UNIWERSYTET RZESZOWSKI
WYDZIAŁ MATEMATYCZNO-PRZYRODNICZY

Kierunek: Informatyka

Studia stacjonarne

PATRYK SADOK

PROJEKT: Baza danych czołgów Oracle

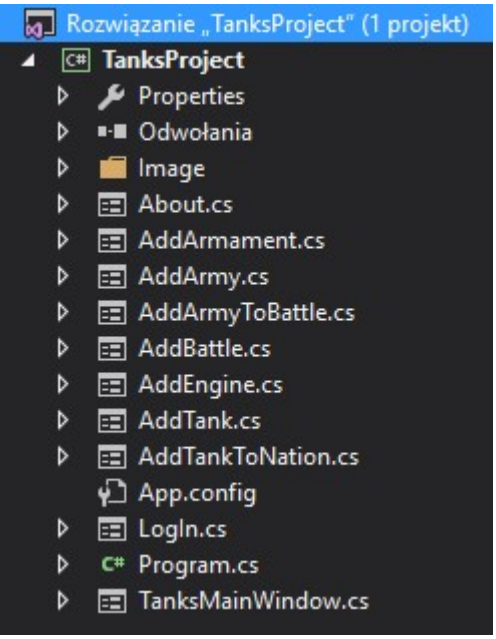
Projekt z przedmiotu: Bazy Danych II
Opiekun projektu: dr Barbara Pękala

Rzeszów 2017

Spis treści:

1. Zawartość płyty CD.

1.1 Katalog projektu Visual Studio 2017 Enterprise



Projekt zawiera:

- klasy formularzy służących do dodawania rekordów do bazy (*AddArmament, AddArmy, AddArmyToBattle, AddBattle, AddEngine, AddTank, AddTankToNation*),
- klasę główną, służącą do wyświetlania informacji bezpośrednio z bazy danych (*TanksMainWindow),*
- klasę wstępną, służącą do nawiązania połączenia z bazą oraz zalogowania się (*LogIn*).

1.2 Folder „Projekt Oracle 2017”

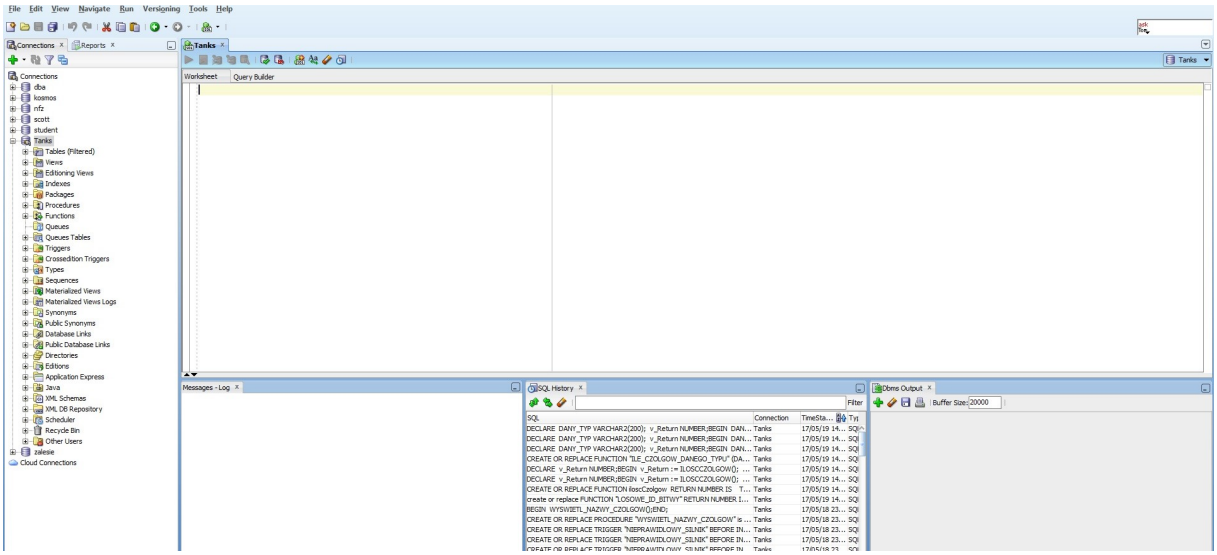
Folder ten zawiera:

- folder *Debug*, w którym znajduje się plik *TanksProject.exe* wraz z potrzebnymi bibliotekami oraz dokumentacją w formacie *XML*, program do prawidłowego funkcjonowania wymaga zainstalowania platformy *.NET* przynajmniej w wersji 4.6,
- folder projektowy *TanksProject*, utworzony za pomocą programu *Visual Studio 2017 Enterprise*, możliwy do otwarcia przez środowiska programistyczne obsługujące język *C#* oraz platformę *.NET*,
- plik eksportowy bazy *Tanks.sql*, utworzony dzięki programowi *SQL Developer*
- diagram *ERD* bazy danych,
- podręcznik użytkownika zapisany pod nazwą *UserGuide* w formacie *PDF.*

2. Wykorzystane technologie

2.1 Baza danych Oracle

Oracle Database – system służący do zarządzania relacyjnymi bazami danych, stworzony przez firmę *Oracle Corporation.* Relacyjna baza danych *Oracle* posługuje się standardowym językiem zapytań jakim jest *SQL*, posiada wbudowany wewnętrzny język tworzenia procedur *PL/SQL*.

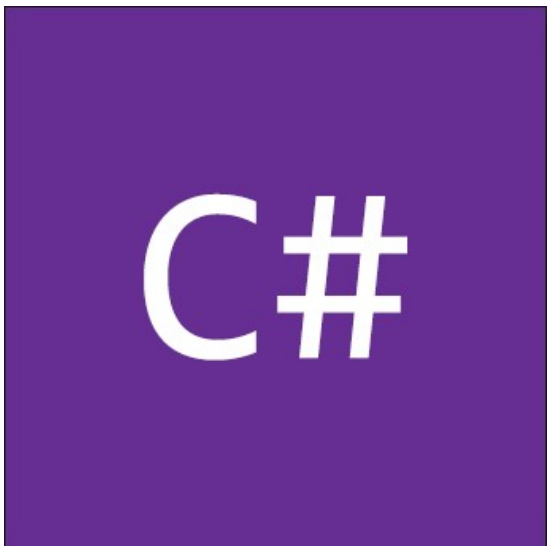


2.2 Język programowania C# wraz z platformą .NET

C# jest obiektowym językiem programowania zaprojektowanym w latach 1998-2001 dla firmy *Microsoft*. Język ten cechuje się obiektowością z hierarchią o elemencie nadrzędnym, kod programu jest zbiorem klas, posiada mechanizm *GarbageCollector*, który skutecznie czyści pamięć operacyjną komputera z nieużywanych obiektów, pozwala on też na dynamiczne tworzenie kodu w czasie działanie programu oraz włączanie tegoż kodu do aktualnie wykonywanego kodu. Język *C#* posiada też bogatą ilość bibliotek, pozwalającą tworzyć programy na wiele platform, zarówno mobilnych jak i desktopowych.

.NET Framework jest natomiast platformą programistyczną opracowaną przez *Microsoft*. Obejmuje środowisko uruchomieniowe oraz biblioteki klas dostarczające standardowej funkcjonalności dla aplikacji. Platforma nie jest ta powiązana ściśle z jednym językiem, pozwala tworzyć programy w językach takich jak *C++, C#, F#, J#, Delphi, Visual BASIC .NET*. Zadaniem tej platformy jest zarządzanie różnymi elementami systemu: kodem aplikacji, pamięcią i zabezpieczeniami.

Flagowym produktem pozwalającym na programowanie w języku *C#* na platformie *.NET* jest *Microsoft Visual Studio* (używany również do stworzenia programu na potrzeby projektu). Obecnie platforma *.NET* posiada wersję oznaczoną numerem 4.7.

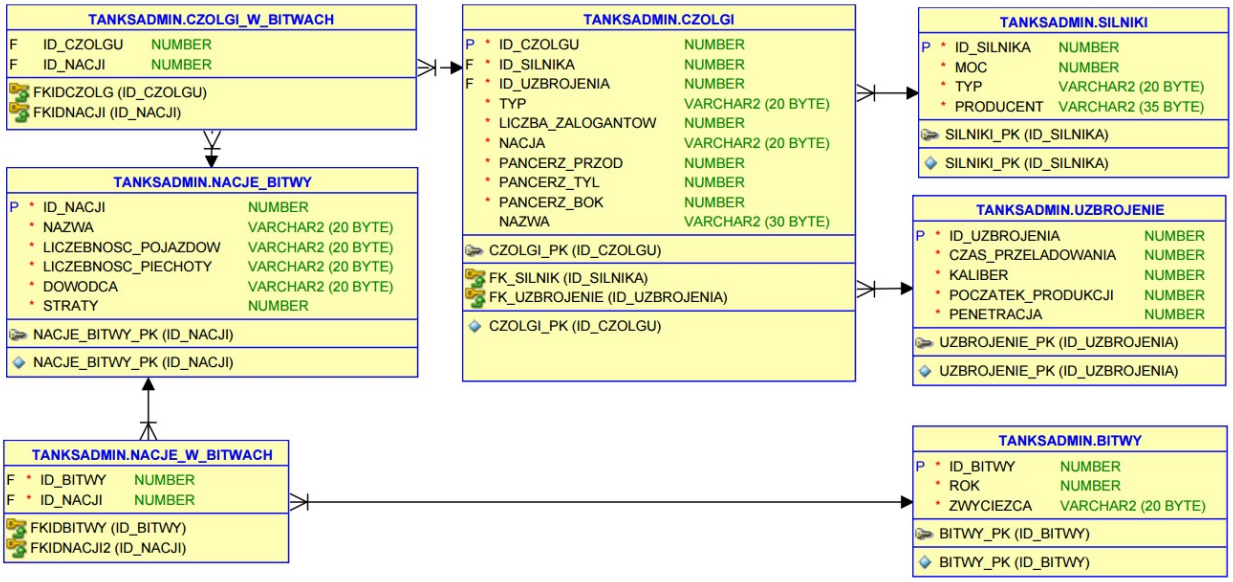


2.3 Biblioteka WinForms będąca częścią platformy .NET

Windows Forms – jest to nazwa interfejsu programowania graficznych aplikacji w ramach *Microsoft .NET Framework*, umożliwia natywny dostęp do elementów graficznego interfejsu *Microsoft Windows*.

3. Omówienie zawartości bazy danych

3.1 Diagram ERD bazy



3.2 Tabele

Jak widać na diagramie, w bazie danych znajduje się 5 tabel głównych, 2 tabele łączące, służące do wspomagania relacji wiele-do-wielu. Każda tabela posiada właściwości, które uniemożliwiają dodania pustych rekordów do bazy lub skasowania rekordów, które mają powiązania z innymi rekordami. Baza zawiera dane o czołgach, które posiadają charakterystyczne silniki oraz uzbrojenie. Dodatkowo, czołgi na przestrzeni czasu mogły być w różnych dywizjach pancernych, te zaś mogły brać udział w różnych bitwach. Dane czołgów zostały wprowadzone w oparciu o realne dane, natomiast dane dot. Oddziałów oraz bitew zostały spreparowane w celu potwierdzenia funkcjonowania relacji między tabelami.

3.3 Wyjaśnienie funkcji, procedur oraz triggerów

- funkcja zwracająca ilość czołgów w bazie o podanym typie:

```
  CREATE OR REPLACE EDITIONABLE FUNCTION "TANKSADMIN"."ILE_CZOLGOW_DANEGO_TYPU" (DANY_TYP varchar2) RETURN number is
LICZBA_CZOLGOW number;
BEGIN
  Select count(*) into LICZBA_CZOLGOW from CZOLGI where TYP = DANY_TYP;
  Return LICZBA_CZOLGOW;
END ILE_CZOLGOW_DANEGO_TYPU;
```

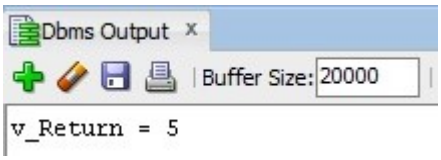
Użycie:

```
DECLARE
  DANY_TYP VARCHAR2(200);
  v_Return NUMBER;
BEGIN
  DANY_TYP := 'Sredni';

  v_Return := ILE_CZOLGOW_DANEGO_TYPU(
    DANY_TYP => DANY_TYP
  );
  /* Legacy output: */
DBMS_OUTPUT.PUT_LINE('v_Return = ' || v_Return);
  :v_Return := v_Return;
END;
```

W miejscu DANY_TYP := []; należy wpisać pożądany typ czołgu, dostępne typy w bazie to: Lekki, Sredni, Ciezki.

Wyjście:



- funkcja zwracająca maksymalne ID spośród tabeli czołgów:

```
  CREATE OR REPLACE EDITIONABLE FUNCTION "TANKSADMIN"."MAKS_ID_CZOLGU" RETURN NUMBER IS
id_number NUMBER;
BEGIN
  select max(ID_CZOLGU) into id_number from CZOLGI;
  if id_number is null then
  id_number:=0;
  end if;
  return id_number;
END;
```

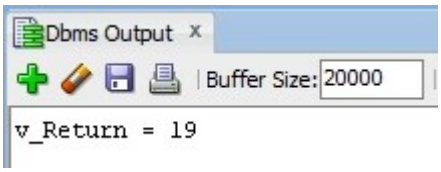
Użycie:

```
DECLARE
  v_Return NUMBER;
BEGIN

  v_Return := MAKS_ID_CZOLGU();
  /* Legacy output: */
DBMS_OUTPUT.PUT_LINE('v_Return = ' || v_Return);

  :v_Return := v_Return;
END;
```

Wyjście:



Jak widać, maksymalne ID Czołgu znajdującego się w bazie wynosi 19.

- funkcja zwracająca maksymalne ID spośród tabeli silników, stworzona na potrzeby procedury pokazanej później:

```
  CREATE OR REPLACE EDITIONABLE FUNCTION "TANKSADMIN"."MAKS_ID_SILNIKA" RETURN NUMBER IS
id_number NUMBER;
BEGIN
  select max(ID_SILNIKA) into id_number from SILNIKI;
  if id_number is null then
  id_number:=0;
  end if;
  return id_number;
END;
```

- funkcja zwracająca średni kaliber spośród wszystkich dział umieszczonych w tabeli UZBROJENIE:

```
  CREATE OR REPLACE EDITIONABLE FUNCTION "TANKSADMIN"."SREDNI_KALIBER" RETURN number is
z_kaliber uzbrojenie.kaliber%type;
BEGIN
  Select AVG(kaliber) into z_kaliber from uzbrojenie;
  Return z_kaliber;
END SREDNI_KALIBER;
```

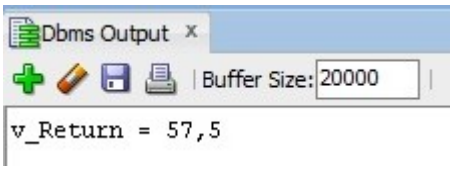
Użycie:

```
DECLARE
  v_Return NUMBER;
BEGIN

  v_Return := SREDNI_KALIBER();
  /* Legacy output: */
DBMS_OUTPUT.PUT_LINE('v_Return = ' || v_Return);

  :v_Return := v_Return;
END;
```

Wyjście:



Jak widać, średni kaliber wszystkich dział wynosi 57,5 mm.

- procedura dodawania nowego silnika do bazy danych:

```
set define off;

  CREATE OR REPLACE EDITIONABLE PROCEDURE "TANKSADMIN"."DODAJ_NOWY_SILNIK" (
  p_Moc  IN      silniki.moc%TYPE,
  p_Typ IN   silniki.typ%TYPE,
  p_Producent IN      silniki.producent%TYPE,
  p_wynik OUT varchar2) AS
ID number;
id_silnik silniki.id_silnika%type;
BEGIN
  SELECT maks_id_silnika into ID FROM dual;
  id_silnik := 1+ID;
  INSERT INTO silniki (id_silnika, moc,typ,producent) VALUES (id_silnik,p_Moc,p_Typ,p_Producent);
  p_wynik := 'W bazie umieszczono nowy silnik o ID='||id_silnik;
  COMMIT;
EXCEPTION
    WHEN OTHERS THEN p_wynik := 'Wystąpił błąd';
END DODAJ_NOWY_SILNIK;
```

Procedura ta wywołuje funkcję, która sprawdza maksymalne ID spośród tabeli silników, co pełni rolę autoinkrementacji ID.

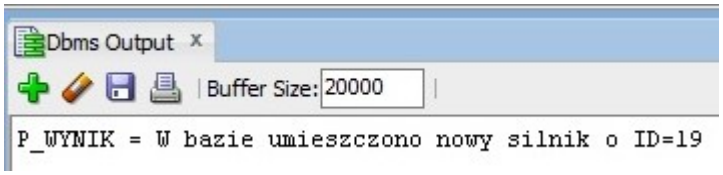
Użycie:

```
DECLARE
  P_MOC NUMBER;
  P_TYP VARCHAR2(20);
  P_PRODUCENT VARCHAR2(35);
  P_WYNIK VARCHAR2(200);
BEGIN
  P_MOC := 999;
  P_TYP := 'Benzyna';
  P_PRODUCENT := 'Mercedes';

  DODAJ_NOWY_SILNIK(
    P_MOC => P_MOC,
    P_TYP => P_TYP,
    P_PRODUCENT => P_PRODUCENT,
    P_WYNIK => P_WYNIK
  );
  /* Legacy output: */
DBMS_OUTPUT.PUT_LINE('P_WYNIK = ' || P_WYNIK);

  :P_WYNIK := P_WYNIK;
END;
```

Wyjście:



Jak widać, pomyślnie dodany został nowy rekord do tabeli SILNIKI o ID_SILNIKA wynoszącym 19.

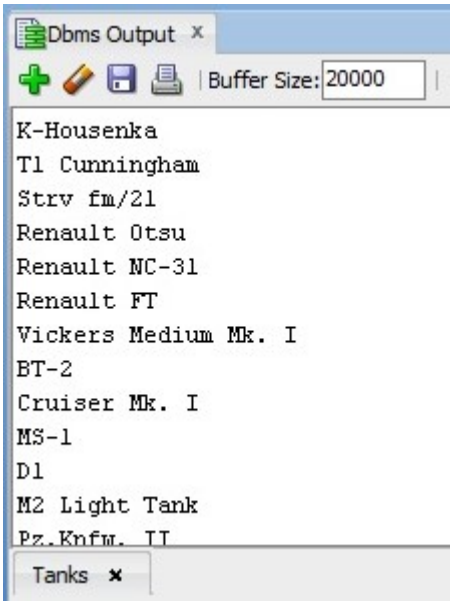
- procedura wyświetlająca nazwy wszystkich czołgów spośród tabeli CZOŁGI:

```
CREATE OR REPLACE EDITIONABLE PROCEDURE "TANKSADMIN"."WYSWIETL_NAZWY_CZOLGOW" is
    cursor allTanks is select * from czolgi;
    currentTank allTanks%ROWTYPE;
begin
    open allTanks;
    loop
      fetch allTanks into currentTank;
      exit when allTanks%NOTFOUND;
      DBMS_OUTPUT.put_line(currentTank.nazwa);
    end loop;
    close allTanks;
end;
```

Użycie:

```
BEGIN
  WYSWIETL_NAZWY_CZOLGOW();
END;
```

Wyjście:



- procedura zwiększająca wartość kolumny LICZEBNOSC_POJAZDOW w tabeli NACJE_BITWY:

```
  CREATE OR REPLACE EDITIONABLE PROCEDURE "TANKSADMIN"."Zwieksz_sily_bitewne" (
  p_Liczebnosc_Pojazdow IN  NACJE_BITWY.liczebnosc_pojazdow%TYPE,
  p_wynik OUT varchar2) AS
BEGIN
  update nacje_bitwy set liczebnosc_pojazdow = (liczebnosc_pojazdow + p_Liczebnosc_Pojazdow);
  p_wynik := 'Zwiekszono oddzialy pancerne o '||p_Liczebnosc_Pojazdow||' pojazdow';
  COMMIT;
EXCEPTION
    WHEN OTHERS THEN p_wynik := 'Wystąpił błąd';
END "Zwieksz_sily_bitewne";
```

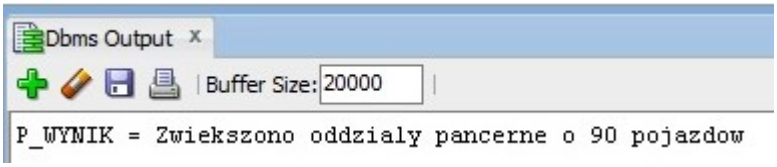
Użycie:

```
DECLARE
  P_LICZEBNOSC_POJAZDOW VARCHAR2(20);
  P_WYNIK VARCHAR2(200);
BEGIN
  P_LICZEBNOSC_POJAZDOW := 90;

  "Zwieksz_sily_bitewne"(
    P_LICZEBNOSC_POJAZDOW => P_LICZEBNOSC_POJAZDOW,
    P_WYNIK => P_WYNIK
  );
  /* Legacy output: */
DBMS_OUTPUT.PUT_LINE('P_WYNIK = ' || P_WYNIK);

  :P_WYNIK := P_WYNIK;
END;
```

Wynik:



- trigger uniemożliwiający dodanie działa do tabeli UZBROJENIE o kalibrze mniejszym niż 45 mm:

```
  CREATE OR REPLACE EDITIONABLE TRIGGER "TANKSADMIN"."ZBYT_MALY_KALIBER"
BEFORE INSERT OR UPDATE
of kaliber
on UZBROJENIE
for each row
declare
  v_error VARCHAR2(2000);
begin
  if :new.kaliber < 45
    then
      v_error:='Dzialo o ID='||:old.id_uzbrojenia||' posiada zbyt maly kaliber, minimum wynosi 45 mm.';
      raise_application_error(-20999,v_error);
    end if;
end;
/
ALTER TRIGGER "TANKSADMIN"."ZBYT_MALY_KALIBER" ENABLE;
```

- trigger uniemożliwiający dodanie do bazy silnika o innym paliwie niż ‘Benzyna’ oraz ‘Diesel’ z powodów historycznych:

```
  CREATE OR REPLACE EDITIONABLE TRIGGER "TANKSADMIN"."NIEPRAWIDLOWY_SILNIK"
BEFORE INSERT OR UPDATE
of typ
on silniki
for each row
declare
  v_error VARCHAR2(2000);
begin
  if initcap(:new.typ)!='Benzyna' and initcap(:new.typ)!='Diesel'
    then
      v_error:='Silnik o ID= '||:old.id_silnika||' posiada nieprawidowy typ. Dopuszczalne typy to: Benzyna, Diesel.';
      raise_application_error(-20999,v_error);
    end if;
end;
/
ALTER TRIGGER "TANKSADMIN"."NIEPRAWIDLOWY_SILNIK" ENABLE;
```

4. Omówienie kodu aplikacji

4.1 Klasy odpowiadające za obsługę bazy danych

W moim projekcie, główną klasą odpowiadającą za połączenie z bazą jest *TanksMainWindow*. Powoduje ona połączenie się z bazą na podstawie podanego hosta, portu, loginu i hasła administratora. Po połączeniu lub w przypadku jakiegokolwiek błędu odpowiednie komunikaty wyświetlane są na specjalnie przygotowanej konsoli. Do połączenia się z bazą użyta została biblioteka *System.Data.OracleClient*, będąca częścią pakietu *.NET Framework*.

```
/// <summary>
/// Konstruktor formularza okna głównego, dodatkowo nawiązuje automatycznie połączenie z bazą
/// </summary>
public TanksMainWindow()
{

    InitializeComponent();
    textBox_console.ReadOnly = true;
    string strCon = @"Data Source=(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=TCP)(HOST=localhost)(PORT=1521)))
                      (CONNECT_DATA=(SERVER=DEDICATED)(SERVICE_NAME=Tanks)));
                      User Id = tanksadmin; Password = tankspass1; ";
    string text;
    this.FormBorderStyle = FormBorderStyle.FixedDialog;

    try
    {
        oracleConnection = new OracleConnection(strCon);
       oracleConnection.Open();
        text = "Nawiązano połączenie z bazą danych.";
        textBox_console.AppendText("\n");
        textBox_console.AppendText(text);
        textBox_console.AppendText("\n");
    }
    catch (Exception ex)
    {
        textBox_console.AppendText("\n");
        textBox_console.AppendText(ex.Message);
        textBox_console.AppendText("\n");
    }


}
```

Połączenie jest nawiązywane od razu po uruchomieniu formularza. Blok Try-Catch pozwala przechwycić jakiekolwiek błędy, które mogą wystąpić w trakcie użytkowania programu, a następnie wyświetlić ich komunikaty.

Aktualizacja danych w bazie:

Aktualizacja również odbywa się za pomocą klasy *TanksMainWindow*, ponieważ to właśnie tam znajduje się tabela wyświetlająca dane.

Kod funkcji, która obsługuje aktualizację bazy:

```
/// <summary>
/// Metoda aktualizująca bazę danych na podstawie edycji wybranego w tabeli rekordu
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void button_updateArmy_Click(object sender, EventArgs e)
{
    try
    {
        int indexRow = dataGridView1.CurrentCell.RowIndex;
        int indexColumn = 0;
        String IndeksTabeli = dataGridView1.CurrentCell.RowIndex.ToString();
        String daneZTabeli = dataGridView1.Rows[indexRow].Cells[indexColumn].Value.ToString();
        OracleCommand cmd = new OracleCommand("update nacje_bitwy set nazwa='" + dataGridView1.Rows[indexRow].Cells[1].Value.ToString() + "', liczebn
        cmd.ExecuteNonQuery();
        textBox_console.AppendText("\n");
        textBox_console.AppendText("Zapytanie wykonane pomyślnie.");
        textBox_console.AppendText("\n");
    }
    catch (Exception ex)
    {
        textBox_console.AppendText("\n");
        textBox_console.AppendText(ex.Message);
        textBox_console.AppendText("\n");
    }
}
```

Dzięki możliwości zaprogramowania dynamicznej tabeli, która w przeciwieństwie do języka *Java* wyświetla dane bezpośrednio z bazy danych (W *Javie* wymagane jest zaimportowanie bazy danych do kolekcji, a następnie ręczne spreparowanie odpowiadających im tabel, natomiast *C#* z pomocą pewnych bibliotek umożliwia bezpośredni odczyt danych do adaptera). Aktualizacja w programie odbywa się poprzez edycję tabeli wybranego rekordu a następnie użycia przycisku „Aktualizuj”, powoduje to złożenie zapytania *UPDATE*, które aktualizuje rekord o danym ID w wierszu tabeli. Zaletą takiego rozwiązania jest z pewnością zwięzłość kodu, jednakże ma to swój mankament – zaprogramowanie aktualizacji całej bazy danych w taki sposób, aby można było edytować dowolną ilość wierszy a następnie aktualizować bazę wymagałaby złożenia dużego zapytania, które musiałoby uwzględniać wszystkie rekordy w danej tabeli.

Dodawanie danych do bazy:
Dodawanie danych odbywa się w innych formularzach niż *TanksMainWindow*, jest to celowy zabieg mający na celu zapobiegnięcie powstania zbyt dużej ilości funkcjonalności w jednym oknie, co sprawiłoby, iż program nie byłby czytelny oraz intuicyjny w obsłudze.
Użycie przycisku „Dodaj…” w programie powoduje wyświetlenie odpowiedniego formularza, który posiada pole tekstowe. Do pól należy wprowadzić odpowiednie dane, a następnie użyć przycisku „Dodaj”.

Kod funkcji dodawania rekordu:

```
/// <summary>
/// Metoda dodająca rekord do bazy danych
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void buttonAdd_Click(object sender, EventArgs e)
{
    try
    {
        if (textEngineID.Text.Equals(null) || textPower.Text.Equals(null) || textProducent.Text.Equals(null) || textType.Text.Equals(null))
        {
            throw new ArgumentNullException();
        }
        int n1, n2;
        Boolean isNumeric1 = int.TryParse(textEngineID.Text, out n1);
        Boolean isNumeric2 = int.TryParse(textPower.Text, out n2);

        String zapytanieSprawdzajace = "SELECT * FROM SILNIKI WHERE ID_SILNIKA LIKE " + textEngineID.Text;
        OracleCommand test = new OracleCommand(zapytanieSprawdzajace, localOracleConnection);
        int sqlRezultat = test.ExecuteNonQuery();
        if (isNumeric1 || isNumeric2 || n1 > 0 || n2 > 0 || sqlRezultat < 0)
        {
            Zapytanie = "INSERT INTO SILNIKI VALUES(" + textEngineID.Text + ", " + textPower.Text + ", '" + textType.Text + "', '"+textProducent.Text
            OracleCommand insert = new OracleCommand(Zapytanie, localOracleConnection);
            insert.ExecuteNonQuery();
            consoleBox.AppendText("\n");
            consoleBox.AppendText(Zapytanie + " zostało wykonane pomyślnie.");
            consoleBox.AppendText("\n");
        }
    }
    catch (ArgumentNullException ane)
    {
        consoleBox.AppendText("\n");
        consoleBox.AppendText("Któryś z argumentów jest pusty. Sprawdź dane wejściowe.");
        consoleBox.AppendText("\n");
    }
    catch (Exception ex)
    {
        consoleBox.AppendText("\n");
        consoleBox.AppendText(ex.Message);
        consoleBox.AppendText("\n");
    }
}
```

Jak widać, nie umieściłem tutaj walidacji dodawanych rekordów, ponieważ:
- w przypadku niektórych danych, jak nazwy czołgów, producentów silników etc pochodzą z różnych narodowości, niemożliwym jest więc odpowiednie ich zwalidowanie,
- dane są walidowane bezpośrednio przez bazę danych, jeżeli rekord nie jest dodany, to konsola wyświetla odpowiedni komunikat.

Usuwanie rekordów:
Opcję usuwania zamieściłem tylko w przypadku tabel łączących z kilku powodów:
- w mojej bazie praktycznie wszystkie tabele udostępniają swoje ID innym tabelom,
- niemożliwe jest tutaj usunięcie rekordu, który udostępnia swoje ID innej tabeli jako klucz obcy,
- dodatkowo, zamiast usuwać np. rekord związany z czołgiem można go w łatwy sposób edytować.

Usuwanie rekordów z tabel łączących odbywa się w klasie *TanksMainWindow*.

Kod usuwania rekordu z bazy danych:

```
/// <summary>
/// Metoda aktualizująca bazę danych na podstawie edycji wybranego w tabeli rekordu
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void buttonDeleteTankNation_Click(object sender, EventArgs e)
{
    try
    {
        int indexRow = dataGridView1.CurrentCell.RowIndex;
        int indexColumn = 0;
        String IndeksTabeli = dataGridView1.CurrentCell.RowIndex.ToString();
        String daneZTabeli = dataGridView1.Rows[indexRow].Cells[indexColumn].Value.ToString();
        OracleCommand cmd = new OracleCommand("delete czolgi_w_bitwach where ID_czolgu=" + dataGridView1.Rows[indexRow].Cells[0].Value.ToString() +
        cmd.ExecuteNonQuery();
        textBox_console.AppendText("\n");
        textBox_console.AppendText("Zapytanie wykonane pomyślnie.");
        textBox_console.AppendText("\n");
    }
    catch (Exception ex)
    {
        textBox_console.AppendText("\n");
        textBox_console.AppendText(ex.Message);
        textBox_console.AppendText("\n");
    }
}
```

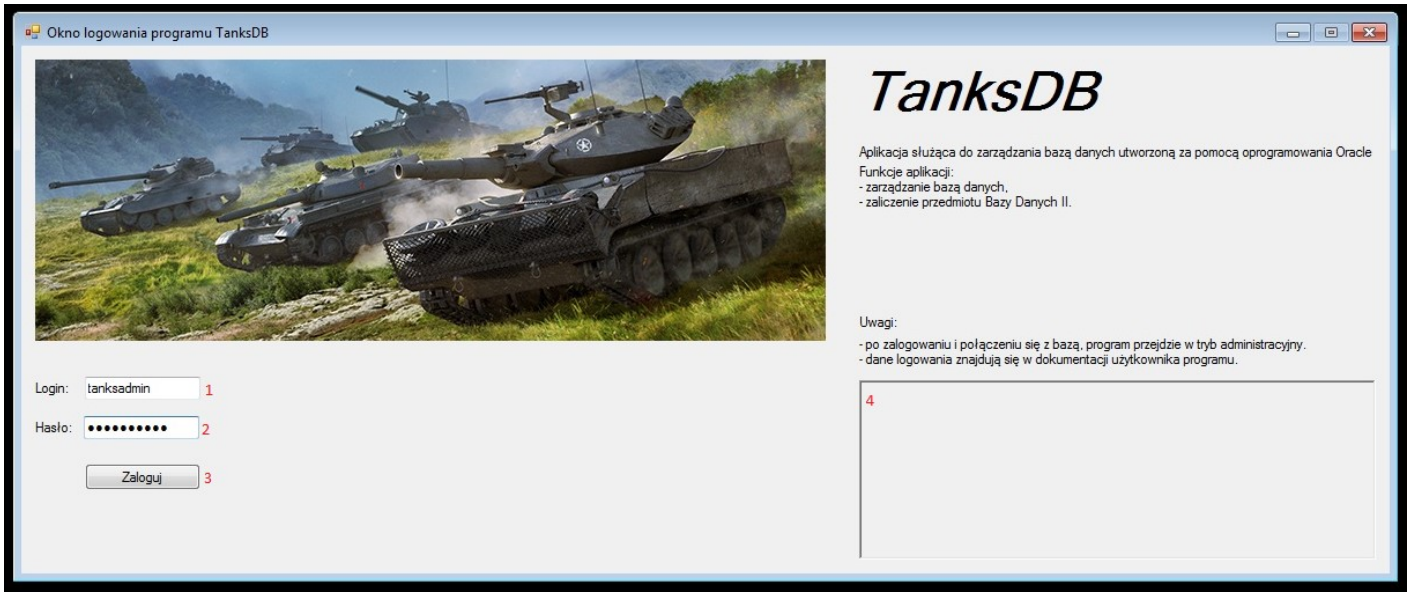
Usuwanie odbywa się podobnie jak aktualizacja, wg. ID z wybranego rekordu w bazie danych.

4.2 Klasy odpowiadające za graficzny interfejs

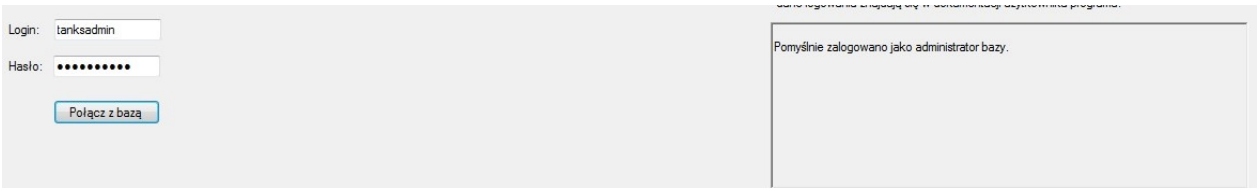
W projekcie każda klasa odpowiada za pewien formularz, przykładowy kod został wklejony wcześniej, przy sekcji połączenia się z bazą danych, gdzie konstruktor formularza automatycznie wywoływał połączenie.

5. Wyjaśnienie przykładowych funkcji programu

Logowanie do bazy danych:

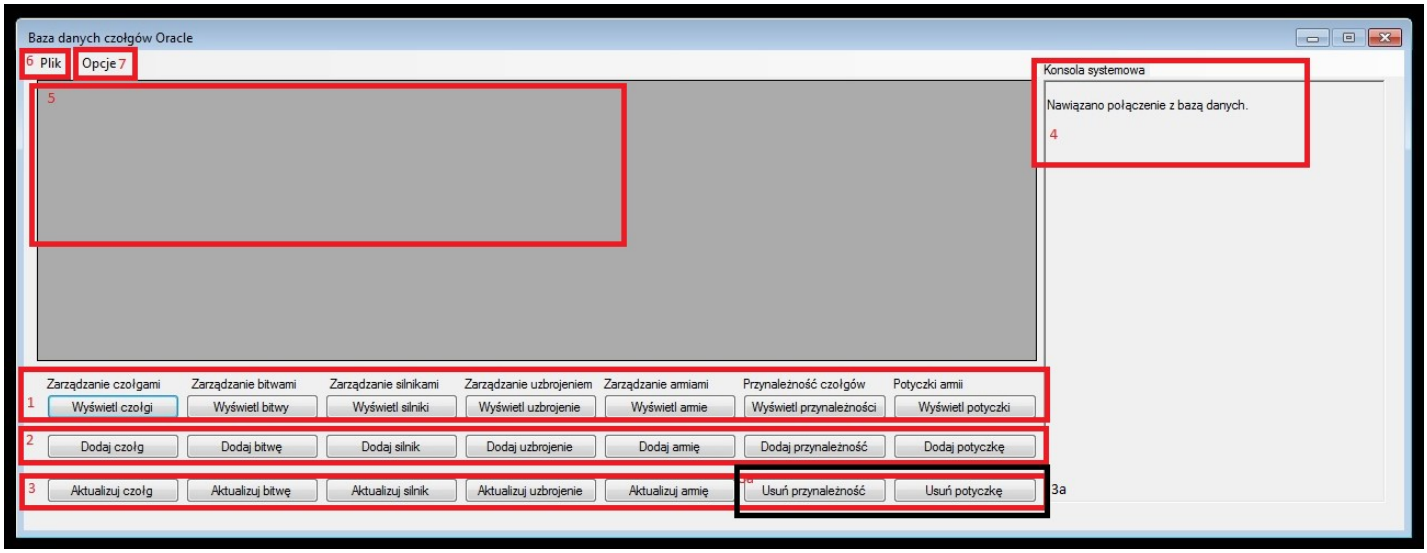


Logowanie odbywa się poprzez wpisanie poprawnego loginu do pola tekstowego oznaczonego numerem „1”, oraz hasła do pola tekstowego oznaczonego numerem „2”.
Login: tanksadmin
Hasło: tankspass1
Następnie, po zalogowaniu za pomocą użycia przycisku oznaczonego numerem „3” w konsoli oznaczonej numerem „4” pojawi się stosowny komunikat.
Uwaga: jeśli podany login lub hasło będą nieprawidłowe, program nie zaloguje się do bazy i połączenie będzie niemożliwe.



Jak widać na powyższym zrzucie ekranu, po poprawnym zalogowaniu się w konsoli został wyświetlony komunikat, na dodatek przycisk „Zaloguj” został zmieniony na przycisk „Połącz z bazą”. Jego użycie spowoduje wyłączenie aktualnego okna, włączenie okna głównego oraz nawiązanie połączenia.

Okno główne programu:



Sekcja okna głównego jest podzielona na następujące części:

1 – Sekcja wyświetlania, użycie jakiegokolwiek przycisku spowoduje wykonanie zapytania, które ma za zadanie wyświetlić dane z odpowiedniej tabeli w bazie w tabeli aplikacji, która znajduje się w sekcji nr. 5.
Przykład wyświetlonych danych:

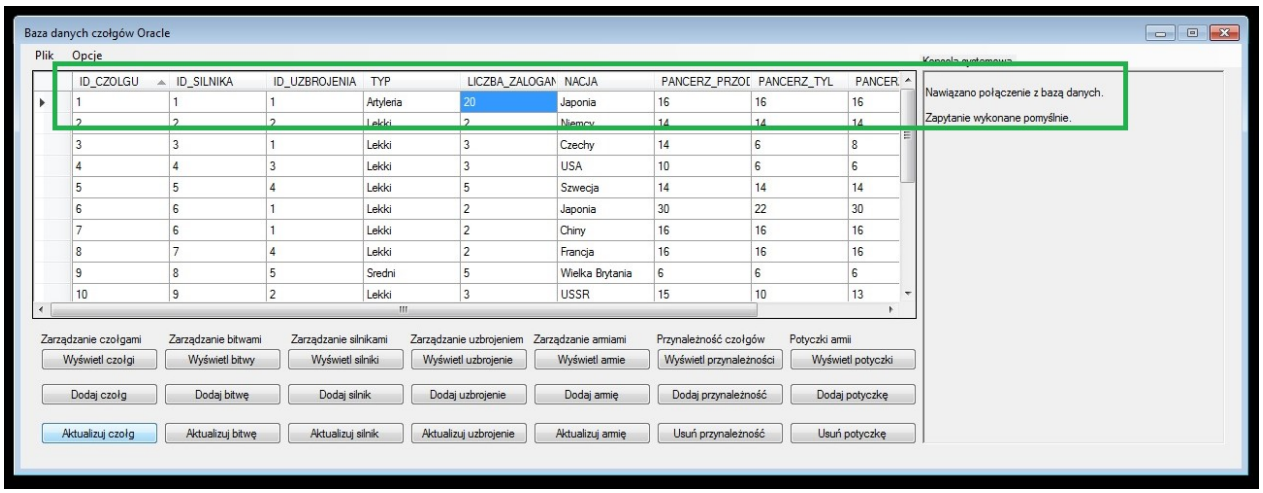
ID_CZOLGU	ID_SILNIKA	ID_UZBROJENIA	TYP	LICZBA_ZALOGAN	NACJA	PANCERZ_PRZOD	PANCERZ_TYL	PANCER
1	1	1	Lekki	2	Japonia	16	16	16
2	2	2	Lekki	2	Niemcy	14	14	14
3	3	1	Lekki	3	Czechy	14	6	8
4	4	3	Lekki	3	USA	10	6	6
5	5	4	Lekki	5	Szwecja	14	14	14
6	6	1	Lekki	2	Japonia	30	22	30
7	6	1	Lekki	2	Chiny	16	16	16
8	7	4	Lekki	2	Francja	16	16	16
9	8	5	Sredni	5	Wielka Brytania	6	6	6
10	9	2	Lekki	3	USSR	15	10	13

Tabela wyświetla wyniki posortowane wg. Głównego ID w każdej tabeli, dodatkowo klikając na nazwy kolumn można sortować całą tabelę wg. Innych właściwości, np. ID silnika, typu czołgu.

Wyświetlenie innej tabeli z bazy spowoduje zresetowanie tabeli aplikacji tak, aby tabele z bazy nie nakładały się na siebie.

Sekcja 3: aktualizacja.
Odbywa się ona poprzez edycję danego rekordu bezpośrednio w tabeli, a następnie

kliknięciu przycisku odpowiadającego danej sekcji:



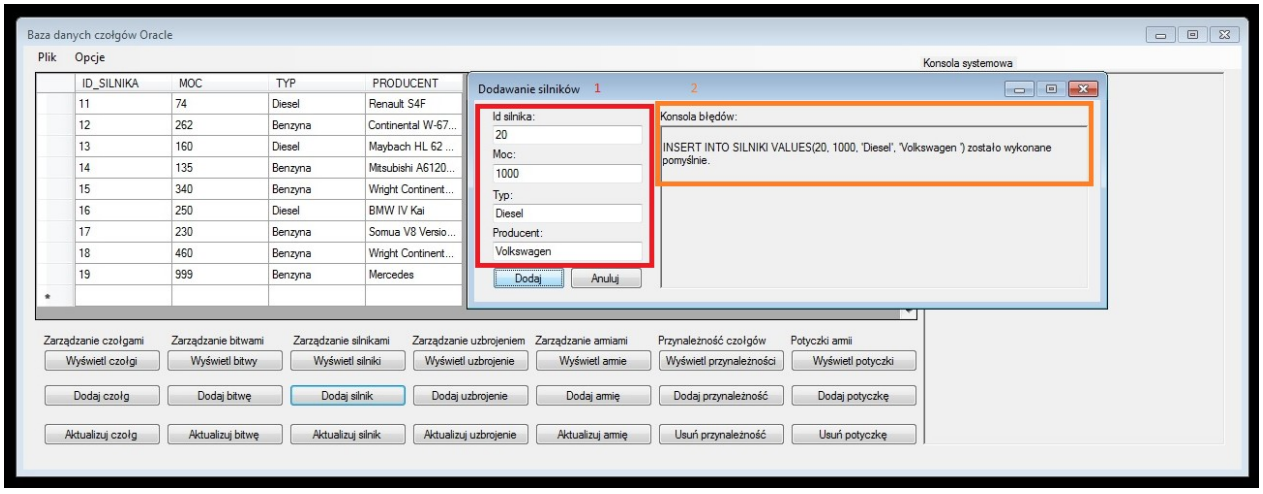
ID_CZOLGU	ID_SILNIKA	ID_UZBROJENIA	TYP	LICZBA_ZALOGAN	NACJA	PANCERZ_PRZOI	PANCERZ_TYL	PANCER
1	1	1	Artyleria	20	Japonia	16	16	16
2	2	2	Lekki	2	Niemcy	14	14	14
3	3	1	Lekki	3	Czechy	14	6	8
4	4	3	Lekki	3	USA	10	6	6
5	5	4	Lekki	5	Szwecja	14	14	14
6	6	1	Lekki	2	Japonia	30	22	30
7	6	1	Lekki	2	Chiny	16	16	16
8	7	4	Lekki	2	Francja	16	16	16
9	8	5	Sredni	5	Wielka Brytania	6	6	6
10	9	2	Lekki	3	USSR	15	10	13

Jak widać, na przykładzie tego screena, oraz screena wyjaśniającego wyświetlanie tabel, rekord dla ID_CZOLGU=1 został zmieniony w przypadku typu (nowy typ to Artyleria) oraz liczby załogantów (20). Dodatkowo, konsola informuje użytkownika o prawidłowym wykonaniu zapytania.

W przypadku sekcji 3a (zaznaczonej na poprzedniej stronie na czarno) w przypadku tabel łączących (przynależność czołgów oraz potyczki armii), wybranie rekordu oraz użycia przycisku USUŃ powoduje skasowanie danego rekordu. Kasowanie odbywa się poprzez pobieranie obydwu ID w tabelach łączących, pozwala to uniknąć usuwania danych, które nie miały być usuwane.

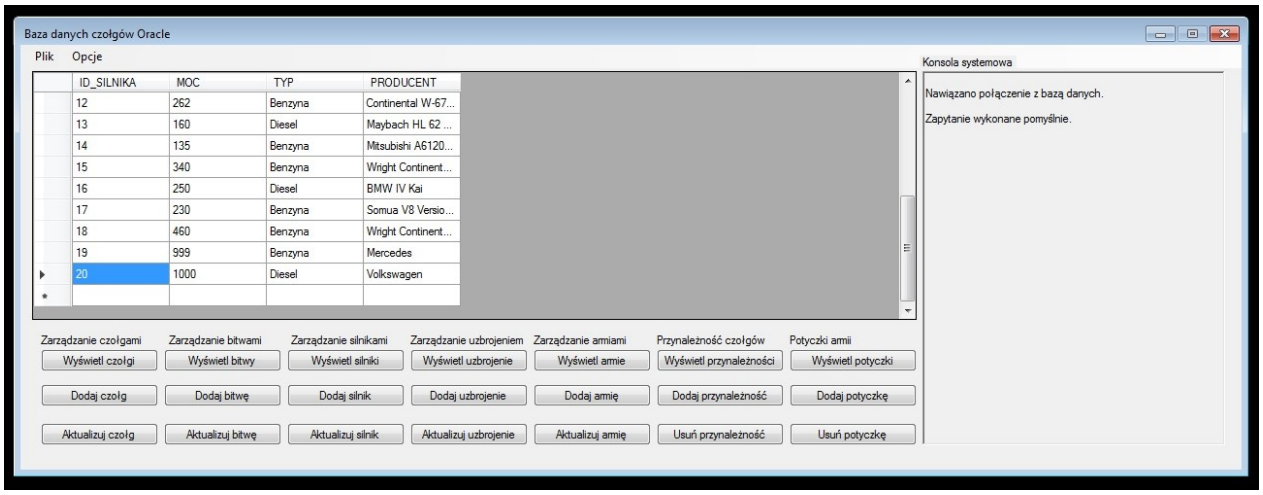
Sekcja 2: Dodawanie.

Dodawanie odbywa się w nieco inny sposób niż wyświetlanie i aktualizacja tabeli. Użycie przycisku „Dodaj” niezależnie od wyświetlanej tabeli wywołuje nowe okno, które służy do wprowadzania i dodawania danych do bazy, przykładem tutaj będzie dodawanie silnika:



ID_SILNIKA	MOC	TYP	PRODUCENT
11	74	Diesel	Renault S4F
12	262	Benzyna	Continental W-67...
13	160	Diesel	Maybach HL 62 ...
14	135	Benzyna	Mitsubishi A6120...
15	340	Benzyna	Wright Continent...
16	250	Diesel	BMW IV Kai
17	230	Benzyna	Somua V8 Versio...
18	460	Benzyna	Wright Continent...
19	999	Benzyna	Mercedes

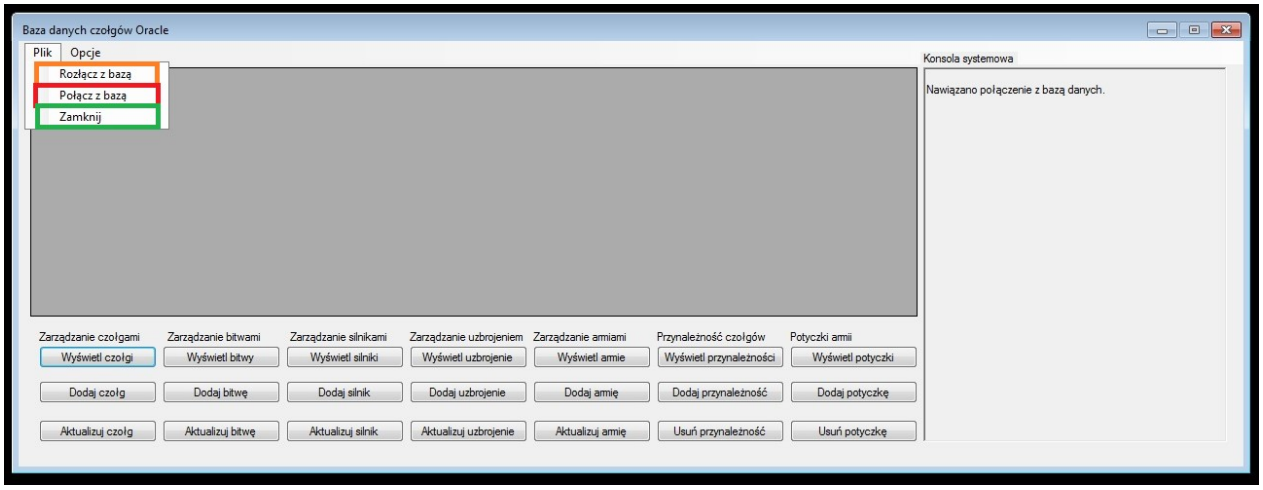
Użycie przycisku „Dodaj silnik” spowodowało wywołanie nowego okna, w którym trzeba uzupełnić dane. Po wprowadzeniu danych w odpowiednie pola tekstowe program konstruuje zapytanie, wykonuje je, a następnie wyświetla je wraz z rezultatem w konsoli. Po dodaniu rekordu należy na nowo użyć przycisku „Wyświetl silniki”, aby upewnić się, że rekord został dodany prawidłowo:



Jak widać, ostatnim rekordem jest rekord o ID=20 (taki, jaki został dodany w instrukcji) wraz z pozostałymi danymi. Dane wyświetlane są bezpośrednio z bazy danych, nie ma więc powodów aby martwić się, że rekord został dodany tylko do tabeli w aplikacji, nie zaś do tabeli w bazie.

Dodatkowe funkcjonalności zapewniają sekcja 6 oraz 7.

Sekcja 7 pozwoli wywołać menu, które pokazuje podstawowe informacje o programie, natomiast sekcja 6 jest nieco bardziej rozbudowana:



Po użyciu przycisku „Rozłącz z bazą” znajdującego się w sekcji pomarańczowej nastąpi przerwanie połączenia oraz dezaktywacja wszelkich przycisków związanych z obsługą bazy danych. Ma to na celu prewencję powstawania niechcianych błędów.

Przycisk „Połącz z bazą” znajdujący się w czerwonej sekcji pozwala na wznowienie połączenia, dodatkowo po pomyślnym połączeniu aktywuje on przyciski odpowiadające za obsługę bazy, przywracając 100% funkcjonalności programu.

Przycisk „Zamknij” w zielonej sekcji powoduje natychmiastowe rozłączenie z bazą oraz zamknięcie programu.

6. Bibliografia

W trakcie tworzenia projektu byłem zmuszony do poznania nowych technologii, takich jak łączenie się z bazą danych *Oracle* poprzez aplikację stworzoną w języku *C#*. Dodatkowo, musiałem zapoznać się z językiem *PL/SQL*, w celu stworzenia różnych funkcji i procedur. Wykorzystany też został program *Data Modeler* w celu utworzenia schematu *ERD* bazy danych.

Dokumentacja klas System.Data.OracleClient:
https://msdn.microsoft.com/pl-pl/library/system.data.oracleclient.oracleconnection(v=vs.110).aspx

Ćwiczenia dot. Baz danych oraz języka PL/SQL:
http://wazniak.mimuw.edu.pl/index.php?title=Bazy_danych

Oracle SQL Developer Data Modeler:
http://www.oracle.com/technetwork/developer-tools/datamodeler/overview/index.html

Dodatkowo, program posiada następujące wymagania sprzętowe:
- system Windows w wersji 64 bitowej, wraz z zainstalowanym pakietem Oracle,
- 64 MB RAM (w testach debugowania program nie wykazał większego zużycia, niż 34 mb),
- 3 MB miejsca na dysku,
- 64bitowy procesor,
- platforma .NET Framework w wersji 4.6 lub wyższej.