

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

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

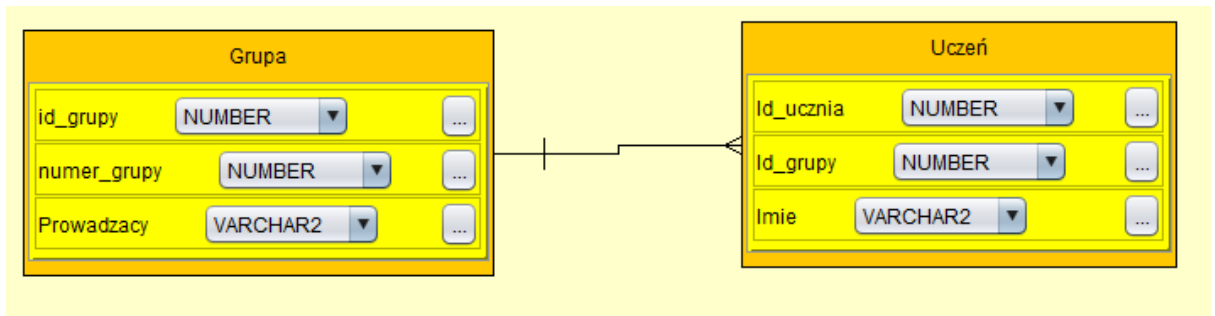
został spopularyzowany przez wprowadzenie C++ który był znaczącym rozszerzeniem do języka programowania C. Do najważniejszych atrybutów programowania obiektowego zalicza się: dziedziczenie, hermetyzacja oraz polimorfizm. Dziedziczenie jest jedną z ważniejszych cech programowania obiektowego, ponieważ 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);
```



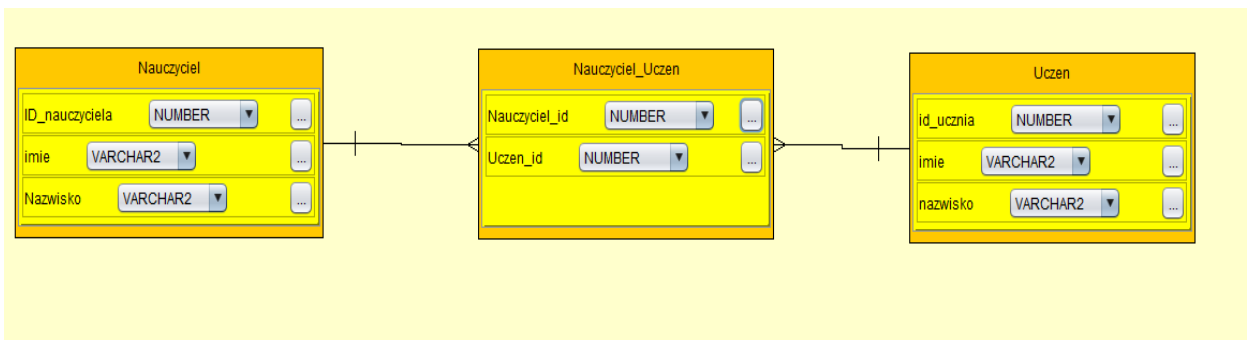
- W relacji wiele-do-wielu wiersz w tabeli A może mieć wiele pasujących wierszy w tabeli B i odwrotnie. Taką relację można utworzyć, definiując trzecią tabelę, która jest nazywana tabelą skrzyżowań. Klucz podstawowy tabeli skrzyżowań składa się z kluczy obcych z tabeli A i tabeli B. Na przykład tabela „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 Ucen (
id_ucznia NUMBER PRIMARY KEY,
imie VARCHAR2 ,
nazwisko VARCHAR2
);

CREATE TABLE Nauczyciel_Ucen (
Nauczyciel_id NUMBER ,
Ucen_id NUMBER
);

ALTER TABLE Nauczyciel_Ucen ADD CONSTRAINT 4CvIr3GE33 FOREIGN
KEY (Nauczyciel_id) REFERENCES Nauczyciel(ID_nauczyciela);
ALTER TABLE Nauczyciel_Ucen ADD CONSTRAINT ByDQ4O4sJy FOREIGN
KEY (Ucen_id) REFERENCES Ucen(id_ucznia);
```

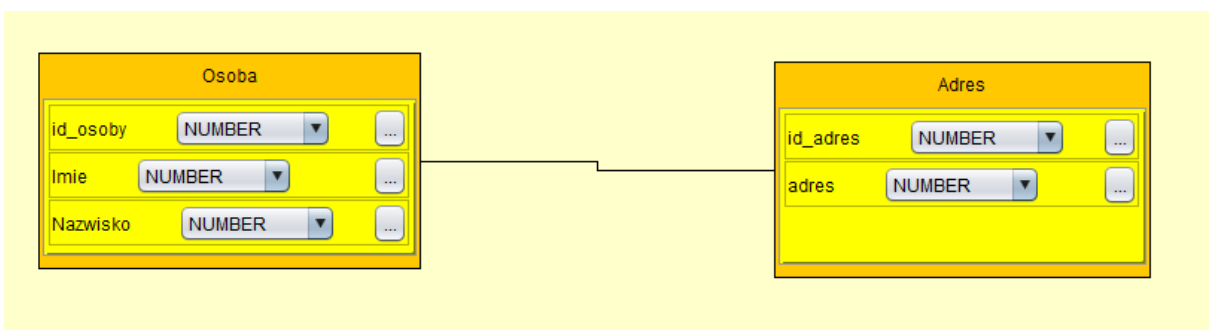


- W relacji jeden-do-jednego wiersz w tabeli A może mieć nie więcej niż jeden pasujący wiersz w tabeli B i odwrotnie. Relacja jeden-do-jednego jest tworzona, jeśli obie kolumny powiązane są kluczami podstawowymi lub mają wyjątkowe ograniczenia. Ten rodzaj relacji nie jest powszechny, ponieważ większość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



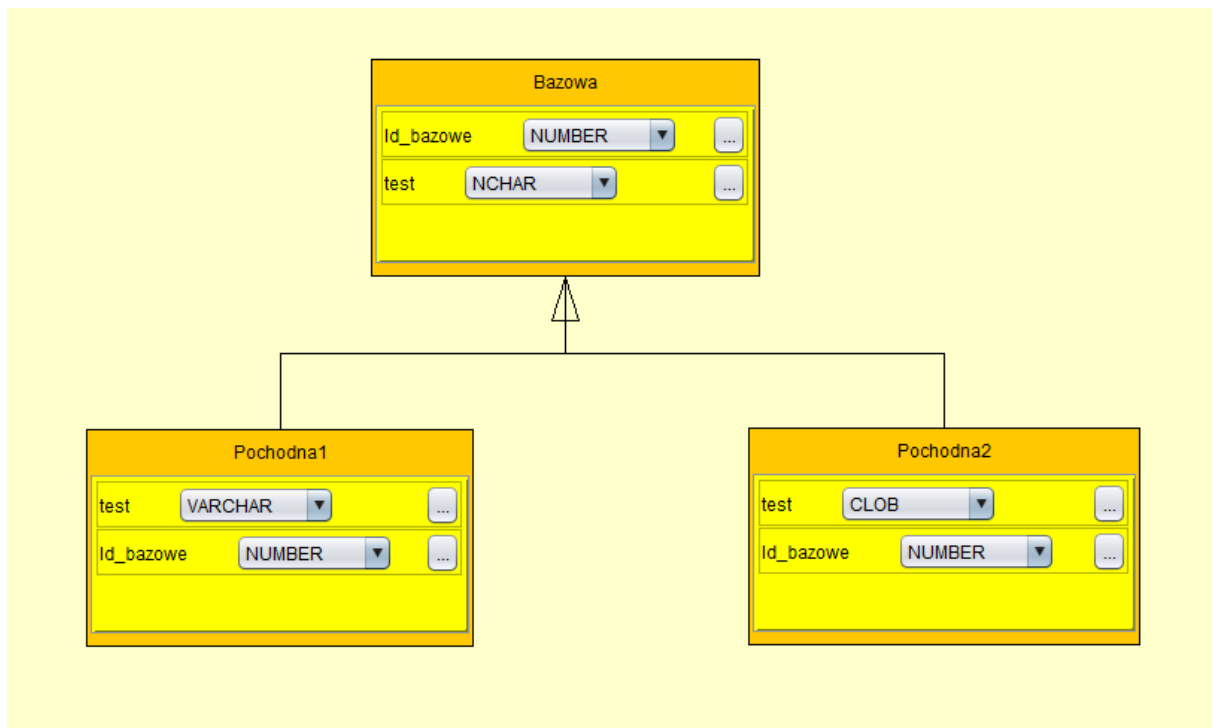
- Techniki dziedziczenia przedstawione w SQL:

'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.

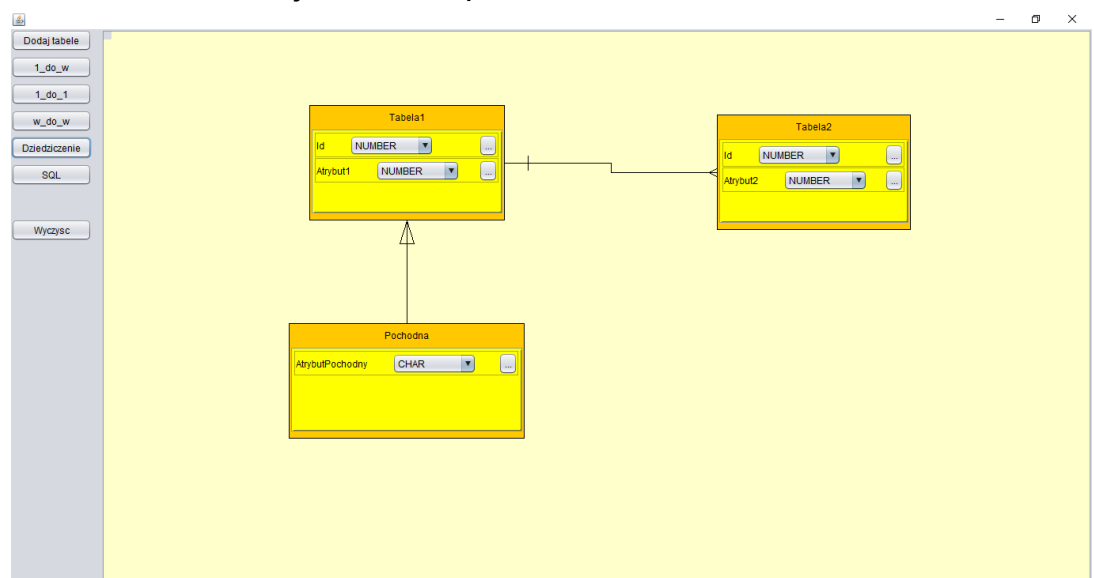
'Table per subclass' wierne odwzorowanie modelu dziedziczenia do postaci tabelowej. Tworzymy tabelę dla nadklasy, oraz po jednej tabeli dla każdej podklasy, przy czym tabela reprezentująca podklasę nie powiela żadnego pola, które wystąpiło w nadklasie. Do rozróżnienia rekordów używane są klucze główne 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 relacja jaki i również poprzez dziedziczenie które jest odzwierciedlone w kodzie SQL jako “table per subclass”.



Rys.3.1 Główne okno aplikacji.

Aby utworzyć tabele użytkownik musi kliknąć w przycisk “Dodaj tabele” 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 tabele 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

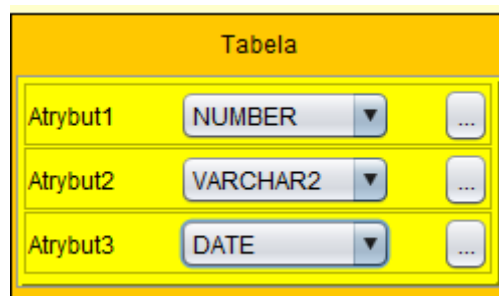
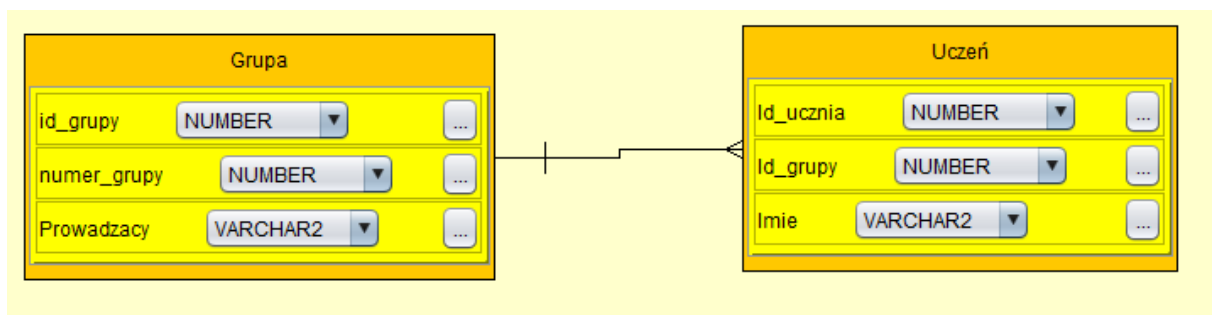


Tabela		
Atrybut1	NUMBER	...
Atrybut2	VARCHAR2	...
Atrybut3	DATE	...

Rys.3.2. Tabela

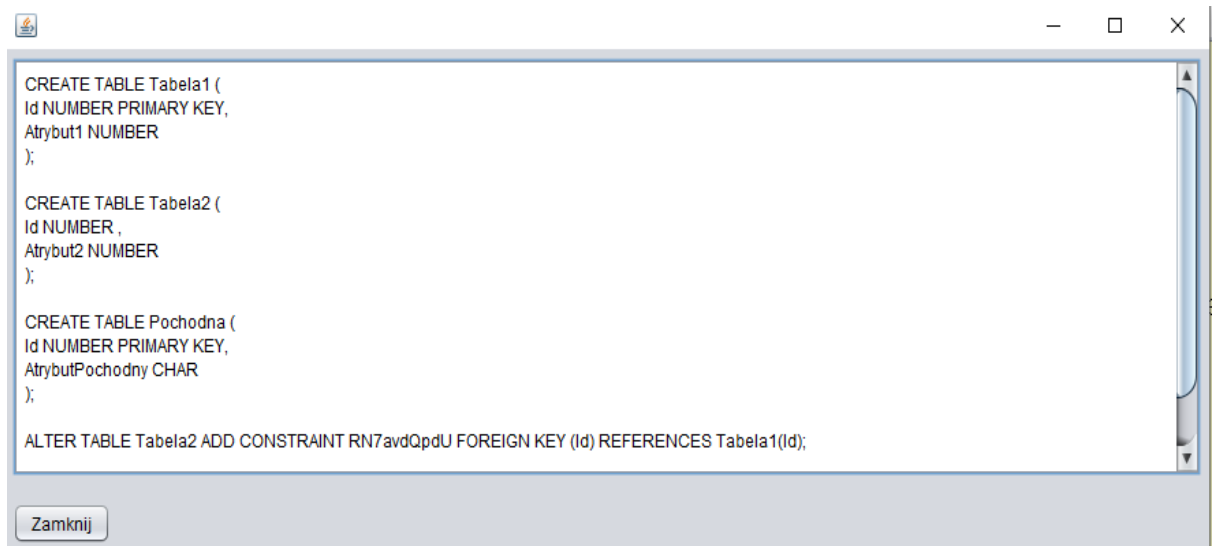
Kolejnym ważnym elementem w projektowaniu relacyjnej bazy danych jest możliwość połączenia tabel relacjami. W przypadku tej aplikacji należy nacisnąć jeden z przycisków “1_do_1”, “1_do_w” oraz “w_do_w”, jak i również istnieje możliwość utworzenia połączenia poprzez dziedziczenie w tym przypadku należy 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

4. Najważniejsze funkcje aplikacji

Tworzenie relacji między tabelami

```
Table table1=
searchTable(tableRelationComboBox1.getItemAt(tableRelationComboBox1.g
etSelectedIndex()));
Table table2=
searchTable(tableRelationComboBox2.getItemAt(tableRelationComboBox2.g
etSelectedIndex()));
Column column =
searchColumn(attributeComboBox1.getItemAt(attributeComboBox1.getSelec
tedIndex()),table1);
Column column2 =
searchColumn(attributeComboBox2.getItemAt(attributeComboBox2.getSelec
tedIndex()),table2);
JConnector connectLine,connectLine2;
if(table1!=null&&table2!=null&&table1!=table2) {
    if(relationTypeLabel.getText().equals("Jeden do wielu")) {
        connectLine = new JConnector(table2, table1, LINE_ARROW_DEST,
Color.BLACK);
        column.FKChangeStateToTrue();
        column.addRelation(table2,column2);
        column2.disablePK();
        saveRelationConectors(table1,connectLine,false);
        saveRelationConectors(table2,connectLine,false);
    }
    else if(relationTypeLabel.getText().equals("Jeden do jednego")) {
        connectLine = new JConnector(table2, table1, LINE_ARROW_NONE,
Color.BLACK);
        if(column2.getPKState()){
            column.FKChangeStateToTrue();
        }
    }
}
```

```

        column.addRelation(table2, column2);
        column2.disablePK();}
    else if(column.getPKState()) {
        connectLine = new JConnector(table1, table2, LINE_ARROW_NONE,
Color.BLACK);
        column.disablePK();
        column2.FKChangeStateToTrue();
        column2.addRelation(table1, column);
    }
    else{
        column.FKChangeStateToTrue();
        column.addRelation(table2, column2);
        column2.setPKState();
        column2.disablePK();
    }
    saveRelaationConectors(table1, connectLine, false);
    saveRelaationConectors(table2, connectLine, false);
}

    else {
        Table subTable = new Table(this);
        createSubTableForManyToManyRelation(subTable, table1, table2);
        connectLine2 = new JConnector(table1, subTable,
LINE_ARROW_DEST, Color.BLACK);
        connectLine = new JConnector(table2, subTable,
LINE_ARROW_DEST, Color.BLACK);
        diagramPanel.add(connectLine2);
        subTable.getjConnectors().add(connectLine);
        subTable.getjConnectors().add(connectLine2);
        subTable.setRelationDeleteMenuList(false);
    }
    diagramPanel.add(connectLine);
    diagramPanel.revalidate();
    diagramPanel.repaint();
    relationDialogWindow.setVisible(false);
}

```

Na samym początku działania tej funkcji znajdowane są obiekty dwóch wybranych tabel oraz obiekty kolumn, poprzez nazwy wybrane w oknie, w którym użytkownik określa które tabele mają być ze sobą w wybranej relacji. Kolejnym krokiem jest sprawdzenie, jaką relacje użytkownik chce utworzyć, w zależności od rodzaju relacji sposób utworzenia połączenia się różni. Dla relacji jeden do wielu kolumna wybrana w pierwszym wierszu zostaje kluczem obcym oraz zapisywana są obiekty tabeli i kolumny, z którą jest w relacji w liście, które przechowuje tego rodzaju dane, dodatkowo dla kolumny drugiej, która określona jest jako klucze główny, zablokowana zostaje możliwość zmiany tego stanu do momentów, w którym dana kolumna jest w relacji z inną tabelą. Relacja jeden do jednego jest tworzona w podobny sposób jak relacja jeden do wielu, z tym wyjątkiem, że zarówno tabela pierwsza, jak i druga może posiadać klucz główny, z tego powodu sprawdzane są wszystkie możliwości i klucz obcy przypisywany jest kolumnie, która nie jest określona jako klucz główny, dodatkowo w wypadku wyboru przez użytkownika dwóch

kolumn bez klucza głównego, klucz główny przypisywany jest do tabeli wybranej w drugim wierszu okna tworzenia relacji. Ostatnim rodzajem tworzonej relacji jest relacja wiele do wielu która różni się tym od pozostałych relacji, że tworzona jest pomiędzy wybranymi tabelami dodatkowa tabela, która zawiera w sobie kolumny z kluczami obcymi odwołującymi się do kolumn z dwóch wybranych tabel. Gdy użytkownik chce utworzyć relację wiele do wielu, w programie tworzony jest nowy obiekt klasy „Table”, który będzie służył jak tabela pomiędzy wybranymi tabelami, następnie w funkcji odpowiedzialnej za stworzenie tej tabeli, tworzone są obiekty kolumn oraz powiązania między tymi tabelami są zapisywane w odpowiednich listach.

Tworzenie dziedziczenia

```
Table table1 =
searchTable(inheritanceComboBox1.getItemAt(inheritanceComboBox1.getSelectedIndex()));
Table table2 =
searchTable(inheritanceComboBox2.getItemAt(inheritanceComboBox2.getSelectedIndex()));
JConnector connectLine;
if(table1!=null && table2!=null && table1!=table2) {
    if(table1.PKAvailable()) {
        connectLine = new JConnector(table2, table1,
        LINE_INHERITANCE, Color.BLACK);
        saveRelationConnectors(table1,connectLine,true);
        saveRelationConnectors(table2,connectLine,true);
        table1.getSubTables().add(table2);
        table2.getSuperTables().add(table1);
        if(!table2.PKAvailable()){
            creatColumnForSubTable(table2,table1);
        }
    }
    else{
        table2.getColumnWithPK().FKChangeStateToTrue();
    }
    table1.disablePKInTable();
    table2.disablePKInTable();
    diagramPanel.add(connectLine);
    diagramPanel.revalidate();
    diagramPanel.repaint();
    inheritanceDialog.setVisible(false);
}
```

Dla dziedziczenia tak sam, jak dla tworzenia relacji na samym początku wyszukiwane są obiekty dwóch wybranych przez użytkownika tabeli: tabeli bazowej oraz pochodnej. Aby utworzyć dziedziczenie pomiędzy dwoma wybranymi tabelami, użytkownik musi wybrać tabelę bazową, w której znajduje się klucz główny, bez spełnienia tego warunku program wyświetli informację, że wybraną złą tabelę. Natomiast w przypadku gdy tabela bazowa posiada kolumnę z kluczem głównym, to w liście dla tabeli bazowej

przechowywany jest obiekt tabeli pochodnej oraz tabela pochodna zapisuje w swojej liście obiekt tabeli bazowej. Kolejnym krokiem jest sprawdzenie, czy tabela pochodna posiada wcześniej utworzony klucz główny w przypadku braku klucza głównego, który mógłby służyć jako odwołanie do klucza głównego w tabeli bazowej, jak zakłada sposób przedstawienia dziedziczenia w kodzie SQL „Table per subclass”, jest on tworzony. W przypadku, gdy tabela pochodna posiada klucz główny to odwołanie do tabeli głównej, tworzone jest poprzez ten wcześniej zdefiniowany klucz główny. Ostatnim elementem jest zabezpieczenie kluczy głównych przed możliwością zmiany ich stanu po utworzeniu połączenia między dwoma wybranymi tabelami oraz wyświetlenie dziedziczenia w oknie aplikacji.

Generowanie kodu SQL

```
public String generateSqlTables(){
    String sql="";
    for(Table table :tables){
        sql= sql+"CREATE TABLE " +table.getTable_name()+" (\n";
        for(Column column:table.getColumns()){
            sql=sql+column.getName()+" "+column.getColumnType()+"
"+getConstrains(column)+"\n";
        }
        sql=sql.replaceAll(", $", "");
        sql=sql+")";\n\n";
    }
    return sql;
}
public String generateSqlForeignKey(){
    String sql="";

    for(Table table :tables) {
        for (Column column : table.getColumns()) {
            for (int i = 0; i < column.getRelation().size();
i++) {
                Table tmpTable = (Table)
column.getRelation().get(i).get(0);
                Column tmpColumn = (Column)
column.getRelation().get(i).get(1);
                sql = sql + "ALTER TABLE " +
table.getTable_name()+" ADD CONSTRAINT " + getRandomString(10) + "
FOREIGN KEY (" + column.getName() + ") "+ "REFERENCES " +
tmpTable.getTable_name() + "(" + tmpColumn.getName() + ");\n";
            }
        }
        sql=sql+"\n";
    }

    for(Table table :tables) {
        for (Table table1 : table.getSubTables()) {
            for(Column column:table.getColumns()) {
                if(column.getPKState()) {
                    sql = sql + "ALTER TABLE " + table1.getTable_name()
```

```

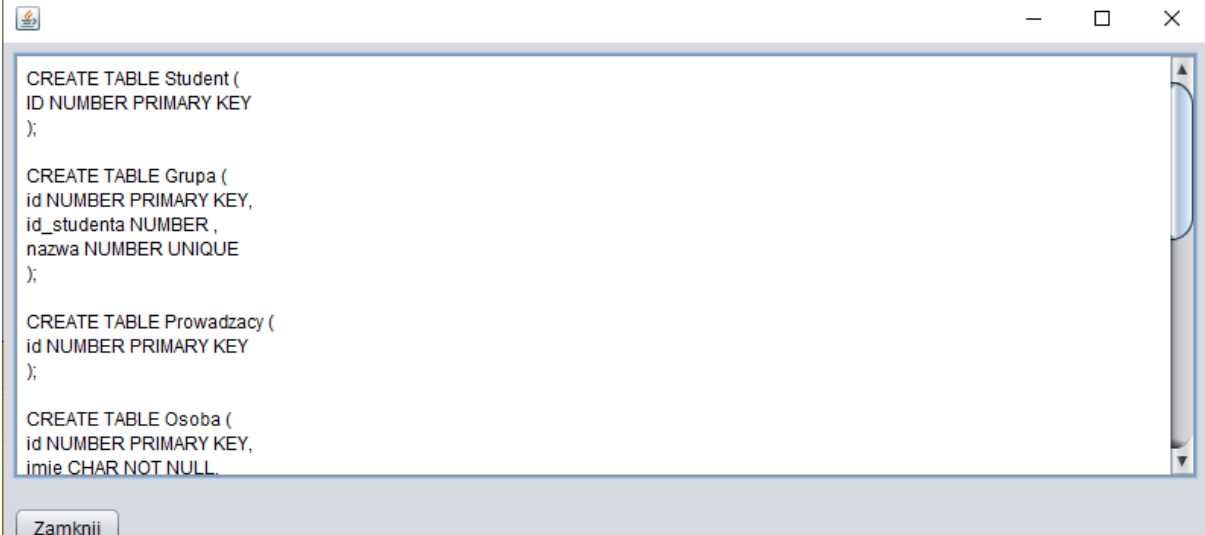
+ " ADD CONSTRAINT " + getRandomString(10) + " FOREIGN KEY (" +
table1.getPKName() + ") " + "REFERENCES " + table.getTableName() +
"(" + column.getName() + ");\n";
        break;
    }
}
    }
}
return sql;
}

```

Do prawidłowego wygenerowania kod SQL należy wywołać dwie metody „generateSQLTables” oraz „generateSQLForeignKey”. Pierwsza z nich służy do wygenerowania kodu potrzebnego do stworzenia tabel oraz ich kolumn wraz z ich typami, oraz własnościami. Działanie tej funkcji zaczyna się od pętli for, która przechodzi przez wszystkie obiekty klasy typu „Table” i dla każdej z nich tworzy ciąg znaków przedstawiający w SQL tworzenie tabeli o podanej nazwie, następnie wywoływana jest kolejna pętla for tym razem dla listy kolumn zapisanej w każdej tablicy i na podstawie tej listy tworzone są wpisy dotyczące kolumn ich typu oraz własności po zakończeniu tej pętli usuwany jest ostatni występujący przecinek. W ten sposób generowany jest kod SQL wykorzystywany do tworzenia tabel w bazie danych. Druga funkcja natomiast jest odpowiedzialna za wygenerowanie kod służącego do połączenia wcześniej wygenerowanych tabel. Jej działanie jest podzielone na dwie części pierwsza, część jest odpowiedzialna za relacje między tabelami, druga natomiast za dziedziczenie. W pierwszej części wyszukiwane są takie kolumny, które posiadają zapisywane dane dotyczące relacji z innymi tabelami, na tej podstawie generowane jest kod typu „ALTER TABLE” poprzez pobranie nazwy tabeli, która posiada kolumn z zapisanymi relacją, następnie dodawany jest losowy ciąg znaków składający się z małych i dużych liter, którego długość wynosi 10 znaków, kolejnymi krokami są pobranie nazwy tej kolumny oraz nazwy tabeli, oraz kolumny, z którym jest ona w relacji. Druga część służąca do przedstawienia dziedziczenia w kodzie SQL przechodzi przez wszystkie obiekty tabel pochodny i na ich podstawie generuje „ALTER TABLE” poprzez pobranie nazwy danej tabeli, wygenerowanie losowego ciągu znaków określającego daną zmianę tabeli, nazwę klucza głównego pochodnej tabeli, oraz nazwy tabeli bazowej i kolumny z kluczem głównym tej tabeli bazowej.

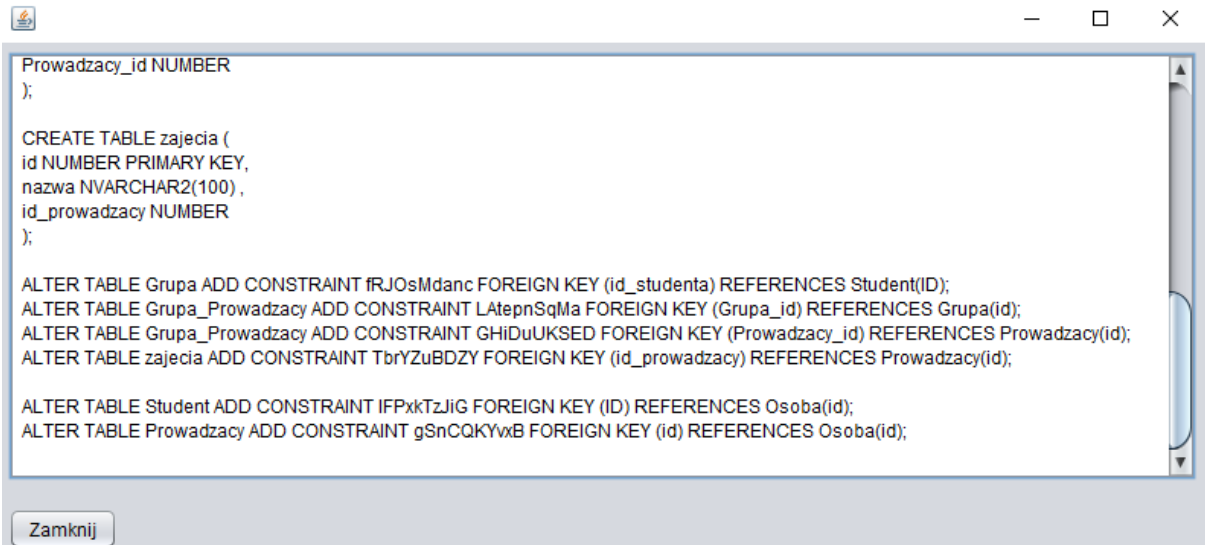
5. Poprawność wygenerowanego kodu

Zrzut ekranu przedstawiający wygenerowany kod



```
CREATE TABLE Student (  
ID NUMBER PRIMARY KEY  
);  
  
CREATE TABLE Grupa (  
id NUMBER PRIMARY KEY,  
id_studenta NUMBER ,  
nazwa NUMBER UNIQUE  
);  
  
CREATE TABLE Prowadzacy (  
id NUMBER PRIMARY KEY  
);  
  
CREATE TABLE Osoba (  
id NUMBER PRIMARY KEY,  
imie CHAR NOT NULL,
```

Zamknij



```
Prowadzacy_id NUMBER  
);  
  
CREATE TABLE zajecia (  
id NUMBER PRIMARY KEY,  
nazwa NVARCHAR2(100) ,  
id_prowadzacy NUMBER  
);  
  
ALTER TABLE Grupa ADD CONSTRAINT fRJOsMdanc FOREIGN KEY (id_studenta) REFERENCES Student(ID);  
ALTER TABLE Grupa_Prowadzacy ADD CONSTRAINT LAtepnSqMa FOREIGN KEY (Grupa_id) REFERENCES Grupa(id);  
ALTER TABLE Grupa_Prowadzacy ADD CONSTRAINT GHIDuUKSED FOREIGN KEY (Prowadzacy_id) REFERENCES Prowadzacy(id);  
ALTER TABLE zajecia ADD CONSTRAINT TbrYZuBDZY FOREIGN KEY (id_prowadzacy) REFERENCES Prowadzacy(id);  
  
ALTER TABLE Student ADD CONSTRAINT IFPxtZjIG FOREIGN KEY (ID) REFERENCES Osoba(id);  
ALTER TABLE Prowadzacy ADD CONSTRAINT gSnCQKYvxB FOREIGN KEY (id) REFERENCES Osoba(id);
```

Zamknij

Skopiowany kod z zrzutu ekranu

```
CREATE TABLE Student (  
ID NUMBER PRIMARY KEY  
);  
  
CREATE TABLE Grupa (  
id NUMBER PRIMARY KEY,  
id_studenta NUMBER ,  
nazwa NUMBER UNIQUE  
);
```



```

CREATE TABLE Prowadzacy (
id NUMBER PRIMARY KEY
);

CREATE TABLE Osoba (
id NUMBER PRIMARY KEY,
imie CHAR NOT NULL,
nazwisko VARCHAR2(100) NOT NULL
);

CREATE TABLE Grupa_Prowadzacy (
Grupa_id NUMBER ,
Prowadzacy_id NUMBER
);

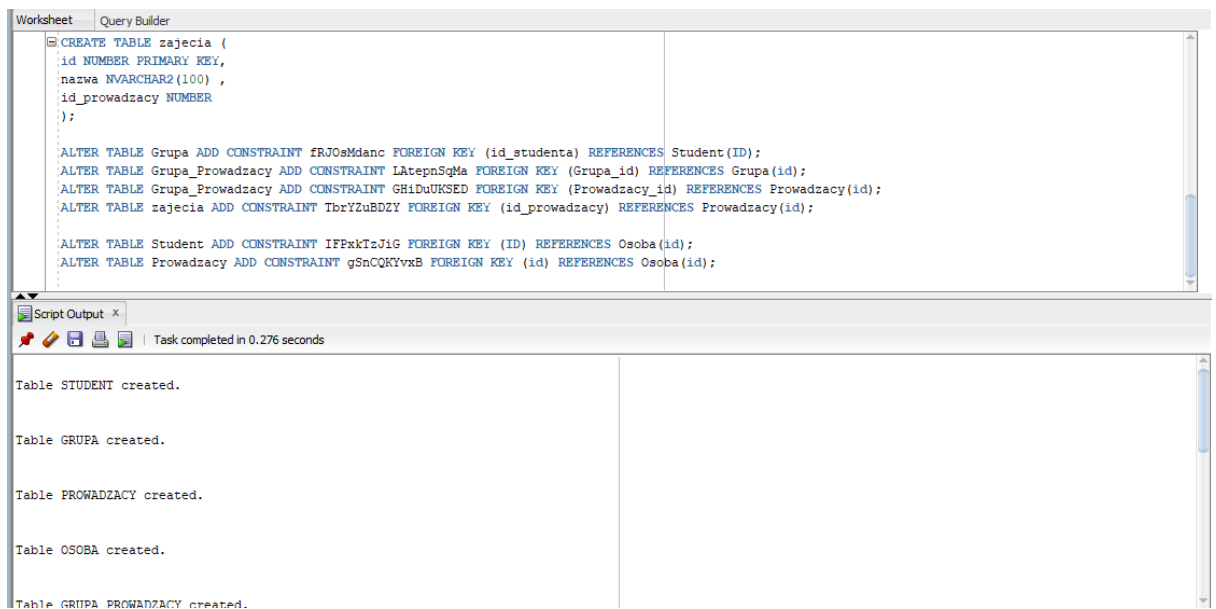
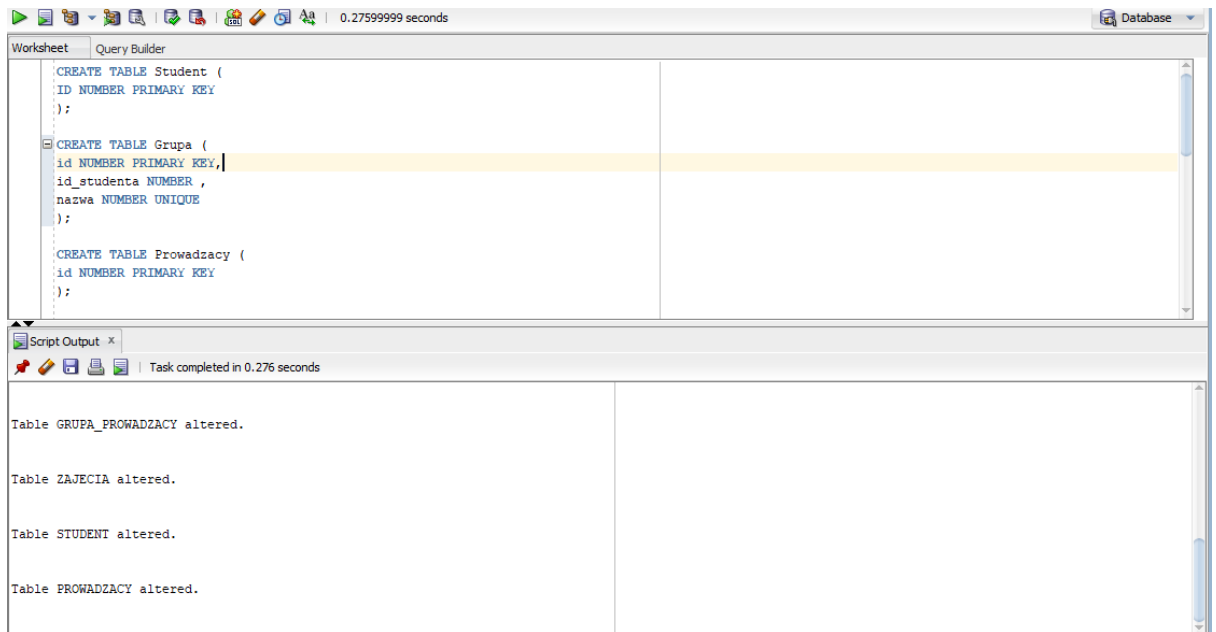
CREATE TABLE zajecia (
id NUMBER PRIMARY KEY,
nazwa NVARCHAR2(100) ,
id_prowadzacy NUMBER
);

ALTER TABLE Grupa ADD CONSTRAINT fRJOsMdanc FOREIGN KEY
(id_studenta) REFERENCES Student(ID);
ALTER TABLE Grupa_Prowadzacy ADD CONSTRAINT LAtepnSqMa FOREIGN
KEY (Grupa_id) REFERENCES Grupa(id);
ALTER TABLE Grupa_Prowadzacy ADD CONSTRAINT GHIDuUKSED FOREIGN
KEY (Prowadzacy_id) REFERENCES Prowadzacy(id);
ALTER TABLE zajecia ADD CONSTRAINT TbrYZuBDZY FOREIGN KEY
(id_prowadzacy) REFERENCES Prowadzacy(id);

ALTER TABLE Student ADD CONSTRAINT IFPxtZJiG FOREIGN KEY (ID)
REFERENCES Osoba(id);
ALTER TABLE Prowadzacy ADD CONSTRAINT gSnCQKYvxB FOREIGN KEY (id)
REFERENCES Osoba(id);

```

Zrzut ekran po wprowadzeniu wygenerowanego kod do sqldeveloper



Komunikaty potwierdzające utworzenia tabel

Table STUDENT created.
 Table GRUPA created.
 Table PROWADZACY created.
 Table OSOBA created.
 Table GRUPA_PROWADZACY created.
 Table ZAJECIA created.
 Table GRUPA altered.
 Table GRUPA_PROWADZACY altered.
 Table GRUPA_PROWADZACY altered.
 Table ZAJECIA altered.
 Table STUDENT altered.
 Table PROWADZACY altered.

6. Podsumowanie

Aplikacji opisane w tej dokumentacji pozwala na tworzenie graficznie schematu bazy danych wraz z możliwością obsługi relacji jeden do wielu, jeden do jednego oraz wiele do wielu, jak i również na przedstawienia dziedziczenia, które następnie jest zapisywane do kodu w SQL według schematu nazywanego „Table per subclass”, i na podstawie tego diagramu generowany jest kod SQL dla bazy danych typu Oracle. Użytkownik, korzystając z tej aplikacji, może modyfikować nazwy tabel, kolumn, jak i również je usuwać bez obaw, że może to spowodować jakieś problemy podczas generowania kodu, dodatkowo istnieje możliwość usunięcia relacji z innej tabelą dla danej wybranej tabeli, wszystkie elementy znajdują się w menu, które wyświetla się po naciśnięciu prawego przycisku myszy. Poprawność wygenerowanego kodu została sprawdzona w programie sqldeveloper, który utworzył bazę danych przedstawioną na schemacie programu.