Instytut Sterowania i Systemów Informatycznych
Uniwersytet Zielonogórski

Bazy Danych

Ćwiczenie 4: Zapoznanie się z wybranym programem wspomagającym projektowanie relacyjnych baz danych

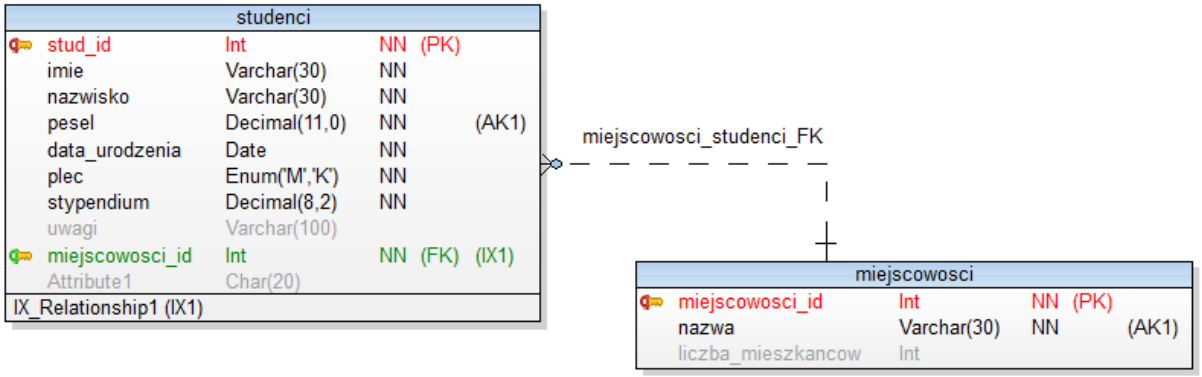
opracował: dr hab. inż. Artur Gramacki (a.gramacki@issi.uz.zgora.pl)

1. Uwagi wstępne

   W tym ćwiczeniu zapoznasz się z wybranym programem wspomagającym projektowanie relacyjnych baz danych. Program ten to *Toad Data Modeler* (w skrócie: TDM) w wersji 7.1, licencja 30-dniowa. Po tym okresie pewne funkcjonalności zostają wyłączone, jednak to, co nadal jest aktywne w pełni spełnia nasze potrzeby.

   Innym godnym polecenia programem (całkowicie darmowym) jest *MySQL Workbench* (*https://www.mysql.com/products/workbench*), który posiada bardzo rozbudowaną funkcjonalność, która może zaspokoić potrzeby nawet najbardziej zaawansowanego użytkownika bazy danych MySQL. Ten program jest całkowicie darmowy i nie ma limitów czasowych.

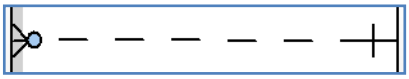
2. Na początek należy utworzyć bardzo prostą bazę danych, składającą się z dwóch tabel połączonych ze sobą kluczem obcym. Na poniższym rysunku pokazano tą strukturę.



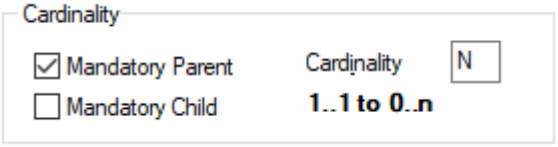
   Uwagi dotyczące modelu:

   - para *klucz główny* - *klucz obcy* to najbardziej podstawowy i najczęściej stosowany w praktyce sposób łączenia ze sobą tabel relacyjnych,
   - klucze główne są stworzone z opcją *AUTO_INCREMENT*, która pozwala na automatyczne generowanie unikalnych wartości dla tych kluczy,
   - większość kolumn ma ograniczenie *not null*, w praktyce nie zawsze tak musi być,
   - kolumna *pesel* ma założone ograniczenie *unique*, na rysunku jest to oznaczone symbolem AK1 (od *alternate key*),

- kolumna *plec* jest typu *enum,* dopuszczalne wartości to *M* oraz *K*,
- kolumna *stypendium* ma zdefiniowaną wartość domyślną (liczba 0),
- klucz obcy jest typu *NULL*, co oznacza, że rejestrując dane studentów nie musimy obowiązkowo wskazywać miejscowości. Na tej kolumnie automatycznie zakładany jest indeks (oznaczony skrótem IX1),
- *PESEL* jest zawsze liczbą 11. cyfrową, stąd taki a nie inny typ danych,
- kolumna *stypendium* ma typ *decimal(8,2)*, co oznacza, że można tam wpisać liczbę mającą 6 cyfr przed kropką dziesiętną oraz 2. cyfry po kropce dziesiętnej (w sumie 8 cyfr).
- Pojawiające się na rysunku symbole kółka oraz małej pionowej kreseczki



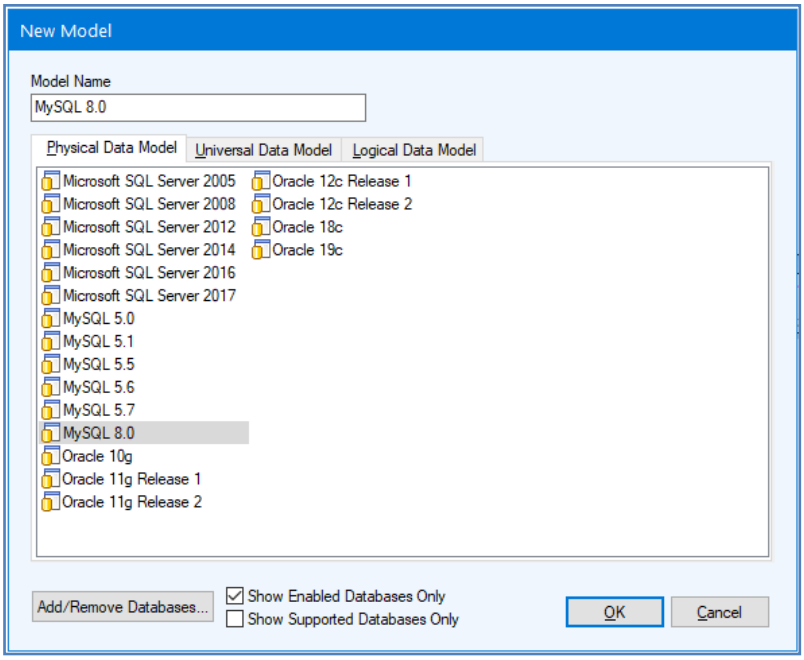
oznaczają tzw. kardynalność (nag. cardinality). Wartości te można dopasowywać do indywidualnych wymagań edytując daną relację (prawy klawisz na kresce i potem Edit).



Jest to liczba rekordów tabeli nadrzędnej powiązanych z rekordami tabeli podrzędnej. Na końcu instrukcji zamieszczono szczegółowy omówienie kwestii, jak należy interpretować te symbole (rysunki skopiowano z dokumentacji TDM).

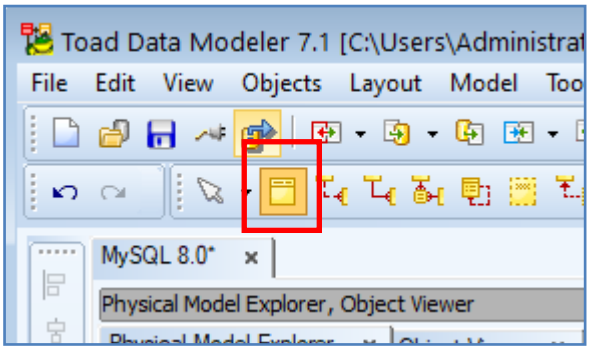
3. Szczegóły tworzenia modelu

W programie TDM utworzyć nowy model. Z dostępnej listy wybrać MySQL 8[1].

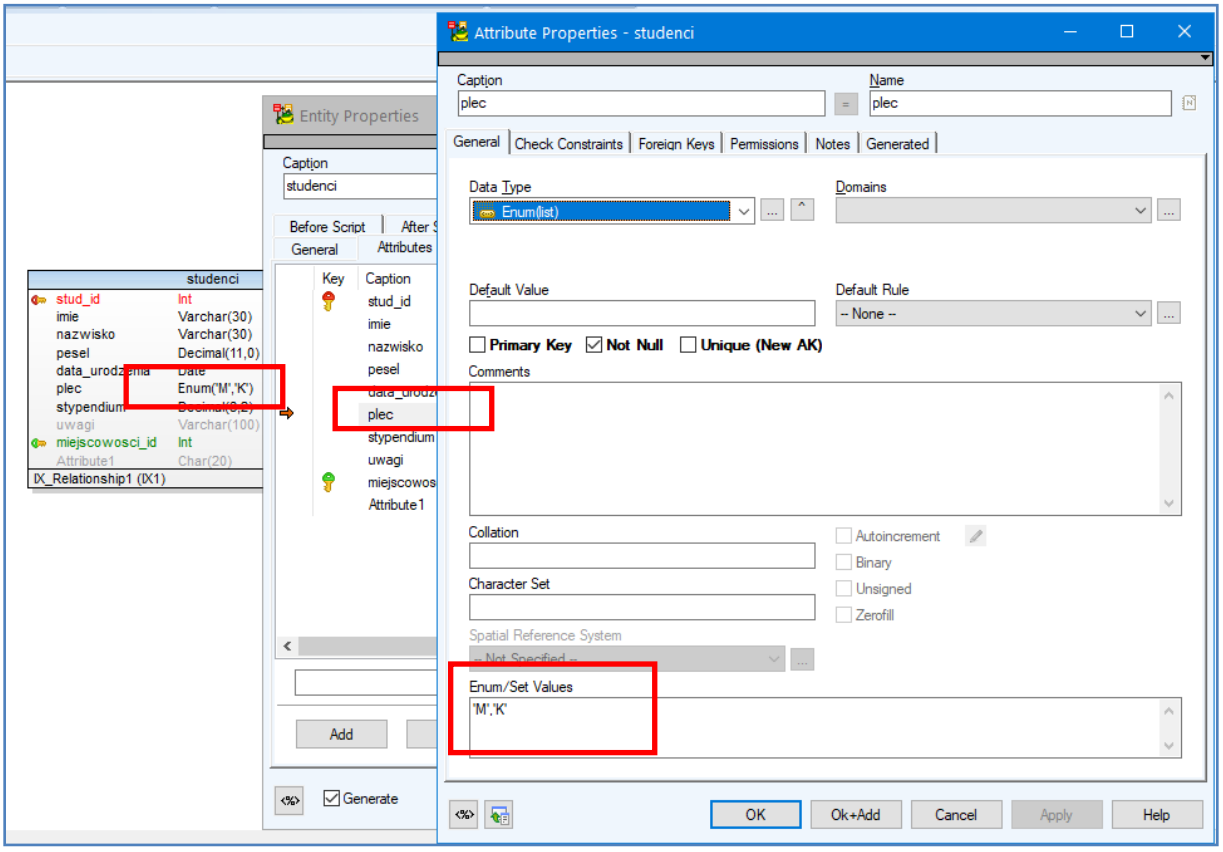


[1] W zasadzie można by wybrać każdą inną wersję MySQL-a, gdyż bazę tą poznajemy w bardzo podstawowym zakresie i dlatego też nie ma dla nas większego znaczenia, czym poszczególne wersje różnią się między sobą (bo różnią się stosunkowo niewiele).

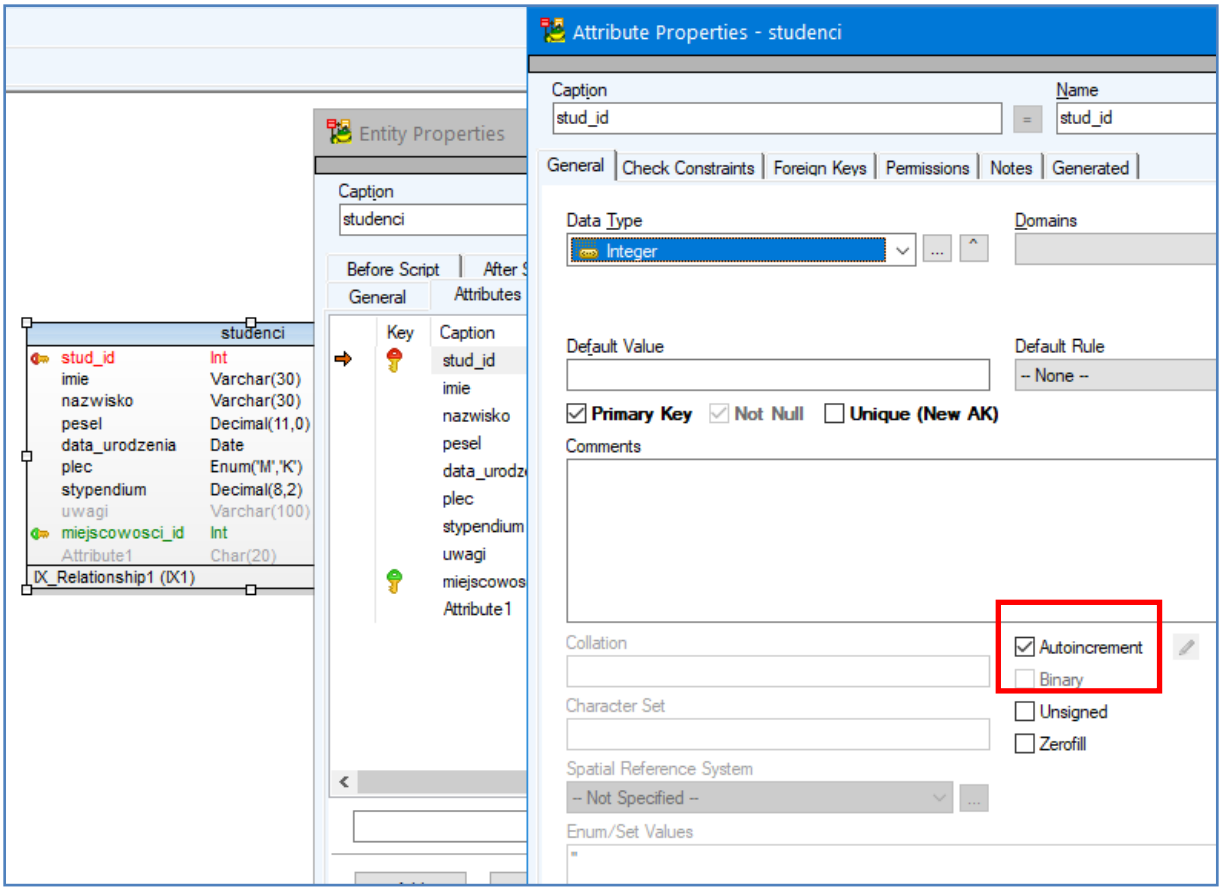
Wstawić dwie puste tabele.



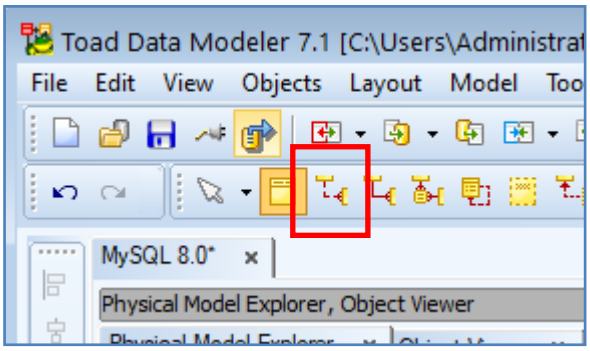
Kliknąć na każdą tabelę i rozpocząć definiowanie poszczególnych tabel. Większość czynności wykonuje się tutaj intuicyjnie. Definiując ograniczenie dla kolumny *plec* skorzystać z podpowiedzi pokazanej na rysunku niżej.



W czasie tworzenia kluczy obcych zaznaczyć odpowiednią opcję pozwalającą wykorzystać istniejącą w MySQL funkcjonalność *AUTO_INCREMENT* (automatyczne generowanie unikalnych wartości dla klucza głównego w czasie wstawiania nowych rekordów do tabeli). Na rysunku poniżej pokazano szczegóły.



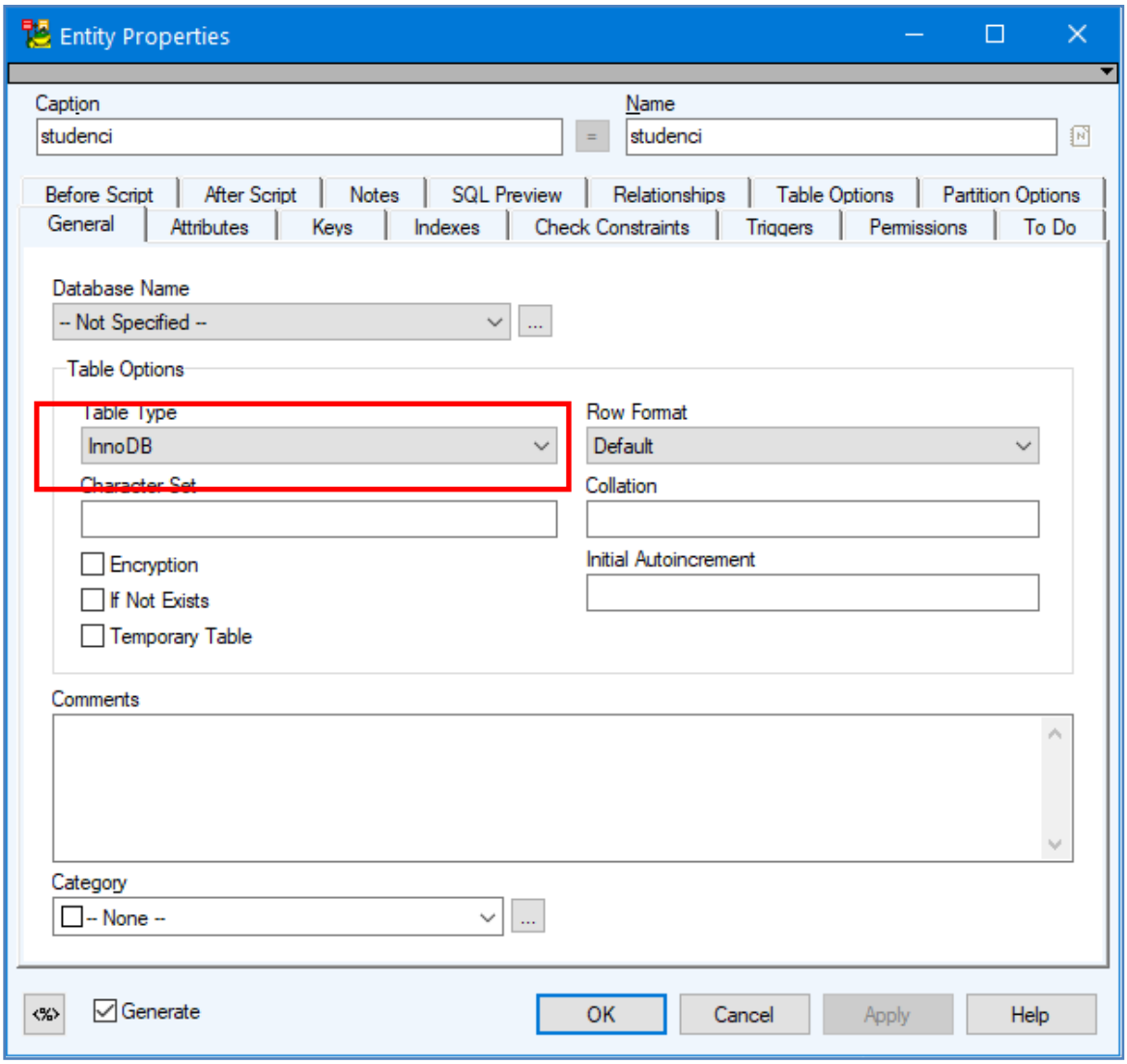
W tabeli nie tworzyć samodzielnie kolumny *miejscowosc_id*. Zostanie ona utworzona automatycznie, gdy stworzymy relację między tabelami. Będzie to tzw. relacja *nieidentyfikująca*. Istnieją też relacje *identyfikujące*, przykład ich wykorzystania będzie podany niżej. Należy z paska narzędziowego wybrać pozycję zaznaczoną czerwonym prostokątem i przeciągnąć ją od tabeli *miejscowości* do tabeli *studenci*.



Po utworzeniu relacji musimy zmodyfikować ręcznie kolumnę *miejscowosc_id* i usunąć ograniczenie *NOT NULL*. Ponadto musimy zdefiniować ręcznie nazwę ograniczenia (*miejscowosci_studenci_FK*).

Należy pamiętać, aby dla każdej tabeli ustawić tzw. mechanizm składowania na *InnoDB*. Wówczas tworzone klucze obce będą rzeczywiście działały Inny, popularny wiele lat temu ale ciągle jeszcze spotykany w różnych projektach, mechanizm *MyISAM* nie obsługuje kluczy obcych. W obecnych czasach ich brak należy uznać za niedopuszczalny. W ramach samodzielnego studiowania przedmiotu doczytaj w dostępnej literaturze, co to są mechanizmy składowania,

jakie są dostępne w poszczególnych wersjach serwera MySQL i czym się od siebie różnią (oprócz wspomnianych wyżej mechanizmów, MySQL oferuje jeszcze kilka innych, rzadziej używanych w praktyce, choć mających ciekawe właściwości).



Po zdefiniowaniu wszystkich elementów naszego modelu możemy przejść do najważniejszej „umiejętności” programu TDM a mianowicie do automatycznego wygenerowania kompletnego i całkowicie zgodnego ze standardem SQL (oferowanym przez serwer MySQL) skryptu tworzącego model. Wchodzimy więc do meny *Model -> Generate DDL Script -> Run*.

Zwróćmy uwagę, że domyślnie TDM używa odwrotnych apostrofów (ang. *backtick*) do otaczania nimi wszystkich identyfikatorów. Można z nich zrezygnować, jeżeli nie planujemy w naszej bazie używać „dziwnych” nazw dla obiektów (np. słów zarezerwowanych w języku SQL, w praktyce raczej powinniśmy unikać takich sytuacji, zysk jest niewielki, a mogą pojawić się nieoczekiwane kłopoty). Zapis z użyciem odwrotnych apostrofów jest nieco uciążliwy w praktyce i można rozważyć ich nieużywanie (ale trzeba to zrobić świadomie).

Po wciśnięciu guzika *Generate*, TDM rozpocznie najpierw weryfikację formalną modelu, gdy są jakieś błędy lub ostrzeżenia wygeneruje stosowne komunikaty. Następnie utworzony zostanie wynikowy skrypt, który pokazujemy poniżej.

```
/*
Created: 26.03.2020
Modified: 30.03.2020
Model: MySQL 8.0
Database: MySQL 8.0
*/

-- Drop foreign keys (relationships) section -------------------------

ALTER TABLE `studenci` DROP FOREIGN KEY `miejscowosci_studenci_FK`
;

-- Drop tables section -----------------------------------------------

DROP TABLE IF EXISTS `miejscowosci`
;
DROP TABLE IF EXISTS `studenci`
;
-- Create tables section ---------------------------------------------

-- Table studenci

CREATE TABLE `studenci`
(
  `stud_id` Int NOT NULL AUTO_INCREMENT,
  `imie` Varchar(30) NOT NULL,
  `nazwisko` Varchar(30) NOT NULL,
  `pesel` Decimal(11,0) NOT NULL,
  `data_urodzenia` Date NOT NULL,
  `plec` Enum('M','K') NOT NULL,
  `stypendium` Decimal(8,2) NOT NULL DEFAULT 0,
  `uwagi` Varchar(100),
  `miejscowosci_id` Int NOT NULL,
  `Attribute1` Char(20),
  PRIMARY KEY (`stud_id`)
) ENGINE = InnoDB
;

CREATE INDEX `IX_Relationship1` ON `studenci` (`miejscowosci_id`)
;

ALTER TABLE `studenci` ADD UNIQUE `pesel` (`pesel`)
;

-- Table miejscowosci

CREATE TABLE `miejscowosci`
(
  `miejscowosci_id` Int NOT NULL AUTO_INCREMENT,
  `nazwa` Varchar(30) NOT NULL,
  `liczba_mieszkancow` Int,
  PRIMARY KEY (`miejscowosci_id`)
) ENGINE = InnoDB
;

ALTER TABLE `miejscowosci` ADD UNIQUE `nazwa` (`nazwa`)
;

-- Create foreign keys (relationships) section -----------------------
----------------

ALTER TABLE `studenci` ADD CONSTRAINT `miejscowosci_studenci_FK` FOREIGN KEY
(`miejscowosci_id`) REFERENCES `miejscowosci` (`miejscowosci_id`) ON DELETE
RESTRICT ON UPDATE RESTRICT
;
```

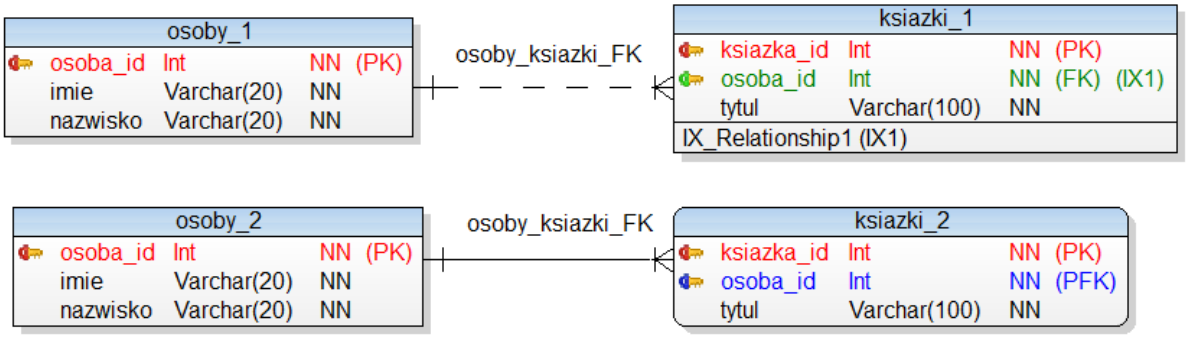
Skrypt ten można teraz uruchomić w konsoli MySQL. Zostaną utworzone obie tabele oraz relacja między tabelami. Zwróćmy uwagę też na jeden drobny błąd w działaniu programu TDM. Otóż na początku skryptu są dwa polecenia kasowania tabel. Są one niestety w niewłaściwej kolejności, gdyż jako pierwszą trzeba wykasować tabelę *studenci* (dlaczego?) a dopiero potem tabelę *miejscowości*. Na szczęścia polecenia kasowania tabel poprzedzone są poleceniem wykasowania klucza obcego i dzięki temu można potem wykasować tabele w dowolnej kolejności (dlaczego?).

```
mysql> source db_lab4.sql
Query OK, 0 rows affected (0.09 sec)

Query OK, 0 rows affected (0.12 sec)

Query OK, 0 rows affected (0.56 sec)

Query OK, 0 rows affected (0.39 sec)

Query OK, 0 rows affected (0.86 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> show tables;
+---------------+
| Tables_in_lab |
+---------------+
| miejscowosci  |
| studenci      |
+---------------+
2 rows in set (0.01 sec)
```

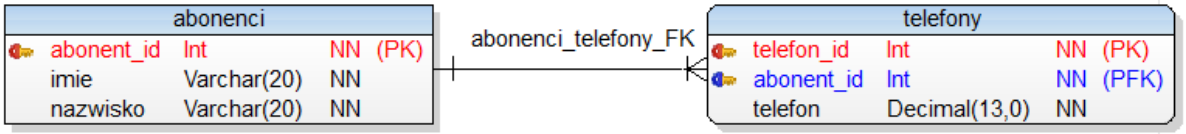
4. Utworzyć dwa podobne do siebie modele, w jednym będziemy mieli zdefiniowaną relację nieidentyfikującą a w drugim identyfikującą. Ta pierwsza jest rysowana linią przerywaną, druga ciągłą. Ponadto raz tabela ma rogi zaokrąglone a raz ostre. Relacja identyfikująca to taka, gdzie kolumna (kolumny) będąca kluczem obcym jest też częścią klucza głównego. Oznaczane jest to symbolem PFK. Różnica między obu typami relacji jest dość subtelna ale też i zasadnicza zarazem.

   W przypadku relacji nieidentyfikującej nasz prosty model należy rozumieć w taki sposób, że rejestrujemy książki, które mają jakiegoś właściciela a ten właściciel może mieć wiele różnych książek. Książka może także istnieć bez właściciela (gdy usuniemy ograniczenie *NOT NULL* z kolumny będącej kluczem obcym) lub ten właściciel może się zmieniać.

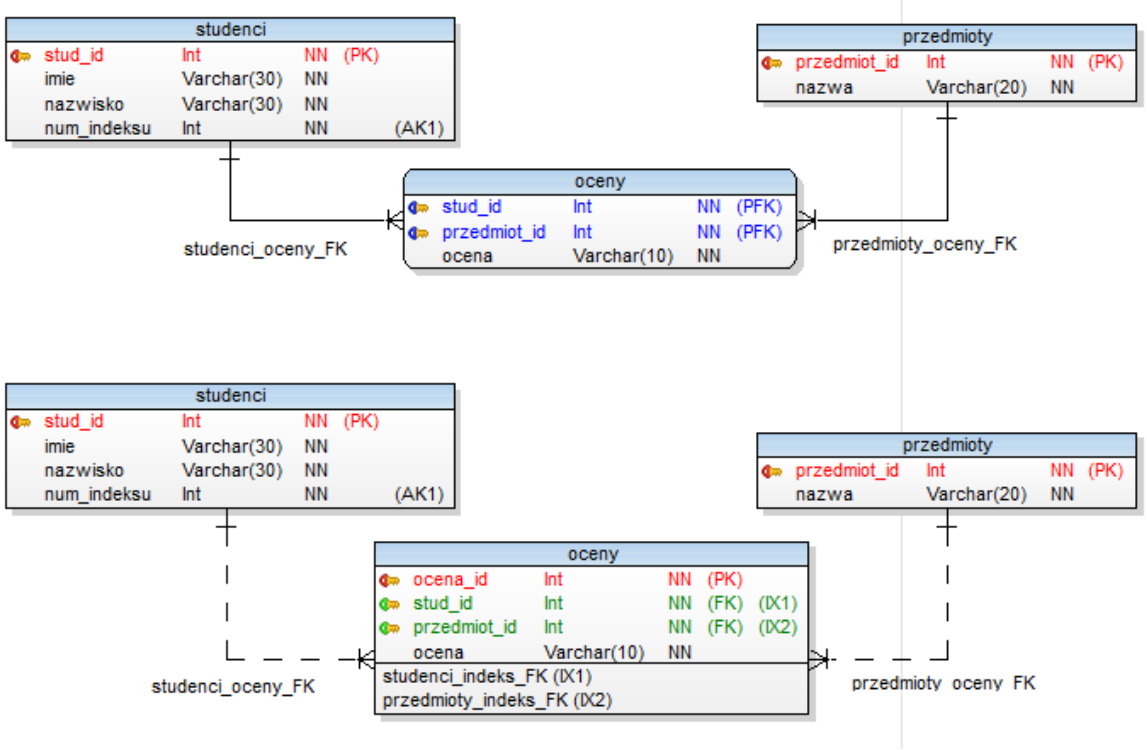
   W przypadku relacji identyfikującej nieco inaczej interpretujemy nasz model. Tym razem w tabeli osoby będą pojawiać się autorzy książek. Każdy autor może oczywiście napisać więcej niż jedną książkę. Książka nie zmienia swojego autora, ktoś ją przecież napisał i to jest już fakt historyczny. Książka bez autora nie istnieje, więc zabraniamy utworzenia takiego przypadku.



Inny przypadek, gdzie ma zastosowanie relacja identyfikująca pokazano niżej. Tym razem chodzi o sytuację, gdy numer telefonu jest „przywiązany" do abonenta. Zakładamy, że nie istnieją numery telefonów bez przypisanych do nich konkretnych osób. Jeżeli z jakiegoś powodu firma telekomunikacyjna chce przydzielić istniejący już numer innemu abonentowi, musi zmodyfikować też kolumnę *abonent_id* i tym samym przypisać zwolniony z jakiegoś powodu numer telefonu innej osobie.



5. Utworzyć prosty model relacyjny z relacją typu N:M. W modelu tym będziemy chcieli rejestrować dane studentów oraz oceny jakie uzyskują z poszczególnych przedmiotów. W jednym modelu wykorzystano relacje identyfikujące a w drugim nieidentyfikujące. Samodzielnie przeanalizuj oba modele i odpowiedź na pytanie, który model będzie miał zastosowanie, w jakich konkretnych sytuacjach. Podaj konkretne zastosowania.

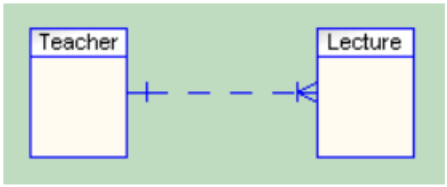


6. Cardinality

Parent: **Mandatory**

Child: **Mandatory**

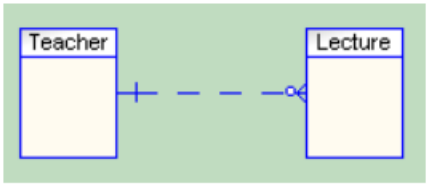
A teacher MUST have a lecture (the record related to lecture is mandatory), a lecture MUST be attached to a teacher (the record related to teacher is also mandatory. Teacher is Mandatory.)



Parent: **Mandatory**

Child: **Optional**

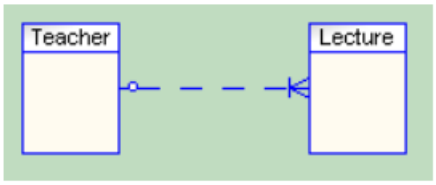
A teacher MAY have a lecture (the record related to lecture is Optional), a lecture MUST be attached to a teacher (Teacher is Mandatory.)



Parent: **Optional**

Child: **Mandatory**

A teacher MUST have a lecture (lecture is Mandatory), a lecture MAY be attached to a teacher (teacher is Optional).



Parent: **Optional**

Child: **Optional**

A teacher MAY have a lecture (lecture is Optional), a lecture MAY be attached to a teacher (teacher is Optional).