Dokumentacja projektu SzkolaJK

1. Analiza projektu
   1 Opis stanu zastanego
   2 Wymagane funkcje aplikacji
   3 Specyfikacja projektu
      a) Informacje przechowywane w bazie danych
      b) Zdarzenia powodujące aktualizację danych
   4 Specyfikacja implementacji
      a) Tabele wraz z określonymi typami
      b) Klucze tabel i relacje

2. Definicja bazy danych
   1 Normalizacja
   2 Ograniczenie dostępu do bazy
   3 Procedury składowane
   4 Triggery
   5 Widoki

3. Implementacja
   1 Spis formularzy
   2 Spis raportów
   3 Dokładny opis funkcjonalności formularzy i raportów
   4 Dane testowe

4. Instrukcja użytkownika
   1 Instalacja i zarządzanie serwerem
      a) Tworzenie bazy danych
      b) Dostęp dobrzy danych
      c) Zmiana i dodawanie użytkowników
      d) Zarządzanie istniejącą bazą danych
   2 Instalacja i zarządzanie aplikacją
      a) Instalacja aplikacji
      b) Ustanowienie połączenia z MS SQL Server
      c) Dokładny opis funkcjonalności poszczególnych formularzy z punkty widzenia użytkownika końcowego

5. Załączniki
   1 Spis tabel wraz z atrybutami, kluczami, relacjami
   2 Diagram ERD
   3 Skrypt SQL
   4 Schemat dostępu do formularzy w aplikacji

Jedynak Mirosław
Kamiński Maciej

SZKOŁA

Cel projektu.

Celem projektu było stworzenie bazy danych dla liceum, która spełniałaby wszystkie niezbędne funkcje i zastąpiła nieefektywne i zajmujące dużo miejsca dotychczasowe archiwum.

Opis:

Zanim przystąpiliśmy do projektowania bazy przeprowadziliśmy rozmowy z nauczycielami, pracownikami sekretariatu i dyrektorami, aby dowiedziec się, jakie stawiają wymagania takiej bazie danych. Dzięki tym rozmowom stworzyliśmy listę funkcji i tabeli, które musi zawierać nasza baza, aby mogła być w pełni funkcjonalna.

Oprócz listy uczniów w bazie powinna znajdować się lista nauczycieli, klas, przedmiotów, ocen uczniów i stypendiów.

W bazie będą oceny półroczne i końcoworoczne.

Baza danych zostałaby wprowadzona do użytku na początku roku szkolnego. Przed rozpoczęciem nauki do bazy należy wprowadzić wszystkie potrzebne dane. Należałoby to do obowiązków pracowników sekretariatu. Po zakończeniu edukacji przez danego ucznia do jego kartoteki byłaby wpisywana informacja o dacie odejścia. Zgodnie z obowiązującym prawem kartoteka ucznia byłaby usuwana po okresie 5 lat od zakończenia edukacji w danej szkole.

Ograniczyliśmy się w naszym projekcie do przypadku liceum ogólnokształcącego. Każdego roku tworzy się nowe klasy o konkretnych profilach. Aby identyfikować poszczególne klasy tego samego rocznika oznacza się je kolejnymi literami alfabetu. Każda klasa musi posiadać wychowawcę spośród nauczycieli. Następnie klasom, według ich profilu, zostają przyporządkowane

przedmioty, ilośc godzin nauczania danego przedmiotu i jego waga. Gdy istnieją już klasy można dodawać do nich uczniów. Szkoła musi posiadać w swej bazie danych pełne dane osobowe każdego ucznia.

Szkoła nie może, oczywiście istnieć bez nauczycieli. Każdy z nich posiada tytuł naukowy i może nauczać więcej niż jednego przedmiotu. Nauczyciel może być wychowawcą jednej klasy.

Szkoła musi przechowywać w swojej bazie danych oceny półroczne i końcoworoczne każdego ucznia. Musi znać datę wystawienia oceny, nauczyciela wystawiającego i przedmiot. Na podstawie tych ocen wyliczana jest średnia dla poszczególnych uczniów. Podczas wyliczania uwzględnia się wagę danego przedmiotu, która zależy od klasy i profilu klasy do jakiej uczęszcza uczeń. Jeśli średnia jest odpowiednio wysoka, to uczniowi może zostać przyznane stypendium. Będzie je otrzymywał w stałej kwocie przez określony czas.

Szkoła nie może również zapomnieć o wypłatach dla nauczycieli. Na podstawie tego ile godzin w tygodniu nauczają wyliczana jest pensja, przyznawana co miesiąc.

Funkcje bazy danych:

- wprowadzanie, usuwanie i aktulizacja informacji o uczniach;
- wprowadzani i aktualizacja ocen uzyskanych przez uczniów;
- dodawanie,usuwanie i aktualizacja danych o nauczycielach;
- modyfikacja przypisania nauczyciela do przedmiotu i klasy;
- przyznawania i odbieranie stypendium uczniom;
- utworzenie i rozwiązanie klasy;
- dodanie i usunięcie przedmiotu;

- WYSZUKIWANIE z bazie uczniów wg. różnych kryteriów:
  1 przyznane stypendia;
  2 klasy;
  3 wyniki w nauce (średnia);
  4 dane osobowe ucznia;

- tworzenie RAPORTÓW:
  1 zestawienie uczniów, którzy nie zakończyli roku szkolnego;
  2 uczniowie pretendujący do otrzymania stypendium;
  3 przyznane stypendia;

Inforamcje przechowywane w bazie danych

- informacje dotyczące UCZNIÓW:
  - PESEL
  - numer legitymacji
  - imię
  - nazwisko
  - adres zamieszkania
  - data urodzenia
  - data rozpoczęcia nauki
  - data zakończenia nauki
  - klasa
  - stypendium

- informacje dotyczące PRZEDMIOTU:
  - nauczyciel uczący
  - pełna nazwa
  - ilość
  - godzin w poszczególnych klasach
  - waga przedmiotu

- informacje dotyczące KLASY:
  - rok rozoczecia
  - podrok {a,b...z}
  - wychowawca
  - rodzaj klasy {profil}
  - rok

- informacje dotyczące OCEN:
  - uczeń który ją otrzmał
  - przedmiot
  - data wystawienia
  - ocena
  - nauczyciel, który wystawił ocenę
  - średnia ze wszystkich ocen danego ucznia

- informacje dotyczące NAUCZYCIELI:
  - PESEL
  - imię
  - nazwisko
  - tytuł
  - klasa, której jest wychowawcą
  - zarobki

    - przedmiot których uczy

- informacje dotyczące STYPENDIUM:
    - data przyznania
    - data zakończenia
    - kwota
    - uczeń, który je otrzymał

Baza składa się z tabel:

- uczniowie – tabelę tę wypełniają wszyscy uczniowie uczęszczający do szkoły.Identyfikowani są poprzez numer PESEL. Oprócz adresu, daty urodzenia, daty przyjęcia i odejścia ze szkoły – przechowywany jest klucz klasy,do której uczęszcza uczeń.
- rodzaje_klas – tabela słownikowa, która zwiera wszystkie rodzaje klas,jakie mogą występować w szkole.
- nauczyciele – w tabeli tej znajdują się imię,nazwisko i tytuł naukowy nauczycieli uczących w szkole.
- klasy – tu znajdują się informacje o poszczególnych klasach. Przechowujemy rok rozpoczęcia nauki danej klasy, jej podrok(czyli oznaczenie od 'a' do 'z'), indeks profilu klasy oraz id wychowawcy.
- przedmioty – tablica słownikowa wszystkich przedmiotów nauczanych w szkole.
- oceny – jedna z najważniejszych tabel zawierająca oceny półroczna i końcoworoczne uczniów. Do każdej oceny dodajemy informacje o uczniu, który ją otrzymał, nauczycielu wystawiającym, datę wystawienia oraz przedmiot.
- przedmiot_klasa_nauczyciel – tabela łącząca przedmioty nauczane w poszczególnych klasach z konkretnymi nauczycielami. Tabela zawiera równiez ilość godzin danego przedmiotu w tygodniowo.
- klasa_przedmiot – ta tabela zawiera informację o wadze danych przedmiotów w poszczególnych profilach klas.

  stypendia – w tej tabeli znajdują się stypendia przyznawane uczniom za wyniki w nauce. Od daty przyznania do daty zakończenia musi minąc minimum 30 dni.Tabela zawiera również wartość stypendium.

Zdarzenia powodujące zmiany w bazie:

- wpisanie, usunięcie lub aktualizacja danych ucznia
- wpisanie, usunięcie,zmiana posady lub aktualizacja danych nauczyciela
- zmiana wychowawcy klasy
- zmiana w przydziale przedmiotów do klas
- przyznanie lub zawieszenie stypendium
- uzyskanie lub zmiana oceny
- dodanie lub usunięcie przedmiotu
- utworzenie lub rozwiązanie klasy

Normalizacja

Baza danych spełnia następujące *postaci normalne*:

- Posiada tylko pola z wartościami atomowymi – nie jest w naszej bazie podział pola na części składowe – spełnia warunki **1 postaci normalnej**
- Kolumny klucza nie wchodzące w skład klucza zależne są od całego klucza (konieczna była dekompozycja na dwie tabele *nauczyciel_klasa_przedmiot* i *klasa_przedmiot_waga*, ponieważ waga oceny nie była zależna od całego klucza(nauczyciela)) – spełnia warunki **2 postaci normalnej**
- Wszystkie kolumny nie należące do klucza są funkcjonalnie zależne od klucza kandydującego – spełnia warunki **3 postaci normalnej**
- BCPN – postać normalna Boyce-Codd'a nie dotyczy bazy danej ponieważ żadna tabela nie ma dwóch kluczy kandydujących składających się z więcej niż jednej kolumny

Wniosek: Baza danych jest w **3 postaci normalnej**, więc może zostać zaimplementowana w *systemie zarządzania bazą danych* bez obawy o utratę integralności bazy danych.

Dostęp do naszej bazy danych w szkole będzie miała sekretarka, która będzie miała pełen dostęp do bazy danych. Do jej czynności będzie należało:

- wprowadzanie do bazy danych uczniów,przedmiotów,nauczycieli i klas;
- przypisywanie uczniów do klas;
- ustalanie ilości godzin i nauczyciela każdego przedmiotu dla klas;
- tworzenie raportów;

- usuwanie klas i uczniów,którzy zakończyli już edukację;
- usuwanie nieaktualnych ocen;

Również dyrektor szkoły będzie miał pełne prawa dostępu do bazy danych. Ograniczone możliwości operacji na bazie danych otrzymają nauczyciele, którzy będą dodawali, aktualizowali lub usuwali oceny uczniów.

Użytkownik będzie korzystał z bazy danych poprzez prosty interface.

Założenia przy tworzeniu bazy danych

Tworzona przez nas baza danych ma 20 Mb, ale może maksymalnie osiągnąć 100Mb.Plik tymczasowy ma 5 Mb i może zajmować maksymalnie 15Mb. Ograniczenia te nie są sztywne i można je zmienić

Utworzone procedury składowane

- pupil_in_school – sprawdza,czy uczeń uczęszcza nadal do szkoły;
- year_of_class – podaje rok,na którym jest klasa oraz 0 jesli klasa juz zakonczyla naukę;
- count_subjects_of_class – liczy wszystkie przedmioty których uczy się klasa;
- count_marks_of_student – liczy ilośc uzyskanych ocen przez ucznia w danym przedziale czasu;
- count_last_marks_of_student - ilosc ocen dla danego ucznia w caigu ostatnich 3 miesiecy;
- is_teacher_of_class_subject – informuje czy nauczcielem uczy przedmiotu w danej klasie;
- count_marks_for_class - zwraca ilosc ocen jaki trzeba uzyskac w danej klasie;
- weight_of_mark - zwraca wage danej oceny dla danego ucznia;
- average_of_marks – zwraca średnią ocen z ostatnich 7 miesięcy;

Triggery użyte w bazie danych Szkola_JK

- dbają, aby daty wpisywane w różnych polach nie przekraczały dozwolonych granic;
- aby nie można było usunąć klasy zanim nie usunie się wszystkich jej uczniów;
- kontrola poprawności formatu wpisywanych danych
- aby nauczyciel wystawial oceny tylko z przedmiotu, którego uczy;
- jeżeli po dodaniu iceny, średnia ucznia będzie wyższa niż pewien poziom,

automatycznie zostanie dodane stypendium;

- sprawdzają, aby nie można było dodać stypendium, jeśli uczeń nie ma jeszcze wszystkich potrzebnych ocen;

SZKOŁA

Tabele z atrybutami i ich typy:

UCZNIOWIE:

Klucz	*Atrybut*	*Typ atrybutu*
PK	PESEL	INT
	Imię	CHAR
	Nazwisko	CHAR
	Adres zamieszkania	VARCHAR
K	Numer legitymacji	INT
	Data urodzenia (< aktualna)	DATETIME
	Data przyjęcia (< aktualna)	DATETIME
	Data odejścia (< aktualna) (NULL)	DATETIME
	Klasa	INT
	Stypendium	BOOLEAN

KLASY:

Klucz	*Atrybut*	*Typ atrubutu*
PK	ID_Klasy	INT
K	Rok ropoczęcia	DATETIME
K	Podrok (a...z)	CHAR
	Wychowawca	CHAR
	Rodzaj klasy	INT
	Rok (1,2,3)	INT

KLASA_PRZEDMIOT:

Klucz	*Atrybut*	*Typ atrybutu*
PK	Przedmiot	INT
PK	Rodzaj klasy	INT
	Waga	INT
PK	Rok	INT

PRZEDMIOTY:

Klucz	*Atrybut*	*Typ atrybutu*
PK	ID_Przedmiotu	INT
K	Nazwa	CHAR

PRZEDMIOT_KLASA_NAUCZYCIEL:

Klucz	*Atrybut*	*Typ atrybutu*
PK	Nauczyciel	INT
PK	Przedmiot	INT
PK	Klasa	INT
	Godzin	INT

RODZAJE_KLAS:

Klucz	*Atrybut*	*Typ atrybutu*
	ID_Rodzaju	INT
K	Nazwa rodzaju	CHAR

NAUCZYCIELE:

Klucz	*Atrybut*	*Typ atrybutu*
PK	PESEL	INT
	Imię	CHAR
	Nazwisko	CHAR
	Tytuł	CHAR

OCENY:

Klucz	*Atrybut*	*Typ atrybutu*
PK	ID_Oceny	INT
	Ocena (1..6)	INT
	Uczeń	INT
	Przedmiot	INT
	Data	DATETIME
	Nauczyciel	INT

STYPENDIA:

Klucz	*Atrybut*	*Typ atrybutu*
PK	ID_Stypendium	INT
	ID_Ucznia	INT
	Średnia	INT
	Data początku	DATETIME
	Data końca	DATETIME

WARTOŚĆ STYPENDIUM:

Klucz	*Atrybut*	*Typ atrybutu*
	Dolna granica	INT
	Górna granica	INT
	Wartość	INT

WYPŁATY:

Klucz	*Atrybut*	*Typ atrybutu*
	Data	DATETIME
	Nauczyciel	CHAR
	Zarobki	INT

Skrypt SQL (Szkola_JK)

```
--usuwanie bazy danych

USE master
 IF EXISTS(SELECT name FROM sysdatabases WHERE name='Szkola_JK')
        BEGIN
           DROP DATABASE Szkola_JK

        END

-- tworzenie bazy danych


USE master
GO
CREATE DATABASE Szkola_JK ON PRIMARY
        ( NAME = SzkolaData,
        FILENAME =
        'C:\Szkola_JK\SzkolaData.mdf',
        SIZE = 20MB,
        MAXSIZE = 100 MB,
        FILEGROWTH = 10 MB )
        LOG ON
        (NAME = SzkolaLog,
        FILENAME =
        'C:\Szkola_JK\SzkolaLog.ldf',
        SIZE = 5MB,
        MAXSIZE = 15MB,
        FILEGROWTH = 1 MB)
        COLLATE Polish_CI_AI
GO


--tworznie tabel dla bazy danych--
----------------------------------
CREATE TABLE rodzaje_klas
        ( ID_rodzaje_klas       INT                     IDENTITY(1,1)
           CONSTRAINT pk_rodzaje_klas PRIMARY KEY CLUSTERED ,
        rodzaj_klasy            VARCHAR(20) UNIQUE NOT NULL)



CREATE TABLE nauczyciele
        (
        ID_nauczyciele                  INT
           IDENTITY(1,1)
                   CONSTRAINT pk_nauczyciele PRIMARY KEY
        CLUSTERED,
        imie                                            VARCHAR(20) NOT
        NULL,
        nazwisko                        VARCHAR(30) NOT NULL,
        tytul                                           VARCHAR(15) NOT
        NULL
           CONSTRAINT df_tytul
                   DEFAULT ('mgr')     )
```

```
CREATE TABLE klasy
        (
        ID_klasy                        INT
           IDENTITY(1,1)
           CONSTRAINT pk_klasy PRIMARY KEY CLUSTERED,
        rok_rozpoczecia                 DATETIME
           NOT NULL
           CONSTRAINT ck_rok_rozpoczcia
                   CHECK (rok_rozpoczecia<= YEAR(GETDATE())),
--      rok_rozpoczecia                                         INT
           NOT NULL
--         CONSTRAINT ck_rok
--                 CHECK ((rok >=1) AND (rok <=3))
--         CONSTRAINT df_rok
--                 DEFAULT (1),
        pod_rok                                         CHAR(1)  NOT NULL
           CONSTRAINT ck_pod_rok
                   CHECK (pod_rok LIKE '[a-z]'),
        wychowawca                      INT                     NOT NULL
           CONSTRAINT rf_wychowawca_klasy REFERENCES
        nauczyciele(ID_nauczyciele),
        rodzaj_klasy            INT                     NOT NULL
           CONSTRAINT rf_rodzaj_klasy_klasy REFERENCES
        rodzaje_klas(ID_rodzaje_klas)
        CONSTRAINT unq_rok_pod_rok UNIQUE
        (rok_rozpoczecia,pod_rok))



CREATE TABLE przedmioty
        (
        ID_przedmioty                   INT
           IDENTITY(1,1)
           CONSTRAINT pk_przedmioty PRIMARY KEY CLUSTERED,
        przedmiot                       VARCHAR(25) NOT NULL
           CONSTRAINT unq_przedmiot UNIQUE (przedmiot)
        )

CREATE TABLE uczniowie
        (
        PESEL                                           CHAR(11)
           CONSTRAINT pk_uczniowie    PRIMARY KEY CLUSTERED
           CONSTRAINT ck_PESEL
                   CHECK (PESEL LIKE '[0-9][0-9][0-9][0-9][0-9][0-
        9][0-9][0-9][0-9][0-9][0-9]'),

        imie                                            VARCHAR(15) NOT
        NULL,
        nazwisko                        VARCHAR(25) NOT NULL,
        nr_legitymacji                  VARCHAR(15)       UNIQUE
        NOT NULL,
        data_ur                                         DATETIME
           CONSTRAINT ck_data_ur
                   CHECK (YEAR(data_ur) BETWEEN
        (YEAR(GETDATE())-100) AND (YEAR(GETDATE())-10)) ,
        data_przyjecia                  DATETIME            NOT NULL
           CONSTRAINT ck_data_przyjecia
                   CHECK (data_przyjecia<=GETDATE())
           DEFAULT (GETDATE()),
        data_odejscia           DATETIME            NULL
```

```
	CONSTRAINT ck_data_odejscia
		CHECK (/*(data_odejscia >= data_przyjecia)
			AND*/
	(data_odejscia<=GETDATE() +7))
	DEFAULT NULL,
klasa				INT
	NULL
	CONSTRAINT rf_klasy_uczniowie
		REFERENCES klasy(ID_klasy)
		ON DELETE NO ACTION
		ON UPDATE NO ACTION
)

CREATE TABLE oceny
(
ID_oceny			INT
	IDENTITY(1,1)
	CONSTRAINT pk_oceny	PRIMARY KEY CLUSTERED,
uczen				CHAR(11)
	NOT NULL
	CONSTRAINT rf_uczen_oceny
		REFERENCES uczniowie(PESEL),
ocena				SMALLINT
	NOT NULL
	CONSTRAINT ck_ocena
		CHECK (ocena BETWEEN 1 AND 6),
data				DATETIME
	NOT NULL
	CONSTRAINT ck_data
		CHECK (data<GETDATE()+14),
nauczyciel			INT		NOT NULL
	CONSTRAINT rf_nauczyciel_oceny
		REFERENCES	nauczyciele(ID_nauczyciele),
przedmiot			INT
	NOT NULL
	CONSTRAINT rf_przedmiot_oceny
		REFERENCES przedmioty(ID_przedmioty)
)

CREATE TABLE przedmiot_klasa_nauczyciel
(
przedmiot			INT		NOT NULL
	CONSTRAINT rf_przedmiot_pkn
		REFERENCES przedmioty(ID_przedmioty),
klasa				INT
	NOT NULL
	CONSTRAINT rf_klasa_pkn
		REFERENCES klasy(ID_klasy),
nauczyciel			INT		NOT NULL
	CONSTRAINT rf_nauczyciel_pkn
		REFERENCES nauczyciele(ID_nauczyciele),
godzin				INT
	NOT NULL
	CONSTRAINT ck_godzin
		CHECK (godzin >=0)
CONSTRAINT pk_przedmiot_klasa_nauczyciel
	PRIMARY KEY(przedmiot,klasa,nauczyciel)
)

CREATE TABLE klasa_przedmiot
(
przedmiot			INT		NOT NULL
	CONSTRAINT rf_przedmiot_kp
		REFERENCES przedmioty(ID_przedmioty),
rodzaj_klasy		INT		NOT NULL
	CONSTRAINT rf_rodzaj_klasy_kp
		REFERENCES rodzaje_klas(ID_rodzaje_klas),
rok					INT
	NOT NULL
	CONSTRAINT ck_rok
		CHECK ((rok >=1) AND (rok <=3)),
waga				INT		NOT NULL
	CONSTRAINT ck_waga
		CHECK ((waga >= 0) AND (waga<=100))
	CONSTRAINT df_waga
		DEFAULT (10)
	CONSTRAINT		pk_klasa_przedmiot
	  PRIMARY KEY (przedmiot,rodzaj_klasy,rok)
	)

CREATE TABLE stypendia
	(
	id_stypendia		INT		NOT NULL
	IDENTITY(1,1)
	  CONSTRAINT pk_stypendia PRIMARY KEY,
	uczen				CHAR(11)
	  NOT NULL
	  CONSTRAINT rf_uczniowie_stypendia
			REFERENCES uczniowie(PESEL)
			ON DELETE CASCADE
			ON UPDATE CASCADE,
	data_rozpoczecia	DATETIME	NOT NULL
	  CONSTRAINT ck_data_rozpoczecia
			CHECK (data_rozpoczecia<=GETDATE())
	  CONSTRAINT df_data_rozpoczecia
			DEFAULT(GETDATE()),
	data_zakonczenia	DATETIME	NOT NULL,
	  --CONSTRAINT ck_data_zakonczenia
	  --		CHECK (data_zakonczenia>data_rozpoczecia+30) ,
				--przynajmniej 30 dni trwa swiadczenie
	wartosc				SMALLMONEY
	  NOT NULL,
	)


---procedury

--sprawdza czy uczen jest uczniem ktory jescze nie zakonczyl
use Szkola_JK
GO
-------------------------------------------------------------------------------------
--pupil_in_school
--args: uczen - PESEL
--ret: {0,1} - ilosc uczniow ktorzy sa jeszcze w szkole
-------------------------------------------------------
IF object_id('pupil_in_school') IS NOT NULL
BEGIN
	DROP PROCEDURE pupil_in_school
END
GO


CREATE PROCEDURE pupil_in_school @uczen CHAR(11)
AS
BEGIN
	RETURN (SELECT count(*)  --u.data_przyjecia, u.data_odejscia,
	  FROM uczniowie u
	  WHERE ((u.PESEL = @uczen)
				AND u.data_odejscia IS NULL ))
END
GO
-------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------
--year_of_class
--args: id_klasy
--ret: integer - rok albo 0 jak klasa juz skonczyla
-------------------------------------------------------
IF object_id('year_of_class') IS NOT NULL
BEGIN
	DROP PROCEDURE year_of_class
END
GO


CREATE PROCEDURE year_of_class @class int

AS
BEGIN
	DECLARE @rok_rozp INT
	SET		@rok_rozp = (
	  SELECT rok_rozpoczecia
	  FROM klasy
```

```
        WHERE ID_klasy=@class
    )

    DECLARE @rok INT
    SET @rok= YEAR(DATEADD(mm,-7,GETDATE()))-@rok_rozp+1
    IF @rok>3
        RETURN 0
    ELSE RETURN @rok

END
GO
-------------------------------------------------------------------------------------------


-------------------------------------------------------------------------------------------
--count_subjects_of_class
--args: class - klasa
--ret: integer - ilosc przedmiotow dla danej klasy
-------------------------------------------------------
IF object_id('count_subjects_of_class') IS NOT NULL
BEGIN
    DROP PROCEDURE count_subjects_of_class
END
GO


CREATE PROCEDURE count_subjects_of_class @class INT
AS
BEGIN
    DECLARE @rok INT
    EXEC @rok=year_of_class @class
    DECLARE @rodzaj_klasy INT
    SET @rodzaj_klasy = (
        SELECT rodzaj_klasy
        FROM klasy
        WHERE id_klasy=@class
    )


    RETURN (
        SELECT COUNT(*)
        FROM klasa_przedmiot
        WHERE (@rodzaj_klasy =rodzaj_klasy) AND (rok =@rok)
    )


END
-------------------------------------------------------------------------------------------
GO
-------------------------------------------------------------------------------------------
--count_marks_of_student
--args: student - klasa
--          begin       - poczatek - od tej daty liczone beda oceny
--          end                     - koniec - do tej daty beda liczone oceny
--ret: integer - ilosc ocen dla danego ucznia w danym przedziale
-------------------------------------------------------
IF object_id('count_marks_of_student') IS NOT NULL
BEGIN
    DROP PROCEDURE count_marks_of_student
END
GO


CREATE PROCEDURE count_marks_of_student @student CHAR(11),
                    @begin DATETIME, @end DATETIME
AS
BEGIN
    RETURN(
        SELECT COUNT(*)
        FROM oceny
        WHERE (uczen=@student) AND (data BETWEEN @begin AND
    @end)
    )
END
-------------------------------------------------------------------------------------------
GO
--count_last_marks_of_student
--args: student - klasa
--ret: integer - ilosc ocen dla danego ucznia w caigu ostatnich 3 miesiecy
-------------------------------------------------------
IF object_id('count_last_marks_of_student') IS NOT NULL
BEGIN
    DROP PROCEDURE count_last_marks_of_student
END
GO


CREATE PROCEDURE count_last_marks_of_student @student CHAR(11)
AS
BEGIN
    DECLARE @begin DATETIME
    DECLARE @end DATETIME
    SET @begin=DATEADD(mm,-7,GETDATE()) --ostatnie 3 miesiace
    SET @end=GETDATE()
    RETURN(
        SELECT COUNT(*)
        FROM oceny
        WHERE (uczen=@student) AND (data BETWEEN @begin AND
    @end)
    )
END
-------------------------------------------------------------------------------------------
GO

GO
--is_teacher_of_class_subject
--args: @teacher - nauczyciel
--          @class      - klasa
--          @subject - przedmiot
--ret: jesli jest nauczcielem danego przedmiotu w danej klasie to zwraca 1
-------------------------------------------------------
IF object_id('is_teacher_of_class_subject') IS NOT NULL
BEGIN
    DROP PROCEDURE is_teacher_of_class_subject
END
GO


CREATE PROCEDURE is_teacher_of_class_subject @teacher INT, @class
    INT, @subject INT
AS
BEGIN
    RETURN (
        SELECT COUNT(*)
        FROM przedmiot_klasa_nauczyciel
        WHERE (@teacher=nauczyciel) AND (@class=klasa) AND
    (@subject=przedmiot)
    )
END
-------------------------------------------------------------------------------------------
GO
GO
--count_marks_for_class
--args: @class           - klasa
--ret: zwraca ilosc ocen jaki trzeba uzyskac w danej klasie
-------------------------------------------------------
IF object_id('count_marks_for_class') IS NOT NULL
BEGIN
    DROP PROCEDURE count_marks_for_class
END
GO


CREATE PROCEDURE count_marks_for_class @class INT
AS
BEGIN
    DECLARE @rok INT
    DECLARE @rodzaj_klasy INT

    SELECT @rodzaj_klasy=rodzaj_klasy
    FROM klasy
    WHERE (ID_klasy=@class)
```

```
	EXEC @rok=year_of_class @class

	RETURN (
	   SELECT COUNT(*)
	   FROM klasa_przedmiot
	   WHERE (@rok=rok) AND ( @rodzaj_klasy=rodzaj_klasy)
	)
END
-----------------------------------------------------------------------------------------
GO
-----------------------------------------------------------------------------------------
--weight_of_mark
--args: @student - uczen
--	@id_mark     - ocena ucznia (ID oceny)
--ret: zwraca wage danej oceny dla danego studenta
-----------------------------------------------------
IF object_id('weight_of_mark') IS NOT NULL
BEGIN
	DROP PROCEDURE weight_of_mark
END
GO


CREATE PROCEDURE weight_of_mark @student char(11), @id_mark int
AS
BEGIN
	DECLARE @class INT --klasa do ktorej chodzi uczen
	DECLARE @rodzaj_klasy INT --rodzaj klasy do ktorej chodzi
	DECLARE @rok INT		--ktory rok danej klasy

	SELECT  @class=klasa
	FROM uczniowie u
	WHERE (@student=u.PESEL)

	SELECT @rodzaj_klasy=rodzaj_klasy
	FROM klasy
	WHERE (ID_klasy=@class)

	EXEC @rok=year_of_class @class

	DECLARE @subject INT		--przemiot z ktorego jest dana
	ocena

	SELECT @subject= przedmiot
	FROM oceny o
	WHERE (o.ID_oceny=@id_mark)

	DECLARE @weight int

	SELECT @weight=waga
	FROM klasa_przedmiot kp
	WHERE (kp.przedmiot=@subject) AND
	(kp.rodzaj_klasy=@rodzaj_klasy)
			AND (kp.rok=@rok)
	RETURN @weight
END
-----------------------------------------------------------------------------------------
GO
-----------------------------------------------------------------------------------------
--average_of_marks
--args: @student - uczen
--	@begin_date - pocztek przedialu licznia ocen
--	@ end_date  - koniec przedialu liczenia ocen
--ret: zwraca srednia ocen (ostatnich 7 miesiecy) * 100!!!
-----------------------------------------------------
IF object_id('average_of_marks') IS NOT NULL
BEGIN
	DROP PROCEDURE average_of_marks
END
GO


CREATE PROCEDURE average_of_marks @student CHAR(11),
	@begin_date DATETIME= '1/1/1900',
				@end_date DATETIME= '1/1/1900'
AS
BEGIN
	--ustwienie domyslnych parametrow
	IF (( @begin_date=CONVERT(DATETIME,'1/1/1900'))
	   AND (@end_date=@begin_date))
	BEGIN
	   SET @begin_date=DATEADD(mm,-7,GETDATE())
	   SET @end_date=GETDATE()
	END
	DECLARE @class INT --klasa do ktorej chodzi uczen
	DECLARE @rodzaj_klasy INT --rodzaj klasy do ktorej chodzi
	DECLARE @rok INT		--ktory rok danej klasy

	SELECT  @class=klasa
	FROM uczniowie u
	WHERE (@student=u.PESEL)

	SELECT @rodzaj_klasy=rodzaj_klasy
	FROM klasy
	WHERE (ID_klasy=@class)

	EXEC @rok=year_of_class @class

	DECLARE @avg FLOAT


--	SELECT o.ocena, p.przedmiot,kp.waga, o.ocena*kp.waga

	SELECT
	@avg=SUM(CONVERT(FLOAT,o.ocena*kp.waga))/SUM(kp.waga)
	FROM oceny o, klasa_przedmiot kp,przedmioty p
	WHERE (o.uczen=@student)AND(kp.przedmiot=o.przedmiot)
	   AND (kp.rodzaj_klasy=@rodzaj_klasy)
	   AND (kp.rok=@rok) AND (p.id_przedmioty=o.przedmiot)
	   AND (o.data BETWEEN @begin_date AND @end_date)

	RETURN @avg*100
END
-----------------------------------------------------------------------------------------
GO

/*

SELECT *
FROM uczniowie

DECLARE @ReturnStatus INT
exec @ReturnStatus=pupil_in_school 89111901557

PRINT ' '
PRINT 'Return code = ' + CAST(@ReturnStatus AS CHAR(10))
IF(@ReturnStatus= 0)
	PRINT '0'
ELSE PRINT 'non 0'
*/



--SELECT * FROM klasa_przedmiot

--SELECT klasa,data_przyjecia from uczniowie where PESEL='90082501727'
--SELECT * FROM klasy  where (id_klasy=10)

--SELECT uczen, ilosc=count(*) from oceny group by uczen order by ilosc

SELECT o.id_oceny, p.przedmiot , p.ID_przedmioty, o.data
 FROM przedmioty p, oceny o, uczniowie u
	where (u.PESEL='90082501727') and
	(o.przedmiot=p.ID_przedmioty)
	   AND( o.uczen=u.PESEL)


DECLARE @ret FLOAT
--EXEC @ret=is_teacher_of_class_subject @teacher=3, @class=1,
	@subject=1
```

```
EXEC @ret=average_of_marks @student='90082501727',
        @end_date='9/9/2006'
--EXEC @ret=weight_of_mark @student='90082501727', @id_mark=389

SELECT @ret/100

/*
INSERT INTO klasa_przedmiot
        VALUES (5,2,3,3)
INSERT INTO klasa_przedmiot
        VALUES (6,2,3,2)
INSERT INTO klasa_przedmiot
        VALUES (9,2,3,1)
*/
/*SELECT @ret


SELECT * FROM przedmiot_klasa_nauczyciel*/

SELECT CONVERT(DATETIME,'1/1/1900')

-----trigery

CREATE TRIGGER nauczyciele_tytul
ON nauczyciele
FOR (INSERT,UPDATE)
AS
BEGIN
IF tytul = 'mgr'OR tytul = 'dr' OR tytul = 'lic.' OR tytul = 'prof.'
PRINT "OK"
ELSE
ROLLBACK TRANSACTION
PRINT "Nieprawidlowy tytul"
END

----------------------------------

CREATE TRIGGER klasy_ID_klasy
ON klasy
FOR DELETE
AS
BEGIN
IF EXISTS ( SELECT PESEL FORM uczniowie WHERE klasa = ID_klasy )
BEGIN
ROLLBACK TRANSACTION
PRINT "Nie mozna usunšć klasy,w której sš jeszcze uczniowie"
END
END

----------------------------------

CREATE TRIGGER klasy_rok_rozpoczecia
ON klasy
FOR INSERT
AS
BEGIN
IF rok_rozpoczecia<= YEAR(GETDATE())
BEGIN
PRINT "Rok rozpoczecia musi byc mniejszy niz aktualna data"
ROLLBACK TRANSACTION
END
END

----------------------------------

CREATE TRIGGER klasy_podrok
ON klasy
FOR (INSERT,UPDATE)
AS
BEGIN
IF pod_rok LIKE '[a-z]'
PRINT "OK"
ELSE
BEGIN
ROLLBACK TRANSACTION
PRINT "Podrok jest litera od a do z 1"
END
END

----------------------------------

CREATE TRIGGER uczniowie_pesel
ON uczniowie
FOR INSERT
AS
BEGIN
IF PESEL LIKE '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]'
PRINT "OK"
ELSE
BEGIN
ROLLBACK TRANSACTION
PRINT "Zla forma PESEL"
END
END

----------------------------------

CREATE TRIGGER uczniowie_data_urodzenia
ON uczniowie
FOR INSERT
AS
BEGIN
IF YEAR(data_ur) NOT BETWEEN (YEAR(GETDATE())-100) AND
        (YEAR(GETDATE())-10)
BEGIN
ROLLBACK TRANSACTION
PRINT "Zle podana data urodzenia"
END
END

----------------------------------

CREATE TRIGGER uczniowie_data_przyjecia
ON uczniowie
FOR INSERT
AS
BEGIN
IF data_przyjecia > GETDATE()
BEGIN
ROLLBACK TRANSACTION
PRINT "Zle podana data przyjecia"
END
END

----------------------------------

CREATE TRIGGER uczniowie_data_odejscia
ON uczniowie
FOR INSERT
AS
BEGIN
IF data_odejscia < data_przyjecia
BEGIN
ROLLBACK TRANSACTION
PRINT "Zle podana data przyjecia"
END
END

----------------------------------


----------------------------------

CREATE TRIGGER oceny_ocena
ON oceny
FOR INSERT
AS
BEGIN
IF ocena NOT BETWEEN 1 AND 6
BEGIN
ROLLBACK TRANSACTION
PRINT "Ocena przyjmuja wartosci od 1 do 6"
```

```
END
END

---------------------------------------

CREATE TRIGGER oceny_data
ON oceny
FOR INSERT
AS
BEGIN
IF data NOT BETWEEN (SELECT data_przyjecia FROM uczniowie
        WHERE PESEL = uczen AND GETDATE())
BEGIN
ROLLBACK TRANSACTION
PRINT "Nieodpowiednia data"
END
END

---------------------------------------

CREATE TRIGGER przedmiot_klasa_nauczyciel_godzin
ON przedmiot_klasa_nauczyciel
FOR INSERT
AS
BEGIN
IF godzin < 0
BEGIN
ROLLBACK TRANSACTION
PRINT "Ilosc godzin musi byc wieksza niz 0"
END
END

---------------------------------------

CREATE TRIGGER klasa_przedmioty_rok
ON klasa_przedmiot
FOR INSERT
AS
BEGIN
IF (rok < 1) OR (rok > 3)
BEGIN
ROLLBACK TRANSACTION
PRINT "Wybierz miedzy 1 ,2 lub 3."
END
END

---------------------------------------

CREATE TRIGGER klasa_przedmioty_waga
ON klasa_przedmiot
FOR INSERT
AS
BEGIN
IF (waga < 0) OR (waga > 100)
BEGIN
ROLLBACK TRANSACTION
PRINT "Waga miedzy 0 ,a 100"
END
END

---------------------------------------
CREATE TRIGGER stypendia_data_rozpoczecia
ON stypendia
FOR INSERT
AS
BEGIN
IF data_rozpoczecia > GETDATE()
BEGIN
ROLLBACK TRANSACTION
PRINT "Niepoprawna data"
END

END

---------------------------------------
CREATE TRIGGER stypendia_data_zakonczenia
ON stypendia
FOR INSERT
AS
BEGIN
IF data_zakonczenia < data_rozpoczecia + 30
BEGIN
ROLLBACK TRANSACTION
PRINT "Niepoprawna data(ma trwac przynajmniej 30 dni)"
END
END

---------------------------------------
CREATE TRIGGER stypendia_wartosc
ON stypendia
FOR INSERT
AS
BEGIN
IF wartosc < 0
BEGIN
ROLLBACK TRANSACTION
PRINT "Kwota musi byc wieksza od 0 !"
END
END

---------------------------------------

CREATE TRIGGER ocena_srednia_stypendium
ON oceny
FOR (INSERT,UPDATE)
AS
BEGIN
DECLARE @avg FLOAT
DECLARE @uczen CHAR(11)
SELECT @uczen=uczen
EXEC @avg=average_of_marks @uczen
IF @avg > 4.75
BEGIN
IF EXISTS ( SELECT wartosc FORM stypendia WHERE uczen = @uczen)
BEGIN
INSERT INTO stypendia
(uczen,data_rozpoczecia,data_zakonczenia,wartosc)
VALUES (@uczen,GETDATE(),GETDATE()+30,4.75)
END
END
END

---------------------------------------

CREATE TRIGGER stypendia_ilosc_ocen
ON stypendia
FOR (INSERT,UPDATE)
AS
BEGIN
DECLARE @ilocen INT
DECLARE @ilpotrzebnych INT
DECLARE @uczen CHAR(11)
DECLARE @klasa INT
SELECT @uczen = uczen
SELECT @klasa = SELECT klasa FROM uczniowie WHERE PESEL =
        @uczen /////??????????????????????
SELECT @ilocen = count_last_marks_of_student @uczen
SELECT @ilpotrzebnych = count_marks_for_class @klasa

IF @ilpotrzebnyc > @ilocen OR @ilpotrzebnyc < @ilocen
BEGIN
ROLLBACK TRANSACTION
PRINT "Uczen nie posiada odpowiedniej ilosci ocen!"
END
END

---------------------------------------

CREATE TRIGGER ocena_vs_klasaprzedmiot
ON oceny
FOR INSERT
AS
```

```
BEGIN
DECLARE @uczen INT
DECLARE @klasa INT
DECLARE @rodzajkl INT
DECLARE @przedmiot INT
SELECT @uczen = uczen
SET @klasa = (
SELECT klasa
FROM uczniowie
WHERE PESEL=@uczen
)
SET @rodzajkl = (
SELECT rodzaj_klasy
FROM klasy
WHERE ID_klasy=@klasa
)
SELECT @przedmiot = przedmiot

IF ( SELECT rodzaj_klasy FROM klasa_przedmiot WHERE
        przedmiot=@przedmiot) = @rodzajkl
PRINT "OK"
ELSE
BEGIN
ROLLBACK TRANSACTION
PRINT "Nie mozna dodac oceny z tego przedmiotu!"
END
END
```

Szkola_JK
Implementacja

Pełną implementację przeprowadziliśmy dla wybranej części funkcjonalności bazy danych. Wybraliśmy część, w której znajdują się największe zależności pomiędzy danymi – moduł dodawania ocen zależy zarówno od przedmiotów, nauczycieli, klasy do której należy uczeń oraz relacji klasa-przedmiot-nauczyciel, w której jest zapisany związek dozwolonych kombinacji tych trzech elementów – odwzorowanie rzeczywistych uprawnień do wystawiania ocen z danego przedmiotu pod warunkiem, że jest się nauczycielem danej klasy z danego przedmiotu.

Pozostała funkcjonalność została zaimplementowana w sposób uproszczony – brak sprawdzania spójności danych czy możliwości wyszukiwania wg różnych kryteriów. Spójność i integralność danych sprawdzana jest na poziomie bazy danych MS SQL Server zgodnie z definicją bazy.

Formularze

Pogrubioną czcionką *zaznaczone są te akcje na formularzach, które powodują otwarcie nowego formularza*

Zestaw funkcji w rozszerzonej części aplikacji

- **Zarządzanie uczniami**
  - **Podgląd ocen**
  - Dodawanie uczniów
  - Usuwanie uczniów
  - Edycja uczniów (zmiana klasy, danych o uczniu)
  - Szukanie uczniów (filtrowanie wg zadanych kryteriów)

- **Zarządzanie klasami**
  - Dodawanie klasy
  - Usuwanie klasy

    - Edycja klasy (zmiana profilu, wychowawcy)
    - Szukanie klasa (filtrowanie wg nr klasy, pod roku, profilu)

- **Zarządzanie ocenami**
    - **Dodawanie ocen**
    - Usuwanie ocen
    - Szukanie ocen

Uproszczony (chociaż w dużej mierze funkcjonalny) interfejs obejmuje pozostałe funkcje aplikacji:

- **Edycja powiązania przedmiot-klasa-nauczyciel**
- **Edycja nauczycieli**
- **Edycja profili klas**
- **Edycja przedmiotów**

Raporty

W aplikacji zostały stworzone raporty dotyczące bazy danych obejmują podstawowe i najczęściej używane zestawienia informacji na temat:

- **Wykaz ocen ucznia**
  (wybranego, z danego przedziału czasowego)

- **Zestawienie uczniów z ocenami niedostatecznymi**
  (którzy są jeszcze uczniami szkoły)

- **Uczniowie wg średniej**
  (uzyskanej w danym przedziale czasowym)

- Zestawienie **średniej ocen** wystawionych przez danego **nauczyciela** w danym przedziale czasowym

Szczegółowy opis formularzy

1) Zarządzanie ocenami

Właściwości formularza:

```
DefaultView = SingleForm
RekordSelector = False
NavigationBar = True
Movable = True
Modal = True
RecordSource = SELECT oceny.ID_oceny, oceny.ocena,
  oceny.data, oceny.nauczyciel AS oceny_nauczyciel,
  oceny.przedmiot AS oceny_przedmiot, oceny.uczen AS
  oceny_uczen, przedmioty.przedmiot,
  nauczyciele.nazwisko AS nazwisko_nauczyciela,
  uczniowie.nazwisko AS nazwisko_ucznia FROM oceny
  INNER JOIN uczniowie ON oceny.uczen = uczniowie.PESEL
  INNER JOIN nauczyciele ON oceny.nauczyciel =
  nauczyciele.ID_nauczyciele INNER JOIN przedmioty ON
  oceny.przedmiot = przedmioty.ID_przedmioty ORDER BY
  oceny.data DESC
EditRecord = True
AddRecord = False
DelRecord = True
```

a) CheckBox umożliwiający włączanie lub wyłączenie filtrowania. Zalezności pomiędzy poszczególnymi polami to AND:

W procedurze `FilterSelectUpdate`:

```
Me.Filter = "oceny_nauczyciel = '" &
Me.NauczycielCombo.Value & "'"
Me.Filter = Me.Filter & " AND oceny_uczen = '" &
Me.UczenCombo.Value & "' "
Me.Filter = Me.Filter & " AND oceny_przedmiot = '" &
Me.PrzedmiotCombo.Value & "' "
```

b) ComboBox wybór ucznia którego oceny zostaną wyświetlona, uczniowie posortowani wg nazwiska. Obok ucznia wyświetlana informacja o klasie do której chodzi i numerze PESEL

   `Me.UczenCombo` - nazwa pola przechowującego dane o uczniu. Przy odwoływaniu się do wartości pola zwracany jest id_ucznia

   Dane czytane z tabeli uczniowie z wykorzystaniem widoku `uczniowie opisowo` - wypisywane są informacje o uczniu: imię nazwisko klasa do której chodzi uczeń oraz numer PESEL. Nie są wypisywani uczniowie, którzy zakończyli edukację

c) ComboBox wybór nauczyciela – zostaną wyświetlone tylko oceny wystawione przez danego nauczyciela, posortowanie wg nazwiska nauczyciela

   `Me.NauczycielCombo` – nazwa pola przechowującego dane o nauczycielu. Przy odwoływaniu się do wartości pola zwracany jest id_nauczyciela

   Dane czytane z widoku `nauczyciele opisowo` – wypisywane są takie informacje jak imię nazwisko i tytuł.

d) ComboBox wybór przedmiotu – zostaną wyświetlone oceny wystawione z danego przedmiotu, posortowane przedmiotu wg nazwy

   `Me.przedmiotCombo` - nazwa pola przechowującego dane o przedmiocie. Przy odwoływaniu się do wartości pola zwracany jest id_przedmiotu

   Dane czytane z widoku `przedmioty_opisowo` – zawarte informacje o przedmiocie z tabeli przedmioty – posortowane alfabetycznie

e) CheckBox zaawansowane pozwala na ręczną edycje filtru. Po naciśnięciu klawisz Enter wpisany filtr staje się bieżącym filtrem

   W procedurze `FilterTextUpdate`

```
Me.use_filter.Value = True
Me.Filter = Me.FilterText
Me.FilterOn = True
```

f) Id oceny – pole zablokowane tylko do odczytu – wewnętrzne pole bazy danych identyfikujące jednoznacznie daną ocenę w tabeli *oceny*

g) Ocena – wartość z zakresu 1-6 (ComboBox) – można ją edytować (wartość przyznanej oceny).

   Zakres dostępnych ocen (1-6) jest zapisany w definicji bazy danych oraz jest sprawdzany poprzez maskę w programie Access

h) Data – data wystawienia oceny – można edytować.

   Format oceny musi być akceptowany przez bazę danych MS SQL Server. Maska wprowadzania jest w Accessie ustawiona na `dd/mm/rr`

i) ComboBox nauczyciel – nauczyciel, który wystawił daną ocenę – przy zmianie nie są sprawdzanie warunki integralności – czy dany nauczyciel uczył dana klasę z danego przedmiotu, co umożliwia edycję również wtedy kiedy ta relacja już nie istnieje w bazie danych

   Dane do wyświetlania pobierane są z widoku `nauczyciele_opisowo` – wyświetlane są następujące pola `tytuł, imię nazwisko`. posortowane wg nazwiska nauczyciela – wykorzystywana tabela nauczyciele

j) ComboBox uczeń – uczeń któremu została wystawiona dana ocena. przy zmianie nie są sprawdzanie warunki integralności – czy dany nauczyciel uczył dana klasę z danego przedmiotu, co umożliwia edycję również wtedy kiedy ta relacja już nie istnieje w bazie danych

   Dane do wyświetlenie pobierane z widoku `uczniowie opisowo`. Wyświetlane są następujące kolumny; `imię, nazwisko, klasa, PESEL` ucznia. Uczniowie posortowanie są wg klas, następnie wg nazwisk.

k) ComboBox przedmiot – przedmiot którego dotyczy dana ocena. przy zmianie nie są sprawdzanie warunki integralności – czy dany nauczyciel uczył dana klasę z danego przedmiotu, co umożliwia edycję również wtedy kiedy ta relacja już nie istnieje w bazie danych

   Dane do wysiedlenia pobierane są z widoku `przedmioty opisowo`. Wyświetlana jest tylko nazwa przedmiotu. Dane posortowane alfabetycznie rosnąco.

l) Przycisk *Dodaje ocenę* otwiera formularz dodawania oceny

   W procedurze `dodajOceneOpen`:

   ```
   stDocName = "dodaj ocene"
   ```

```
DoCmd.OpenForm stDocName, , , stLinkCriteria
```

2) Dodaj ocenę

Właściwości formularza:

```
DefaultView = SingleForm
RekordSelector = False
NavigationBar = False
Movable = True
Modal = True
RecordSource = Null
EditRecord = False
AddRecord = False
DelRecord = False
```

a) ComboBox klasa – wybór klasy ucznia któremu chce się wystawić ocenę. Aktywne staje się wtedy pole wyboru przedmiotu.

W procedurze `KlasaComboChange`:

'zapytanie aktualizujące pola przedmiotu – patrz niżej

```
Me.PrzedmiotCombo.RowSource = "SELECT DISTINCT
przedmioty.ID_przedmioty, przedmioty.przedmiot " & _
    "FROM         przedmioty INNER JOIN
przedmiot_klasa_nauczyciel ON
przedmioty.ID_przedmioty =
przedmiot_klasa_nauczyciel.przedmiot " & _
    " WHERE (przedmiot_klasa_nauczyciel.klasa = " &
Me.KlasaCombo.Value & _
    " )ORDER BY przedmioty.przedmiot,
przedmioty.ID_przedmioty "
```

'zapytanie wyświetlające uczniów w bazie

```
Me.UczniowieList.RowSource = " SELECT nazwisko, imie,
PESEL FROM uczniowie " & _
    " WHERE (klasa = " & Me.KlasaCombo.Value & ")
ORDER BY nazwisko, imie "
```

W celu odświeżenia wyświetlanych wartości wywoływana jest funkcja

```
requery()
Me.Reqery()
```

b) ComboBox przedmiot – wybór przedmiotu z którego chce się wystawić ocenę. Aktywne staję się wtedy pole wyboru nauczyciela (wyświetlane są tylko te przedmioty, które dotyczą przedmiotów, które są uczone w danej klasie – wyprowadzane są zależności z relacji przedmiot-klasa-nauczyciel)

W procedurze `PrzedmiotComboChange`:
'zmiana wartości pola combo odpowiedzialnego za wyświetlanie nauczycieli uczących danego przedmiotu

```
Me.NauczycielCombo.RowSource = " SELECT
[nauczyciele opisowo].nauczyciel_all, [nauczyciele
opisowo].ID_nauczyciele " & _
    " FROM         [nauczyciele opisowo] INNER JOIN "
& _
    " przedmiot_klasa_nauczyciel ON [nauczyciele
opisowo].ID_nauczyciele =
przedmiot_klasa_nauczyciel.nauczyciel " & _
    " WHERE (przedmiot_klasa_nauczyciel.klasa = " &
Me.KlasaCombo.Value & _
     " ) And (przedmiot_klasa_nauczyciel.przedmiot =
" & Me.PrzedmiotCombo.Value & ") "
```

'zmiana zawartości pola wyświetlającego list ocen

```
Me.OcenyList.RowSource = "SELECT
uczniowie.nazwisko + ' ' + uczniowie.imie AS
uczen_all, oceny.ocena, oceny.data, " & _
        " nauczyciele.tytul + ' ' + nauczyciele.imie
+ ' ' + nauczyciele.nazwisko AS nauczyciel_all,
oceny.ID_oceny " & _
        " FROM         oceny INNER JOIN uczniowie ON
oceny.uczen = uczniowie.PESEL INNER JOIN " & _
        " klasy ON uczniowie.klasa = klasy.ID_klasy
INNER JOIN " & _
        " nauczyciele ON oceny.nauczyciel =
nauczyciele.ID_nauczyciele " & _
        " WHERE (klasy.ID_klasy = " &
Me.KlasaCombo.Value & _
        ") And (oceny.przedmiot = " &
Me.PrzedmiotCombo.Value & ")" & _
        " ORDER BY oceny.data DESC,
uczniowie.nazwisko "
```

c) ComboBox nauczyciel – wybór nauczyciela, który wystawia daną ocenę. Wyświetlani są tylko nauczyciele, którzy uczą danej klasy z danego przedmiotu – na podstawie przedmiot-klasa-nauczyciel

   Zmiana tego pola nie wymaga zmiany zawartości innych pól – nie jest wysyłane żadne zapytanie do bazy danych

d) ListBox uczniowie – uczniowie, którzy należą do wybranej powyżej klasy. Należy zaznaczyć ucznia, któremu chce się dodać ocenę.

   Wartości rekordów są zmieniane w czasie zmiany pola KlasaCombo. Jeśli nie jest zaznaczone żadne pole a użytkownik próbuje dodać ocene wyświetlany jest komunikat o błędzie

e) ComboBox Ocena – wartość 1-6, która jest wartości uzyskanej oceny z danego przedmiotu, wybranego ucznia, wystawiona przez wybranego nauczyciela
   Wartości możliwe do wpisania są zapisane w definicji bazy danych. Również pole Combo bez możliwości edycji uniemożliwia wpisanie niewłaściwej wartości. Domyślna wartość to 1

f) Kalendarz – Komponent umożliwiający wybranie daty wystawienia oceny. Aby komponent mógłbyc widoczny musi być on zainstalowany jako dodatek do programy MS Access. W niektórych wersjach pakietu MS Office nie jest on domyślnie instalowany

   Z niewiadomych powodów komponent Kalendarz nie zawsze aktualizuje wartość daty po kliknięciu odpowiedniego pola

g) ListBox oceny – lista ocen wystawiona z danego przedmiotu danej klasie przez różnych nauczycieli.
   Wartość listy modyfikowana jest w czasie zmiany klasy lub przedmiotu – patrz odpowiednie pola Combo

h) Dodaj ocene przycisk – nacisniecie powoduje dodanie nowego rekordy do tabeli oceny

   Wysyłane jest następujące zapytanie do bazy danych:

```
addSQL = "INSERT INTO oceny (uczen, ocena, data,
nauczyciel, przedmiot )" & _ " VALUES ( '" &
Me.UczniowieList.Value & "', " & Me.OcenaCombo.Value
& _ ", '" & Me.Calendar.Day & "/" & Me.Calendar.Month
& "/" & Me.Calendar.Year & _ "', " &
```

```
  Me.NauczycielCombo.Value & ", " & _
  Me.PrzedmiotCombo.Value & ")"
  DoCmd.RunSQL addSQL
```

3) Zarządzanie klasami

Właściwości formularza:

```
DefaultView = SingleForm
RekordSelector = False
NavigationBar = True
Movable = True
Modal = True
RecordSource = SELECT klasy.rok_rozpoczecia AS
  klasy_rok_rozpoczecia, klasy.pod_rok AS
  klasy_pod_rok, klasy.ID_klasy, nauczyciele.imie,
  nauczyciele.nazwisko AS nauczyciele_nazwisko,
  nauczyciele.tytul, rodzaje_klas.rodzaj_klasy AS
  rodzaje_klas_rodzaj_klasy, klasy.rodzaj_klasy AS
  klasy_rodzaj_klasy, klasy.wychowawca AS
  klasy_wychowawca FROM nauczyciele INNER JOIN
  rodzaje_klas INNER JOIN klasy ON
  rodzaje_klas.ID_rodzaje_klas = klasy.rodzaj_klasy ON
  nauczyciele.ID_nauczyciele = klasy.wychowawca
EditRecord = True
AddRecord = True
DelRecord = True
```

a) CheckBox umożliwiający włączanie lub wyłączenie filtrowania. Zależności pomiędzy poszczególnymi polami to AND

   W procedurze `FilterOnClick`:

```
Me.Filter = "klasy_rok_rozpoczecia < '" & Year(Date)
- 2 & "'"
Me.Filter = Me.Filter & " AND klasy_pod_rok = '" &
Me.podrok_filter.Value & "' "
Me.Filter = Me.Filter & " AND klasy_rodzaj_klasy = '"
& Me.profil_kl_filter.Value & "' "
```

b) ComboBox rok klasy – umożliwia filtrowanie rekordów według roku klasy- 1-3 lub klasa która zakończyła naukę w szkole.

Dostępne wartości klasy są zapisane w definicji bazy danych oraz są sprawdzane poprzez wykorzystanie pola combo uniemożliwiającego wprowadzenie nowych (niewłaściwych) danych.

Zmiana wartości pola powoduje wywołanie procedury `KlasaComboUpdate` – wywoływana jest aktualizacja pola filtra – symulowanie wyłączenia i włączenia filtra

c) ComboBox rodzaj klasy – wybór profilu klasy – dostępne możliwości pobierane z tabeli *`profile klas`*

Zmiana wartości pola powoduje wywołanie procedury `ProfilComboUpdate` – wywoływana jest aktualizacja pola filtra – symulowanie wyłączenia i włączenia filtra

d) Pole Filtr – umożliwia ręczną edycję filtra dla rekordów. Filtr zostaje zaktualizowany po naciśnięciu Enter'a

```
Me.Filter = Me.FilterTekst.Value
Me.FilterOn = True
```

e) Id_klasy – wewnętrzne pola bazy danych – zablokowane tylko do odczytu.

Jest to indeks główny fizyczny (clustered) tabeli klasy. Autonumerowanie zwiększające o jeden

f) Rok rozpoczęcia – rok rozpoczęcia nauki przez daną klasę (we wrześniu danego roku)

Wartość z zakresu 1990 – 2100. Ograniczenie dopuszczalnych wartości ma na celu uniemożliwienie przypadkowego wpisanie nieprawdopodobnej daty rozpoczęcia nauki przez klasę.

g) Pod rok – oznaczenie klasy z zakresy [a-z]

W definicji bazy danych zapisane jest ograniczenie małych liter (lower case) od a do z. Poprawność wprowadzanych danych jest sprawdzania też przez maskę wpisywania w programie Access

h) Rok klasy – pole obliczane na postawie pola rok_rozpoczecia. Z niewiadomych powodów Access nie zawsze wywołuje procedurę związaną ze zdarzeniem *przy aktualizacji*. Pola tego nie można zmieniać bezpośrednio– należy zmienić rok rozpoczęcia dla danej klasy.

Pole to jest aktualizowane na podstawie pola `rok_klasy` : `Me.RokRozpEdit.Value`

i) ComboBox Rodzaj klasy – tekstowo opisany rodzaj klasy – dane poprzez widok pobierane są z tabeli `rodzaje klas`. Wartości pól są posortowane alfabetycznie. Wykorzystane pole z Combo uniemożliwia wprowadzenie profilu, który nie jest zdefiniowany w tablicy `rodzaje klas`

Odwołując się do pola `Me.RodzajKlasyCombo` zwracaną wartością jest `id_rodzaje_klas` (a nie nazwa rodzaju klasy). `id_rodzaje_klas` jest to indeks główny tabeli `rodzaje klas`. Powoduje to łatwiejsze odwoływanie się do odpowiednich rodzajów klas oraz umożliwia filtrowanie rekordów

j) ComboBox wychowawca – wychowawca danej klasy – poprzez widok `nauczycie_opisowo` dane pobierane z tabeli `nauczyciele`

Ograniczenie nałożone w definicji bazy danych to posiadanie co najwyżej jednej klasy, w której dany nauczyciel jest wychowawcą. W przypadku próby niedozwolonej aktualizacji serwer SQL zwróci błąd naruszenia warunków integralności.

k) Podformularz uczniowie z klasy – zawiera listę uczniów z danej klasy. W tym podformularzu nie można edytować danych uczniów – należy to robić poprzez formularz zarządzanie uczniami. Wykorzystany jest widok `uczniowie_opisowo`

```
DefaultView = Sheet
RecordSource = SELECT "uczniowie z klasy
  opisowo"."PESEL", "uczniowie z klasy
  opisowo"."uczen_all", "uczniowie z klasy
  opisowo"."nr_legitymacji", "uczniowie z klasy
  opisowo"."data_ur", "uczniowie z klasy
  opisowo"."data_przyjecia", "uczniowie z klasy
  opisowo"."data_odejscia", "uczniowie z klasy
  opisowo"."klasa" FROM "uczniowie z klasy opisowo"
```

4) Formularz nauczyciele

```
DefaultView = ContinousForm
RekordSelector = False
NavigationBar = True
```

```
Movable = True
Modal = True
RecordSource = nauczyciele 'tablica nauczycieli w bazie
EditRecord = True
AddRecord = True
DelRecord = True
```

a) ID nauczyciele – wewnętrzny identyfikator nauczyciela – nie można edytować

   Jest to pole indeksu głównego fizycznego (clustered) w tablicy `nauczyciele`. Autonumerowanie zwiększane o 1

b) tytuł – tytuł naukowy danego nauczyciela np. mgr , dr itd. Tytuł jest wyświetlany zawsze przed nazwiskiem tam gdzie jest możliwość wyboru nauczyciela

c) Imię – imię nauczyciela

d) Nazwisko – nazwisko nauczyciela ograniczenia do liter i '-'.

   W definicji bazy jest ograniczenie do długości zapisanego tekstu do 30 znaków – CHAR[30].W aplikacji MS Access sprawdzane są ;długość wpisywanego tekstu oraz użyte znaki (znaki alfabetu i '-')

5) Relacja przedmiot-klasa-nauczyciel

```
DefaultView = ContinousForm
RekordSelector = False
NavigationBar = True
Movable = True
Modal = True
RecordSource = przedmiot_klasa_nauczyciel 'tablica w
   bazie przechowująca relację przemiot-klasa-nauczyciel
EditRecord = True
AddRecord = True
DelRecord = True
```

a) przedmiot – przedmiot dla danej relacji

   W czasie odwoływania się do wartości tego pola `Me.PrzedmiotCombo.Value` zwracany jest indeks `id_przedmioty` danego przedmiotu w tabeli `przedmioty`

b) klasa – klasa, w której będzie uczony dany przedmiot

   W czasie odwoływania się do wartości tego pola `Me.KlasaCombo.Value` zwracany jest indeks `id_klasy` danego przedmiotu w tabeli `klasy`

c) nauczyciel – nauczyciel, który będzie uczył w danej klasie danego przedmiotu.

   W czasie odwoływania się do wartości tego pola `Me.NauczycielCombo.Value` zwracany jest indeks `id_nayczyciel` danego nauczyciela w tabeli `nauczyciele`

   Nie może się powtarzać trójka przedmiot, klasa, nauczyciel – błąd integralności danych – odpowiedni błąd zostanie wysłany przez serwer SQL

d) godzin – pole obowiązkowe wskazujące ile godzin uczy dany nauczyciel w danej klasie. Domyślnie wartość 1

   W definicji bazy danych nie istnieje wartość domyślna – jest ona ustawiona tylko w aplikacji MS Access jako maska wprowadzania umożliwiająca wprowadzenie liczby z zakresu 1 do 50. Jeśli wartość wysyłana w zapytaniu jest nieprawidłowa zwracany jest błąd dodania rekordu.

6) Rodzaje klas

```
DefaultView = ContinousForm
RekordSelector = True
NavigationBar = True
Movable = True
Modal = True
RecordSource = rodzaje_klas ' tablica w bazie
  przechowująca dane o rodzajach klas
EditRecord = True
AddRecord = True
DelRecord = True
```

a) ID rodzaju – wewnętrzna reprezentacja bazy danych dla danego rodzaju klasy

   W definicji bazy danych jest to pole integer – autonumerowanie o 1

b) rodzaj klasy – tekstowy opis danego rodzaju klasy – nie może się powtarzać.

   Nie można usunąć danego rodzaju klasy jeśli istnieje klasa o danym profilu

W definicji bazy długość pola jest ograniczona do 30 znaków (CHAR[30]). Długość wpisanej nazwy profilu jest też sprawdzana na poziomie aplikacji w MS Access

7) Przedmioty

```
DefaultView = ContinousForm
RekordSelector = False
NavigationBar = True
Movable = True
Modal = True
RecordSource = przedmioty 'tablica w bazie
  przechowująca dane o poszczególnych przedmiotach
EditRecord = True
AddRecord = True
DelRecord = True
```

a) ID przedmiotu – wewnętrzna reprezentacja przedmiotu – tylko do odczytu

   W definicji bazy danych pole `integer` autonumerowane zwiększające wartość o 1

b) Przedmiot – tekstowy opis przedmiotu. Nie może być dwóch przedmiotów o tych samych nazwach.

   Długość nazwy przedmiotu została ograniczona w definicji bazy danych do 30 znaków (`CHAR[30]`). Długość wpisanej nazwy przedmiotu jest też sprawdzana na poziomie aplikacji MS Access.

8) Formularz usuń klasę/ucznia/ocenę

```
DefaultView = ContinousForm
RekordSelector = False
NavigationBar = True
Movable = True
Modal = True
RecordSource = klasy/oceny/uczniowie
EditRecord = True
AddRecord = True
DelRecord = True
```

a) umożliwia usuwanie danych rekordów z bazy danych pod warunkiem ze nie narusza to związków integralności określonych w definicji bazy danych

W każdym formularzu jest pole do filtrowania wszystkich rekordów:
-formularz usuń klasę : rok klasy , `pod_rok, profil_klasy`
-formularz usuń ucznia : nazwisko, `klasa, profil, rok_rozpoczęcia`
-formularz usuń ocene : `uczen_nazwisko, uczen_PESEL, uczen_klasa, ocena_data, ocena_ocena, ocena_nauczyciel`

9) Zarządzanie ocenami

Właściwości formularza:

```
DefaultView = SingleForm
RekordSelector = False
NavigationBar = True
Movable = True
Modal = True
RecordSource = SELECT oceny.ID_oceny, oceny.ocena,
  oceny.data, oceny.nauczyciel AS oceny_nauczyciel,
  oceny.przedmiot AS oceny_przedmiot, oceny.uczen AS
  oceny_uczen, przedmioty.przedmiot,
  nauczyciele.nazwisko AS nazwisko_nauczyciela,
  uczniowie.nazwisko AS nazwisko_ucznia FROM oceny
  INNER JOIN uczniowie ON oceny.uczen = uczniowie.PESEL
  INNER JOIN nauczyciele ON oceny.nauczyciel =
  nauczyciele.ID_nauczyciele INNER JOIN przedmioty ON
  oceny.przedmiot = przedmioty.ID_przedmioty ORDER BY
  oceny.data DESC
EditRecord = True
AddRecord = False
DelRecord = True
```

a) CheckBox umożliwiający włączanie lub wyłączenie filtrowania. Zalezności pomiędzy poszczególnymi polami to AND:

W procedurze `FilterSelectUpdate`:

```
Me.Filter = "oceny_nauczyciel = '" &
Me.NauczycielCombo.Value & "'"
```

```
Me.Filter = Me.Filter & " AND oceny_uczen = '" &
Me.UczenCombo.Value & "' "
Me.Filter = Me.Filter & " AND oceny_przedmiot = '" &
Me.PrzedmiotCombo.Value & "' "
```

b) ComboBox wybór ucznia którego oceny zostaną wyświetlona, uczniowie posortowani wg nazwiska. Obok ucznia wyświetlana informacja o klasie do której chodzi i numerze PESEL

   `Me.UczenCombo` - nazwa pola przechowującego dane o uczniu. Przy odwoływaniu się do wartości pola zwracany jest id_ucznia

   Dane czytane z tabeli uczniowie z wykorzystaniem widoku `uczniowie opisowo` - wypisywane są informacje o uczniu: imię nazwisko klasa do której chodzi uczeń oraz numer PESEL. Nie są wypisywani uczniowie, którzy zakończyli edukację

c) ComboBox wybór nauczyciela – zostaną wyświetlone tylko oceny wystawione przez danego nauczyciela, posortowanie wg nazwiska nauczyciela

   `Me.NauczycielCombo` – nazwa pola przechowującego dane o nauczycielu. Przy odwoływaniu się do wartości pola zwracany jest id_nauczyciela

   Dane czytane z widoku `nauczyciele opisowo` – wypisywane są takie informacje jak imię nazwisko i tytuł.

d) ComboBox wybór przedmiotu – zostaną wyświetlone oceny wystawione z danego przedmiotu, posortowane przedmiotu wg nazwy

   `Me.przedmiotCombo` - nazwa pola przechowującego dane o przedmiocie. Przy odwoływaniu się do wartości pola zwracany jest id_przedmiotu

   Dane czytane z widoku `przedmioty_opisowo` – zawarte informacje o przedmiocie z tabeli przedmioty – posortowane alfabetycznie

e) CheckBox zaawansowane pozwala na ręczną edycje filtru. Po naciśnięciu klawisz Enter wpisany filtr staje się bieżącym filtrem

   W procedurze `FilterTextUpdate`

```
Me.use_filter.Value = True
Me.Filter = Me.FilterText
```

```
Me.FilterOn = True
```

f) Id oceny – pole zablokowane tylko do odczytu – wewnętrzne pole bazy danych identyfikujące jednoznacznie daną ocenę w tabeli *oceny*

g) Ocena – wartość z zakresu 1-6 (ComboBox) – można ją edytować (wartość przyznanej oceny).

   Zakres dostępnych ocen (1-6) jest zapisany w definicji bazy danych oraz jest sprawdzany poprzez maskę w programie Access

h) Data – data wystawienia oceny – można edytować.

   Format oceny musi być akceptowany przez bazę danych MS SQL Server. Maska wprowadzania jest w Accessie ustawiona na `dd/mm/rr`

i) ComboBox nauczyciel – nauczyciel, który wystawił daną ocenę – przy zmianie nie są sprawdzanie warunki integralności – czy dany nauczyciel uczył dana klasę z danego przedmiotu, co umożliwia edycję również wtedy kiedy ta relacja już nie istnieje w bazie danych

   Dane do wyświetlania pobierane są z widoku `nauczyciele_opisowo` – wyświetlane są następujące pola `tytuł, imię nazwisko`. posortowane wg nazwiska nauczyciela – wykorzystywana tabela nauczyciele

j) ComboBox uczeń – uczeń któremu została wystawiona dana ocena. przy zmianie nie są sprawdzanie warunki integralności – czy dany nauczyciel uczył dana klasę z danego przedmiotu, co umożliwia edycję również wtedy kiedy ta relacja już nie istnieje w bazie danych

   Dane do wyświetlenie pobierane z widoku `uczniowie opisowo`. Wyświetlane są następujące kolumny; `imię, nazwisko, klasa, PESEL` ucznia. Uczniowie posortowanie są wg klas, następnie wg nazwisk.

k) ComboBox przedmiot – przedmiot którego dotyczy dana ocena. przy zmianie nie są sprawdzanie warunki integralności – czy dany nauczyciel uczył dana klasę z danego przedmiotu, co umożliwia edycję również wtedy kiedy ta relacja już nie istnieje w bazie danych

   Dane do wysiedlenia pobierane są z widoku `przedmioty opisowo`. Wyświetlana jest tylko nazwa przedmiotu. Dane posortowane alfabetycznie rosnąco.

l) Przycisk *Dodaje ocenę* otwiera formularz dodawania oceny

W procedurze `dodajOceneOpen`:

```
stDocName = "dodaj ocene"
DoCmd.OpenForm stDocName, , , stLinkCriteria
```

Raporty – szczegółowy opis

10) Wykaz ocen – raport

```
RecordSource = SELECT     dbo.uczniowie_opisowo.uczen_all
   AS uczniowie_nazwisko, dbo.uczniowie_opisowo.rok_klasy,
   dbo.uczniowie_opisowo.PESEL, dbo.oceny.data,
   dbo.oceny.ocena, dbo.[nauczyciele
   opisowo].nauczyciel_all,dbo.uczniowie_opisowo.nr_legity
   macji, dbo.przedmioty.przedmiot FROM dbo.oceny INNER
   JOIN dbo.uczniowie_opisowo ON dbo.oceny.uczen =
   dbo.uczniowie_opisowo.PESEL INNER JOIN dbo.[nauczyciele
   opisowo] ON dbo.oceny.nauczyciel = dbo.[nauczyciele
   opisowo].ID_nauczyciele INNER JOIN dbo.przedmioty ON
   dbo.oceny.przedmiot = dbo.przedmioty.ID_przedmioty
   WHERE (dbo.uczniowie_opisowo.PESEL =@uczen ) AND
   (dbo.oceny.data BETWEEN @data_poczatkowa AND
   @data_koncowa)
ImputParameters =
   @uczen=Forms!Raporty!UczenCombo,@data_poczatkowa=Forms!
   Raporty!DataPoczText,@data_koncowa=Forms!Raporty!DataKo
   nText
```

a) data początkowa – data od której zostaną wyświetlone oceny

   Z nieznanych powodów komponent Kalendarz nie zawsze wywołuje metodę `OnUpdate`. Aby ominąć tą niedogodność do aktualizacji pola wywoływana jest też metoda `OnFocus i OnDeactivate`, których ciało jest takie samo jak metody `OnUpdate`. Dala upewnienia się, że została wybrana właściwa data wyświetlane jest jako etykieta (`BeginDateLabel`) aktualnie zwracana data przez komponent Kalendarz

b) data końcowa – data po której wystawione oceny nie będą wyświetlane

   Z nieznanych powodów komponent Kalendarz nie zawsze wywołuje metodę `OnUpdate`. Aby ominąć tą niedogodność do aktualizacji pola wywoływana jest też metoda `OnFocus i OnDeactivate`, których ciało jest takie samo jak metody `OnUpdate`. Dala upewnienia się, że została wybrana właściwa data wyświetlane jest jako etykieta (`BeginDateLabel`) aktualnie zwracana data przez komponent Kalendarz

c) uczeń – ComboBox – pole umożliwia wyświetlenie ucznia którego będzie dotyczył wykaz ocen.

Dane ucznia są pobierane z widoku uczniowie_opisowo. Lista posortowana jest wg nazwisk uczniów. Wyświetlane są następujące pola: imie, nazwisko, PESEL, klasa

11) Wykaz uczniów z niedostatecznymi ocenami

```
RecordSource = SELECT uczniowie.imie + ' ' +
   uczniowie.nazwisko AS uczen, przedmioty.przedmiot,
   [nauczyciele opisowo].nauczyciel_all, [klasy
   opisowo].rok_klasy, uczniowie.nr_legitymacji,
   uczniowie.PESEL, uczniowie.data_przyjecia, oceny.data
   FROM oceny INNER JOIN przedmioty ON oceny.przedmiot =
   przedmioty.ID_przedmioty INNER JOIN [nauczyciele
   opisowo] ON oceny.nauczyciel = [nauczyciele
   opisowo].ID_nauczyciele INNER JOIN uczniowie ON
   oceny.uczen = uczniowie.PESEL INNER JOIN [klasy
   opisowo] ON uczniowie.klasa = [klasy
   opisowo].ID_klasy WHERE (uczniowie.data_odejscia IS
   NULL) AND (oceny.ocena = 1)
InputParameters = Null
```

wyświetla uczniów, którzy chodzą jeszcze do szkoły ale mają oceny niedostateczne

a) data początkowa – data od której zostaną wyświetlone oceny

Z nieznanych powodów komponent Kalendarz nie zawsze wywołuje metodę OnUpdate. Aby ominąć tą niedogodność do aktualizacji pola wywoływana jest też metoda OnFocus i OnDeactivate, których ciało jest takie samo jak metody OnUpdate. Dala upewnienia się, że została wybrana właściwa data wyświetlane jest jako etykieta (BeginDateLabel) aktualnie zwracana data przez komponent Kalendarz

b) data końcowa – data po której wystawione oceny nie będą wyświetlane

Z nieznanych powodów komponent Kalendarz nie zawsze wywołuje metodę OnUpdate. Aby ominąć tą niedogodność do aktualizacji pola wywoływana jest też metoda OnFocus i OnDeactivate, których ciało jest takie samo jak metody OnUpdate. Dala upewnienia się, że została wybrana właściwa data wyświetlane jest jako etykieta (BeginDateLabel) aktualnie zwracana data przez komponent Kalendarz

12) Wykaz średnich ocen wystawionych przez nauczycieli

```
RecordSource = SELECT [nauczyciele
  opisowo].nauczyciel_all, AVG(oceny.ocena * 100) AS
  srednia_ocen, przedmioty.przedmiot FROM oceny INNER
  JOIN [nauczyciele opisowo] ON oceny.nauczyciel =
  [nauczyciele opisowo].ID_nauczyciele INNER JOIN
  przedmioty ON oceny.przedmiot =
  przedmioty.ID_przedmioty GROUP BY [nauczyciele
  opisowo].nauczyciel_all, przedmioty.przedmiot
ImputParameters =
  @uczen=Forms!Raporty!UczenCombo,@data_poczatkowa=Form
  s!Raporty!DataPoczText,@data_koncowa=Forms!Raporty!Da
  taKonText
```

wyświetlane są średnie z ocen liczonych w danym przedziale - grupowane wg przedmiotów dla danego nauczyciela

a) data początkowa – data od której zostaną wyświetlone oceny

Z nieznanych powodów komponent Kalendarz nie zawsze wywołuje metodę `OnUpdate`. Aby ominąć tą niedogodność do aktualizacji pola wywoływana jest też metoda `OnFocus i OnDeactivate`, których ciało jest takie samo jak metody `OnUpdate`. Dala upewnienia się, że została wybrana właściwa data wyświetlane jest jako etykieta (`BeginDateLabel`) aktualnie zwracana data przez komponent Kalendarz

b) data końcowa – data po której wystawione oceny nie będą wyświetlane

Z nieznanych powodów komponent Kalendarz nie zawsze wywołuje metodę `OnUpdate`. Aby ominąć tą niedogodność do aktualizacji pola wywoływana jest też metoda `OnFocus i OnDeactivate`, których ciało jest takie samo jak metody `OnUpdate`. Dala upewnienia się, że została wybrana właściwa data wyświetlane jest jako etykieta (`BeginDateLabel`) aktualnie zwracana data przez komponent Kalendarz

13) Wykaz najlepszych uczniów

```
RecordSource = SELECT     AVG(oceny.ocena * 100) AS srednia,
  uczniowie.imie + ' ' + uczniowie.nazwisko AS uczen_all,
  COUNT(oceny.ocena) AS ilosc_ocen, [klasy
  opisowo].rok_klasy, uczniowie.PESEL, [klasy
```

```
   opisowo].rodzaj_klasy, [klasy opisowo].nauczyciel_all,
   uczniowie.nr_legitymacji, uczniowie.nazwisko,
   uczniowie.klasa AS ID_klasy FROM        uczniowie INNER
   JOIN oceny ON uczniowie.PESEL = oceny.uczen INNER JOIN
   [klasy opisowo] ON uczniowie.klasa = [klasy
   opisowo].ID_klasy GROUP BY uczniowie.nazwisko,
   uczniowie.imie, [klasy opisowo].rok_klasy,
   uczniowie.PESEL, [klasy opisowo].rodzaj_klasy, [klasy
   opisowo].nauczyciel_all, uczniowie.nr_legitymacji,
   uczniowie.klasa
InputParameters = Null
```

wyświetlane są średnie uczniów posortowane malejąco – przejrzyste spojrzenie na średnie, które są podstawą przyznania stypendium

a) data początkowa – data od której zostaną wyświetlone oceny

   Z nieznanych powodów komponent Kalendarz nie zawsze wywołuje metodę `OnUpdate`. Aby ominąć tą niedogodność do aktualizacji pola wywoływana jest też metoda `OnFocus i OnDeactivate`, których ciało jest takie samo jak metody `OnUpdate`. Dala upewnienia się, że została wybrana właściwa data wyświetlane jest jako etykieta (`BeginDateLabel`) aktualnie zwracana data przez komponent Kalendarz

b) data końcowa – data po której wystawione oceny nie będą wyświetlane

   Z nieznanych powodów komponent Kalendarz nie zawsze wywołuje metodę `OnUpdate`. Aby ominąć tą niedogodność do aktualizacji pola wywoływana jest też metoda `OnFocus i OnDeactivate`, których ciało jest takie samo jak metody `OnUpdate`. Dala upewnienia się, że została wybrana właściwa data wyświetlane jest jako etykieta (`BeginDateLabel`) aktualnie zwracana data przez komponent Kalendarz

Dane testowe

Dane testowe zostały wygenerowane na postawie listy imion i nazwisk znalezionej w Internecie. Ponieważ dane były w formacie arkusza kalkulacyjnego Excel, skorzystałem z możliwości prostych funkcji zaimplementowanych w programie. Utworzyłem spójne porcje danych dla różnych tabel z zachowaniem warunków integralności.

Dane dla każdej tabeli zaimportowałem dane testowe korzystając z narzędzia dostarczonego razem z MS SQL Server – *Import and Export Data*.

Niedoskonałości/błędy aplikacji

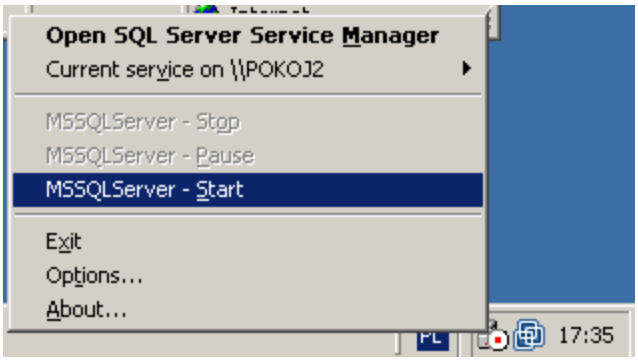
- Aplikacja ma ograniczoną funkcjonalność w takich aspektach jak zarządzanie przedmiotami i relacjami przedmiot-klasa-nauczyciel. Zostały utworzone formularz pozwalające na prostą edycje wszystkich powiązań. Możliwości dostępne przez te formularz są podobne do tych, których funkcjonalność została w pełni zaimplementowana, jednak sposób dostępu do wszystkich możliwości jest bardziej uciazliwy
- Podczas przechodzenia pomiędzy rekordami korzystając z selektora rekordów, nie jest wywoływana procedura aktualizacji pola obliczanego przez aplikacje – rok_klasy. W takim przypadku należy kliknąć na to pole aby zaktualizować wskazanie
- Komponent kalendarz podczas zmiany daty nie zawsze wywołuje zdarzenie onUpdate – w związku z tym czasami nie są poprawnie wyświetlane raporty z przedziałami czasu.

Instrukcja użytkownika

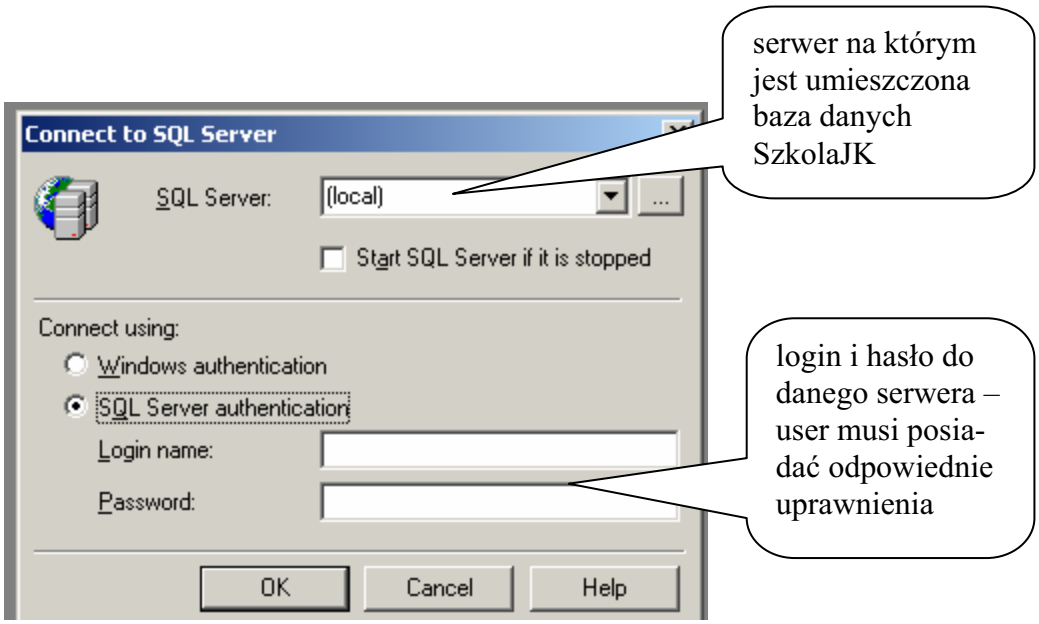
1. Instalacja serwera

- Instalację serwera MS SQL Server należy przeprowadzić zgodnie z instrukcjami dostarczonymi przez producenta
- Utworzenie bazy danych Szkola_JK najprościej jest zrobić przez uruchomienie **skryptu Szkola_JK.sql.** Na serwerze pojawia się pusta baza danych o nazwie Szkola_JK
- Aby utworzyć bazę danych trzeba mieć **odpowiednie uprawnienia** na danym serwerze SQL (uprawnienia pozwalające na tworzenie nowych baz)
- Jeśli istnieje baza danych **o tej samej nazwie** to zostanie ona **usunięta**! bez ostrzeżenia. Jeśli w bazie istniały dane zostaną one utracone! Zachowanie takie występuje pod warunkiem, że użytkownik ma prawo do usuwania baz danych na danym serwerze.
- Na serwerze jest **wymagana 20 MB** na dane w bazie danych oraz **5 MB** na plik dziennika zmian
- Baza danych zostanie utworzona w folderze **C:\Szkola_JK** (dwa pliki z rozszerzeniem mdf i ldf).

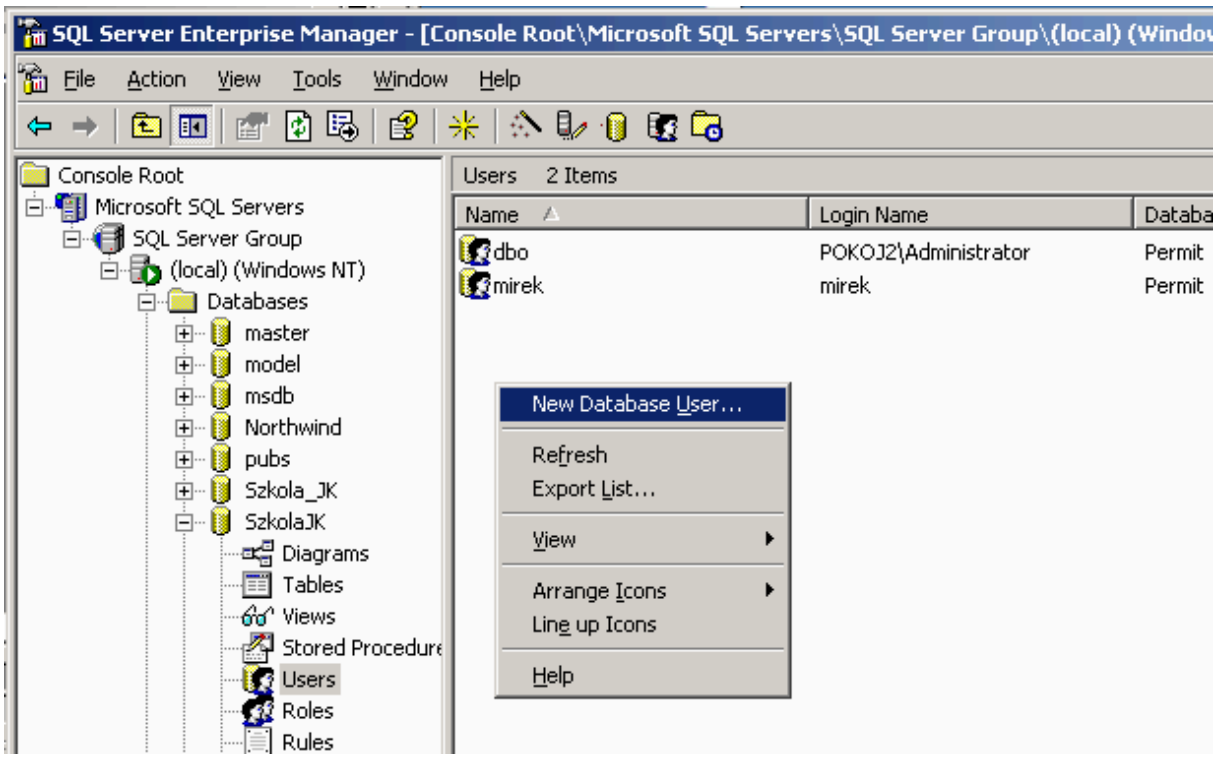
- W zależności od domyślnych ustawień konieczne może być **uruchomienie usługi** MS SQL Sever:



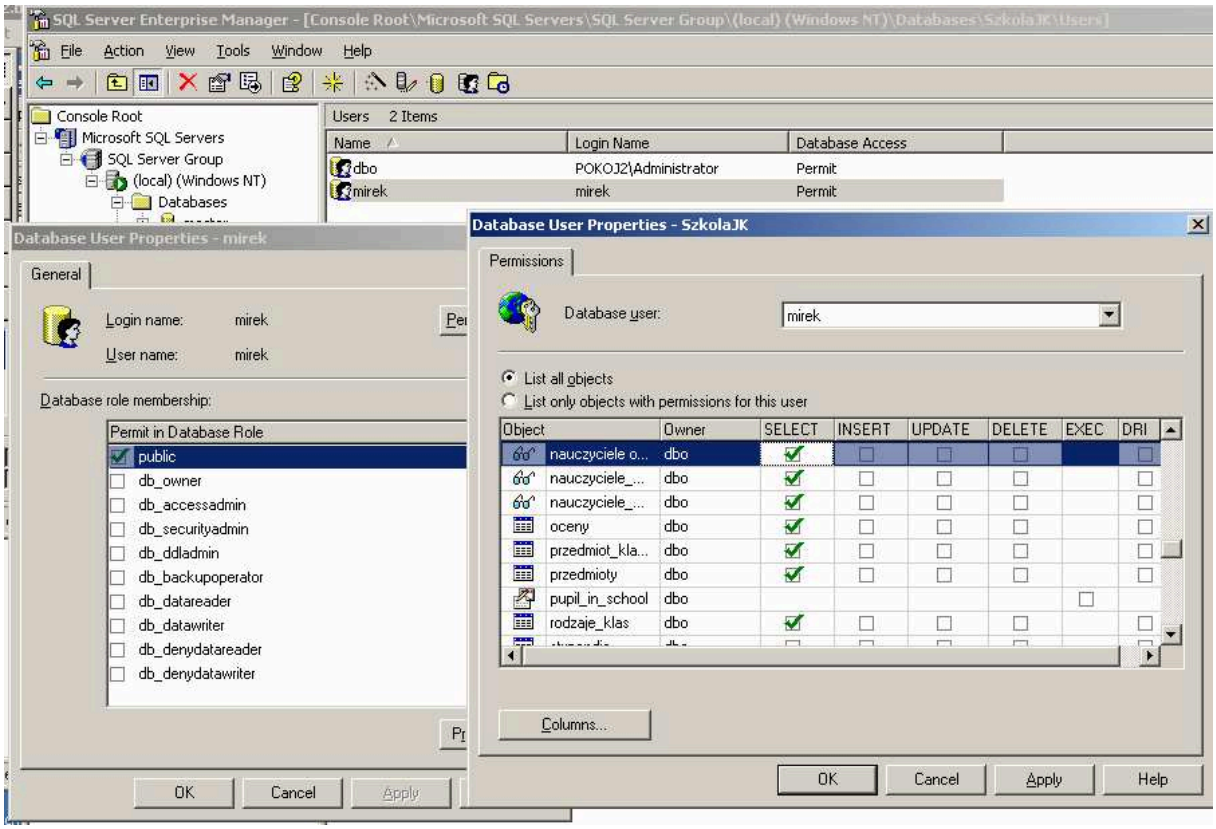
- **Dostęp do bazy** najprościej jest uzyskać korzystając z ***Enterprise Manager***. Przed rozpoczęciem operacji na bazie danych konieczne jest **zalogowanie** się z odpowiednimi uprawnieniami:



- Podstawową funkcjonalności z której będzie korzystał użytkownik końcowy jest **dodawanie użytkowników:**



- Przy dodawaniu użytkowników należy określić **uprawnienia** – zakres czynności dozwolonych do wykonania dla danego użytkownika


SQL Server Enterprise Manager - [Console Root\Microsoft SQL Servers\SQL Server Group\(local) (Windows NT)\Databases\SzkolaJK\Users]
File Action View Tools Window Help
Console Root
Microsoft SQL Servers
SQL Server Group
(local) (Windows NT)
Databases
Users 2 Items
Name
Login Name
Database Access
dbo
POKOJ2\Administrator
Permit
mirek
mirek
Permit
Database User Properties - mirek
General
Login name: mirek
User name: mirek
Database role membership:
Permit in Database Role
public
db_owner
db_accessadmin
db_securityadmin
db_ddladmin
db_backupoperator
db_datareader
db_datawriter
db_denydatareader
db_denydatawriter
OK
Cancel
Apply
Database User Properties - SzkolaJK
Permissions
Database user: mirek
List all objects
List only objects with permissions for this user
Object
Owner
SELECT
INSERT
UPDATE
DELETE
EXEC
DRI
nauczyciele o...
dbo
nauczyciele_...
dbo
nauczyciele_...
dbo
oceny
dbo
przedmiot_kla...
dbo
przedmioty
dbo
pupil_in_school
dbo
rodzaje_klas
dbo
Columns...
OK
Cancel
Apply
Help

```
ERROR: ioerror
OFFENDING COMMAND: flushfile

STACK:

-filestream-
-filestream-
-filestream-
-filestream-
-savelevel-
```