczeń (  
  Id_ucznia NUMBER ,  
  Id_grupy NUMBER ,  
  Imie VARCHAR2  
);  
  
CREATE TABLE Grupa (  
  id_grupy NUMBER PRIMARY KEY,  
  numer_grupy NUMBER ,  
  Prowadzacy VARCHAR2  
);  
  
ALTER TABLE Uczeń ADD CONSTRAINT K22FgIMRoI FOREIGN KEY (Id_grupy)  
REFERENCES Grupa(id_grupy);
```



- W relacji wiele-do-wielu wiersz w tabeli A może mieć wiele pasujących wierszy w tabeli B i odwrotnie. Taką relację można utworzyć, definiując trzecią tabelę, która jest nazywana tabelą skrzyżowań. Klucz podstawowy tabeli skrzyżowań składa się z kluczy obcych z tabeli A i tabeli B. Na przykład tabela „Autorzy” i tabela „Tytuły” są w relacji wiele-do-wielu, która jest definiowana przez relację jeden-do-wielu z każdej z tych tabel do tabeli „TytułAutorzy”.

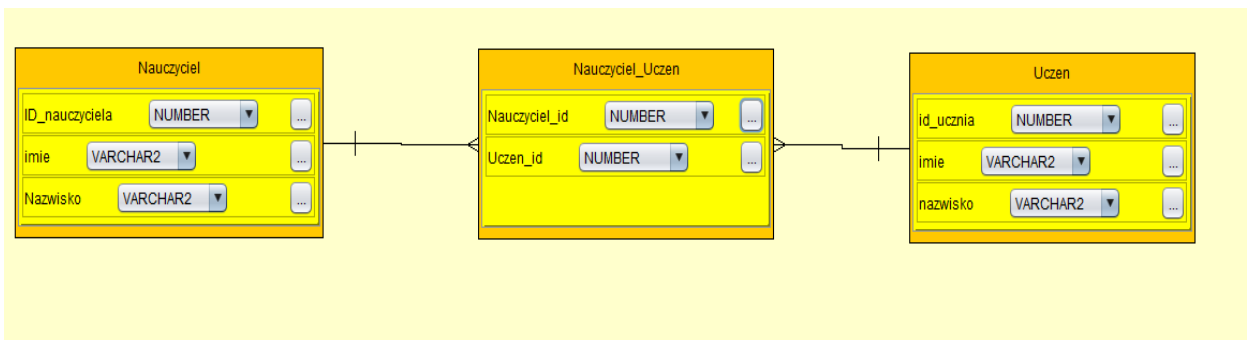
```

CREATE TABLE Nauczyciel (
ID_nauczyciela NUMBER PRIMARY KEY,
imie VARCHAR2 ,
Nazwisko VARCHAR2
);

CREATE TABLE Uczen (
id_ucznia NUMBER PRIMARY KEY,
imie VARCHAR2 ,
nazwisko VARCHAR2
);

CREATE TABLE Nauczyciel_Uczen (
Nauczyciel_id NUMBER ,
Uczen_id NUMBER
);

ALTER TABLE Nauczyciel_Uczen ADD CONSTRAINT 4CvIr3GE33 FOREIGN
KEY (Nauczyciel_id) REFERENCES Nauczyciel(ID_nauczyciela);
ALTER TABLE Nauczyciel_Uczen ADD CONSTRAINT ByDQ4O4sJy FOREIGN
KEY (Uczen_id) REFERENCES Uczen(id_ucznia);
  
```

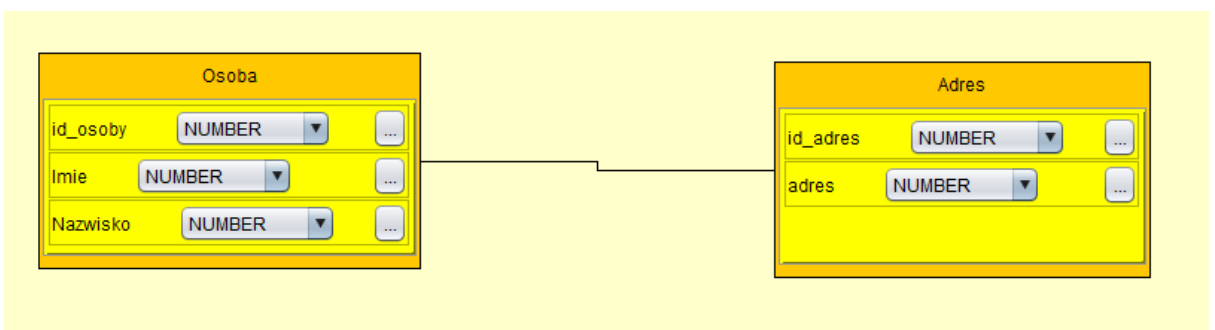


- W relacji jeden-do-jednego wiersz w tabeli A może mieć nie więcej niż jeden pasujący wiersz w tabeli B i odwrotnie. Relacja jeden-do-jednego jest tworzona, jeśli obie kolumny powiązane są kluczami podstawowymi lub mają wyjątkowe ograniczenia. Ten rodzaj relacji nie jest powszechny, ponieważ większość informacji, które są związane w ten sposób znajdzie się w jednej tabeli.

```
CREATE TABLE Adres (
id_adres NUMBER PRIMARY KEY,
adres NUMBER
);

CREATE TABLE Osoba (
id_osoby NUMBER PRIMARY KEY,
Imie NUMBER ,
Nazwisko NUMBER
);

ALTER TABLE Osoba ADD CONSTRAINT Y1RR3ezshU FOREIGN KEY
(id_osoby) REFERENCES Adres(id_adres);
```



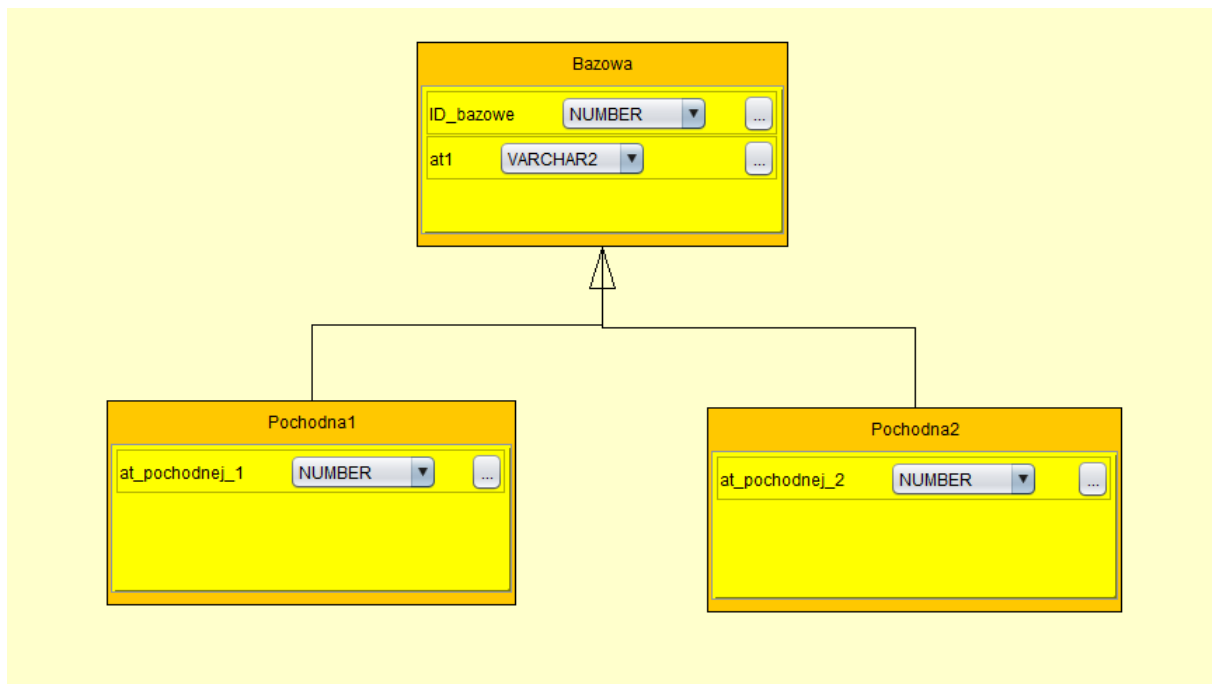
- Techniki dziedziczenia przedstawione w SQL:

**'Table per concrete class'** czyli każdej nie abstrakcyjnej klasie odpowiada tabela w bazie danych. Jest to podejście najprostsze, ale jednocześnie najmniej eleganckie. Na poziomie bazodanowym nie występują jakiegokolwiek związku między klasami dziedziczącymi.

**'Table per class'** czyli utworzenie jednej tabeli dla wszystkich podklas. Taka tabela zawiera kolumny odpowiadające wszystkim trwałym atrybutom wszystkich podklas. Dodatkowo, potrzebna jest jedna kolumna nazywana *discriminator*, która określa, do jakiej podklasy należy obiekt odpowiadający danemu rekordowi w tabeli.

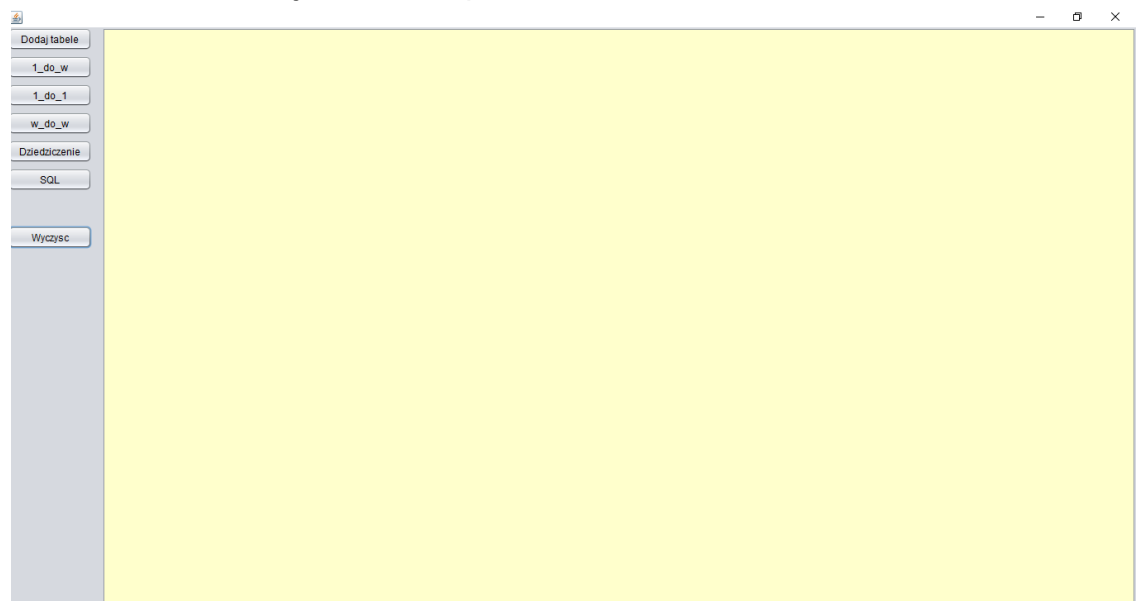
**'Table per subclass'** wierne odwzorowanie modelu dziedziczenia do postaci tabelowej. Tworzymy tabelę dla nadklasy, oraz po jednej tabeli dla każdej podklasy, przy czym tabela reprezentująca podklasę nie powiela żadnego pola, które wystąpiło w nadklasie. Do rozróżnienia rekordów używane są klucze główne tabel podklasy jako klucze obcy do tabeli nadklasy.

```
CREATE TABLE Bazowa (  
ID_bazowe NUMBER PRIMARY KEY,  
at1 VARCHAR2  
);  
  
CREATE TABLE Pochodna1 (  
ID_bazowe NUMBER PRIMARY KEY,  
at_pochodnej_1 NUMBER  
);  
  
CREATE TABLE Pochodna2 (  
ID_bazowe NUMBER PRIMARY KEY,  
at_pochodnej_2 NUMBER  
);  
  
ALTER TABLE Pochodna1 ADD CONSTRAINT cmTnSBBgtc FOREIGN KEY  
(ID_bazowe) REFERENCES Bazowa(ID_bazowe);  
ALTER TABLE Pochodna2 ADD CONSTRAINT VozdgOfYnR FOREIGN KEY  
(ID_bazowe) REFERENCES Bazowa(ID_bazowe);
```



### 3. Instrukcja obsługi

Po uruchomieniu aplikacji zostanie wyświetlone okno z rysunku 3.1 na którym użytkownik może dodawać nowe tabele wraz z ich atrybutami oraz łączyć utworzone tabele relacją jak i również poprzez dziedziczenie które jest odzwierciedlone w kodzie SQL jako “table per subclass”.



Rys.3.1 Główne okno aplikacji.



Aby utworzyć tabelę użytkownik musi kliknąć w przycisk “Dodaj tabelę” oraz wprowadzić jej nazwę w oknie kontekstowym które zostanie wyświetlone, po poprawnym wprowadzeniu nazwy tabela o podanej nazwie zostanie wyświetlona w oknie aplikacji. Każdą z tak utworzonych tabel można edytować poprzez naciśnięcie PPM(prawego przycisku myszy), po naciśnięciu PPM wyświetlone zostanie menu w którym użytkownik może edytować nazwę tabeli, usunąć ją z schematu, dodać atrybut jak i również usunąć relacje jakie są z nią powiązane. Ważnym elementem w tym menu jest możliwość dodania atrybutu, dodaje się go najpierw podając jego nazwę w oknie które się wyświetli, po poprawnym wpisaniu nazwy atrybut zostanie przypisany do tabeli. Atrybuty tak samo jak tabelę można edytować poprzez użycie PPM, ale również można określić ich typ poprzez wybranie go z rozwijalnej list oraz klikając przycisk “...” można określić “Constraints” czyli specjalne zasady które muszą być przestrzegane wobec tego atrybutu, należą do nich np: Primary Key, Unique lub Not Null. Przykładowa tabela wraz z atrybutami jest przedstawiona na rysunek 3.2

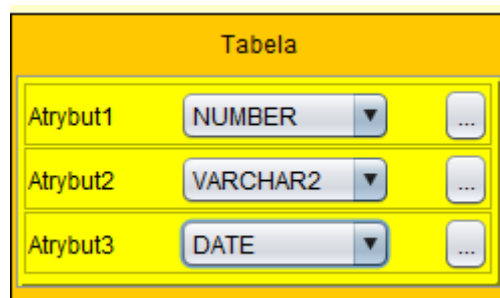
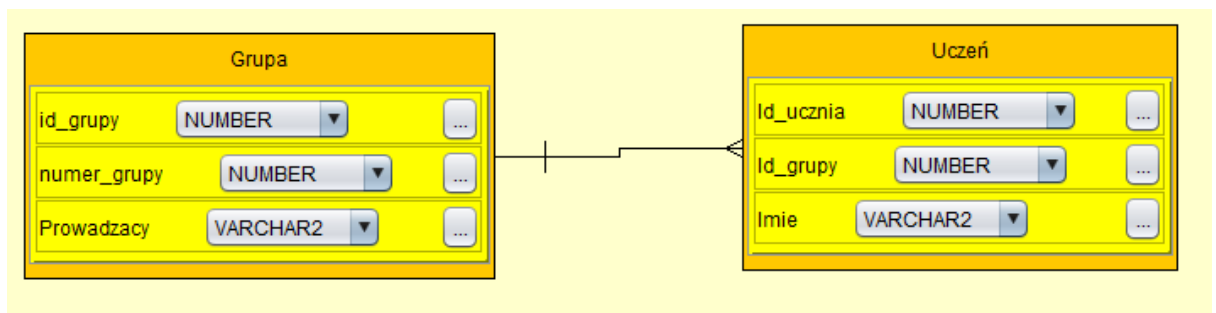


Tabela		
Atrybut1	NUMBER	...
Atrybut2	VARCHAR2	...
Atrybut3	DATE	...

Rys.3.2. Tabela

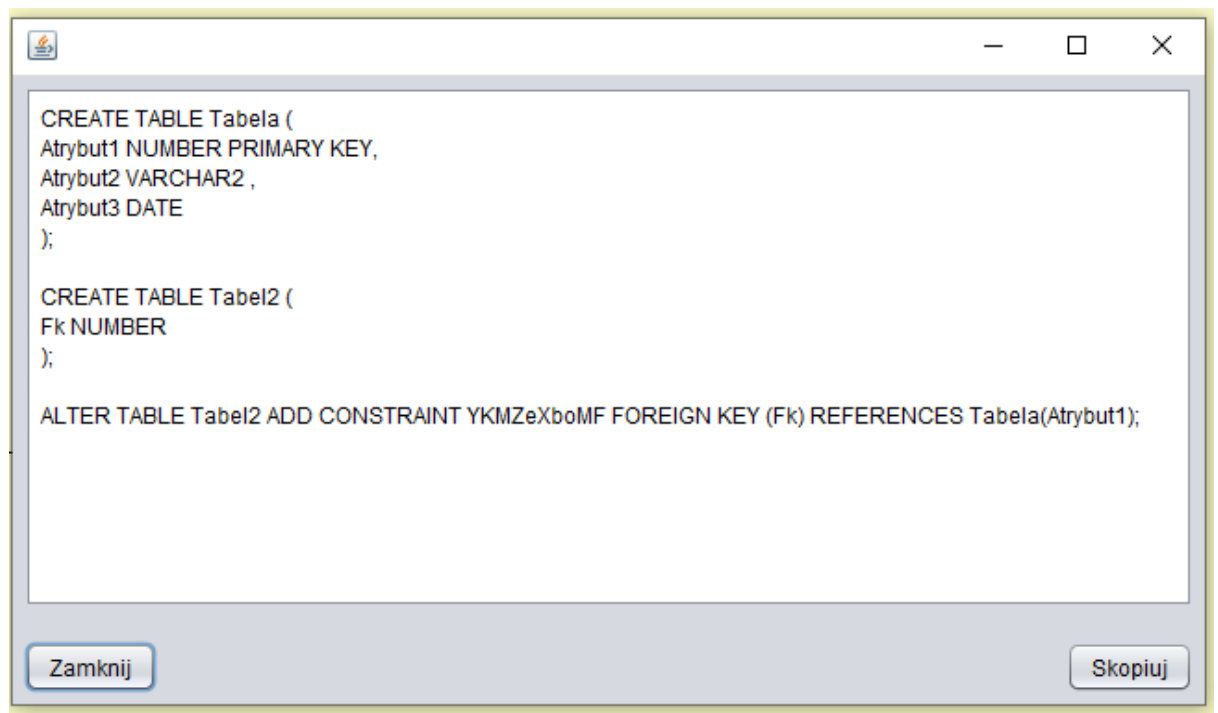
Kolejnym ważnym elementem w projektowaniu relacyjnej bazy danych jest możliwość połączenia tabel relacjami. W przypadku tej aplikacji należy nacisnąć jeden z przycisków “1\_do\_1”, “1\_do\_w” oraz “w\_do\_w”, jak i również istnieje możliwość utworzenia połączenia poprzez dziedziczenie w tym przypadku należy użyć przycisku “Dziedziczenie”. Relacje mogą być tworzone pomiędzy dwoma tabelami oraz każda z nich musi mieć przynajmniej jeden atrybut, jak i również w przypadku relacji jeden do wielu oraz jeden do jednego druga tabela musi mieć atrybut z kluczem głównym, natomiast w wielu do wielu obie tabele muszą posiadać klucze główne ponieważ pomiędzy nimi tworzona jest pośrednia tabela która przechowuje atrybuty z kluczami głównymi do tych tabel. Dodawanie relacji odbywa się poprzez wybranie w

oknie tabel które chcemy ze sobą połączyć relacją oraz atrybutów które mają zawierać klucz obcy oraz główny. Natomiast w przypadku dziedziczenia, aplikacja ta działa w oparciu o “table per subclass”, czyli w każdej tabeli pochodnej tworzony jest dodatkowy klucz główny, który odwołuje się do klucza głównego z tabeli bazowej. Z tego powodu w przypadku tworzenie dziedziczenia pomiędzy tabelami, aplikacji nie umożliwi utworzenia połączenia, gdy tabela bazowa nie posiada klucza głównego. Dodając dziedziczenie wybieramy klasę bazą oraz pochodną w oknie które pojawi się po naciśnięciu przycisku “Dziedziczenie”. Przykładem relacji jeden do wielu jest rysunek 3.3



Rys.3.3 Relacja

Aby wygenerować kod SQL, po zaprojektowanie schematu bazy danych należy skorzystać z przycisku “SQL” które wyświetli okno przedstawione na rysunku 3.4. Z tego okna istnieje możliwość skopiowanie kodu i poprzez uruchomieni go dla bazy danych oracle zostanie wygenerowana taka sam baza danych jak na zaprojektowanym schemacie. Również w przypadku zmian w schemacie po ponownym skorzystaniu z przycisku SQL zostaną wprowadzone zmiany w kodzie.



Rys.3.4 Kod SQL