

# Informatyka II

## EE-DI, WEiI PRz

Instrukcja ta jest dostępna w formie źródłowej pod adresem [http://git.dms-serwis.com.pl/henrietta/inf2\\_eedi](http://git.dms-serwis.com.pl/henrietta/inf2_eedi).

Pod tym adresem będą pojawiać się również instrukcje do kolejnych laboratoriów.

## Laboratorium 4

*“Quidquid latine dictum, altum videtur”*

---

Z tego laboratorium przygotowujesz sprawozdanie. Przygotowujesz je na zajęciach, a przy ich zakończeniu wysyłasz na adres podany na końcu tej instrukcji. Instrukcja dla wszystkich jest taka sama.

Pracę wykonujesz w grupach dwuosobowych. Dobierz sobie współpracownika. Na liście obecności zaznacz numer grupy (dwie osoby - jeden numer grupy).

Rzeczy oznaczone tak, jak poniżej, dotyczą tego, co masz zawrzeć w sprawozdaniu. Na przykład:

Zapisz swoje imię, nazwisko, adres e-mail, kierunek i rok studiów oraz grupę laboratoryjną i numer albumu.

Wykonujesz taki wariant zadania, jaki jest numer grupy. Na przykład wariant, który obecnie czytasz, ma numer 6.

Podaj również numer zajęć laboratoryjnych (nr 4).

Mogą być to też pytania, na które w sprawozdaniu udzielisz odpowiedzi. Możesz pomagać sobie wyszukiwarką internetową, oraz zabrać głos w dyskusji, jeśli się jakaś wywiąże.

## Bazy relacyjne

Relacyjne bazy danych (ang. *relational database management systems*, RDBMS, może bardziej poprawnie *system zarządzania relacyjną bazą danych*), to pewien

pomysł na przechowywanie danych. Pomysł ten polega na rozbiciu naszego świata w szereg takich *obiektów*, *bytów* (w żargonie bazodanowców **encji**, od ang. *entity* czyli w zasadzie *byt*), które będzie można wygodnie umieścić w *tabelach*. Obiekt taki charakteryzuje się tym, że ma swoją pewną *tożsamość*, po której to tożsamości będziemy go potem identyfikować i po tejże się do niego odwoływać.

**Uwaga moja osobista:** nie jestem wielkim fanem słowa *encja*. Powstało ono jako słowo-kalka dla angielskiego *entity*, o którym to każdy słownik powie że jest to *istnienie* tudzież *byt*. Dużo lepiej odzwierciedla to *sens* tego słowa w kontekście baz danych niż jakaś *encja*. Zakorzeniło się ono jednak w *polskiej* gwarze bazodanowej na tyle, że czasem nie używając tego słowa nie zostaniemy zrozumieni, tak więc należy zachować rozwagę. *Opinia skonsultowana z tłumaczem lub anglistą*. Anglista pozdrawia.

Na przykład: tworząc bazę danych do celów poboru podatków w Jaśnie Oświeconej III Rzplitej, potrzebujemy listy podatników. Żeby uprościć nieco sprawę, założmy że rzeczona baza dotyczyć będzie tylko osób fizycznych (słowem będzie bazą PIT, od ang. *personal income tax*, podatek dochodowy od osób fizycznych), czyli *obywateli*.

Jaka cecha obywatela stanowi jego dobrą tożsamość? Imię i nazwisko niezbyt - Janów Kowalskich mamy wszak ilość znaczną. Może imię, nazwisko i data urodzenia? Też niedobrze, wszak możemy wyobrazić sobie sytuację, w której dwóch Janów Kowalskich urodziło się w tym samym dniu. Jest to dużo bardziej prawdopodobne, niż się wam wydaje. Takie *tożsamości*, lub lepiej **klucze podstawowe** (ang. *primary keys*) mogą również składać się z kilku cech (lub lepiej **pól**, ang. *fields*).

Wydaje się po dłuższych deliberacjach że dobrym kluczem podstawowym będzie tu PESEL obywatela. Ryzyko pomyłki z innym obywatelem jest minimalne (chyba że aktualnie jesteśmy komornikiem). Dzięki takiej *tożsamości* możemy potem kazać bazie chociażby *zmienić nazwisko osobie legitymującej się numerem PESEL 95101810106* bez ryzyka pomyłki, w razie gdyby zmieniła nazwisko z racji wyjścia za mąż. Klucz podstawowy przywiązuje nasz wiersz w bazie do konkretnego obiektu istniejącego w rzeczywistości.

Takie rozwiązanie od kilku lat już zastosowano, dzięki czemu ilość druków NIP-7 w obrocie znacznie spadła, a Panowie zapewne do swojego rocznego rozliczenia podatkowego nie wpisywali żadnych NIP-ów, bo wystarczył PESEL.

Pominę tu pytanie jaki klucz podstawowy oraz kolumny należy zastosować do konkretnego, rocznego rozliczenia podatkowego, gdyż ze względu na mnogość możliwych sposobów rozliczeń przewidzianych stosowną ustawą oraz wzorów formularzy PIT stanowi to Bardzo Dobre Pytanie. **Zwalniam z zaliczenia jak ktoś ma dobry i wyczerpujący pomysł**, przy czym od razu ostrzegam że problem jest trudny i bardziej prawno-księgowy niż informatyczny.

Tak czy inaczej, wypiszmy kilka sensownych *pól* w takiej tabeli od obywateli:

- Imię
- Nazwisko
- Adres zameldowania: miasto
- Adres zameldowania: kod pocztowy
- Adres zameldowania: ulica i numer budynku oraz lokalu
- Adres korespondencyjny: miasto
- Adres korespondencyjny: kod pocztowy
- Adres korespondencyjny: ulica i numer budynku oraz lokalu
- PESEL
- Data urodzin
- Numer telefonu

I teraz parę kwestii nazewniczych. Odnosząc się do tego konkretnego przykładu:

- Nazwę pola, wraz z jego typem i przeznaczeniem w konkretnej tabeli nazywamy **kolumną**
- Reprezentację obywatela w tej tabeli nazywamy **wierszem**
- Konkretną cechę konkretnego obywatela nazywamy **polem**

Numeru telefonu obywatel posiadać nie musi. Każdy bieszczadzki zakapior ma prawo do obywatelstwa i obowiązku rozliczenia podatkowego, nawet bez posiadania telefonu. Przy definiowaniu kolumny zawsze musimy ustalić, czy podanie jej jest obowiązkowe. Pola objęte kluczem podstawowym zawsze są wymagalne (w żargonie **not null**), bo inna sytuacja niespecjalnie ma sens (jak zbierać informacje o obywatelu pozbawionym numeru PESEL?).

Od razu zauważyć możemy, że trzymanie dwóch *adresów*. nie jest takim dobrym pomysłem. Analityczny umysł zauważyć może, że adres jest pewnym *bytem samym w sobie*. Możemy więc stworzyć osobną tabelę do przechowywania adresów.

Powstaje z kolei bardzo dobre pytanie, co stanowi *tożsamość* danego adresu. Czasem, gdy nie mamy dobrego pomysłu na tożsamość możemy zażyczyć sobie, aby baza danych stworzyła osobną tożsamość dla danego wiersza. Później, gdy będziemy dodawać rekord, baza poinformuje nas o tym identyfikatorze.

Musimy jednak uważać! Dla potrzeb bazy dwa identycznie wpisane adresy będą *różnymi*, ponieważ będą miały różne identyfikatory (chyba że nie poinformujemy o tym bazy). Czasem jednak możemy to przecierpieć, a jeszcze częściej po prostu nie ma to najmniejszego znaczenia.

Spróbujmy więc odseparować *byt* obywatela od adresu. Otrzymamy zapewne coś takiego:

**Obywatel:**

- Imię

- Nazwisko
- Adres zameldowania - ID adresu
- Adres korespondencyjny - ID adresu
- PESEL (**klucz podstawowy**)
- Data urodzin
- Numer telefonu (**może być pusty**)

#### Adres:

- Identyfikator adresu (**klucz podstawowy, generowany automatycznie**)
- Miasto
- Kod pocztowy
- Ulica i numer lokalu/budynku

Można sprzeczać się, że taki podział nie miał najmniejszego sensu. To możliwe. W każdym razie to co zrobiliśmy teraz nazywa się normalizacją. W epoce baz danych NoSQL staje się ona coraz bardziej zagadnieniem jedynie akademickim, jednak mam wrażenie że będzie na egzaminie (aczkolwiek form normalnych uczylibym się jako pierwszych gdyby mi kazano).

Identyfikatory generowane przez bazę najczęściej są rosnącymi liczbami dodatkowymi i **nie powtarzają się**, nawet gdybyśmy skasowali wiersz. Ma to sens. W końcu gdybyśmy prowadzili policyjną kartotekę dowodów rzeczowych, a ktoś w piśmie wspomniał o dowodzie o określonym identyfikatorze, a w międzyczasie ten dowód został zniszczony ze względu na koniec sprawy, to teraz jego papier mógłby się odwoływać do *istniejącego dowodu w innej sprawie*, a nie po prostu do *niczego*. Byłoby to katastrofalne w skutkach, a błąd programisty mógłby kosztować kogoś pozbawienie wolności.

Istotne w kluczu podstawowy jest to, że w danej tabeli istnieć może tylko jeden rekord o tym kluczu. Nie jest możliwe wstawienie dwóch - mamy więc podstawowy system zapobiegający sytuacjom niemożliwym, na przykład istnieniu dwóch *różnych* osób o identycznym PESEL-u.

Kolumnę, która stanowi klucz podstawowy w innej tabeli - czyli odnosi się do innej tabeli - nazywamy **kluczem obcym** (ang. *foreign key*). W naszej tabeli *Obywatel* mamy dwa klucze obce - oba odnoszą się do tabeli *Adres*. Ponieważ wielu obywatelom przypisać można jeden adres, istnieje więc relacja **wiele-do-jednego** między tabelami *Obywatel* i *Adres*.

Po co stosuje się takie wyróżnienia? Otóż możemy bazę poinformować o istnieniu takiej relacji. Wtedy pilnować ona będzie spójności, czyli jeśli spróbujemy skasować adres, pod którym ktoś jednak zamieszkuje, baza odmówi wykonania takiego polecenia (i dobrze).

Opisując kolumny w tabeli musimy również podać typ tego, co będziemy tam przechowywać. Typy zależą już od konkretnego oprogramowania RDBMS z

którego korzystamy. Na WEiI PRz istnieje parcie na Oracle Database, tak więc tej typologii rekomendowałbym się nauczyć na egzamin.

Na przyszłych laboratoriach będziemy korzystać z PostgreSQL oraz jego typów. Ponieważ istnieje tabela konwersji typów między tymi bazami, radzę myśleć w typach Oracle, a w locie dopasowywać sobie to do PostgreSQL. O typach PostgreSQL można zapomnieć po odesłaniu ostatniej instrukcji.

Projektując bazę danych, dobrze jest ją sobie narysować. Służy do tego diagram związków encji lub inaczej ERD. Na egzaminie najpewniej będzie (jeśli będzie) jej wariant w postaci notacji kruczej stopki.

Użyj Google i naucz się rysować diagramy ERD w maksymalnie 15 minut. Narysuj diagram ERD dla tabel Obywatel i Adres.

Jeśli chcemy zamodelować relację “jedna encja posiada inne” najlepiej to zrobić właśnie na dwóch tabelach. Gdybyśmy chcieli stworzyć tabelę pt. “czy obywatel rozliczył podatek”, możemy zrobić tabelę *CzyRozliczyłPodatek*, która wyglądać będzie mniej więcej tak:

- PESEL obywatela
- Rok fiskalny
- Czy rozliczone?

Przy czym pola PESEL i rok fiskalny stanowiąc będą klucz podstawowy, zaś sam PESEL - klucz obcy. Jest to dobry wybór na tożsamość tego pola, bo przecież jedna osoba rozliczyć za dany rok może się tylko raz (albo tak myślimy). Jak tu widzimy, klucz podstawowy może się również składać z klucza obcego - bo czemu nie.

Wystąpi również relacja **jeden-do-wielu** w stosunku Obywatel - CzyRozliczyłPodatek, bo jedna osoba może (a nawet musi) się rozliczać wielokrotnie - raz z każdego roku.

Dorzuć teraz do wcześniej narysowanego diagramu tabelę *CzyRozliczyłPodatek*.  
Wynik radosnej twórczości umieść w sprawozdaniu.

## Zadanie laboratoryjne

Na dzisiejszym laboratorium przygotujemy schemat bazy danych do pewnego zastosowania.

Każda grupa ma inne zadanie. Twoje zadanie to przygotować schemat bazy komputerowego dziennika szkolnego. Baza ma śledzić klasy, uczniów, oceny i

przedmioty. Minimalnie użyjesz 4 tabel, choć jeśli jesteś w stanie znormalizować ten schemat bardziej, to będzie to mile widziane. Każda z tabel ma zawierać minimalnie 4 pól. Nie wolno tworzyć identyfikatorów automatycznie generowanych, jeśli użyć można istniejącego w rzeczywistości atrybutu (np. EAN).

Jeśli w tabeli chciałbyś zawrzeć więcej informacji, niż to wynika z zadania, to bardzo dobrze! W razie wątpliwości pytaj prowadzącego.

1. Zastanów się, co w zasadzie śledzisz. Gdybyś prowadził kartotekę papierową z tymi informacjami, to jak poukładałbyś szufladki?
2. Jakie *byty* śledzisz w swojej bazie danych?
3. Rozplanuj bazę danych, każdą tabelę z osobna i opisz je słowami.
4. Narysuj jej diagram ERD

Krótki opis zagadnienia umieść w bazie. Odpowiedzi na pytania 1 i 2 również.

Plan bazy i tabel umieść w sprawozdaniu wraz z diagramem ERD.

Diagram ERD koniecznie w postaci zdjęcia narysowanego odręcznie na papierze schematu.

Pamiętaj, aby schematem tabeli uniemożliwić sytuacje niemożliwe (np. dwie osoby mają rezerwację na identyczne miejsca w tym samym seansie kinowym).

**Zanim zakończysz pracę, skonsultuj swoje wyniki z prowadzącym!**

## Wyślij sprawozdanie

*Fast fertig!* Zapisz i wyślij swoje sprawozdanie. Jeśli masz problem z wklejeniem zdjęcia zrobionego z telefonu to wyślij sprawozdanie później. Jeśli nie masz takiego telefonu, wykorzystaj kolegę.

Wyślij je na adres [sprawozdania@henrietta.com.pl](mailto:sprawozdania@henrietta.com.pl).

W tytule umieść imię, nazwisko i numer zadania.

Załącz sprawozdanie, lub wklej je w treść maila.

Pamiętaj, uczestniczysz w zajęciach LABORATORYJNYCH, więc nie pisz "Projekt Adam Nowak".

Podejście takie będzie aktywnie penalizowane.

Twoim zadaniem domowym jest nauczanie się języka SQL. Nauczyć musisz się przynajmniej poleceń:

- SELECT
- INSERT
- UPDATE
- DELETE
- CREATE TABLE

Szczególną uwagę poświęć klauzulom WHERE oraz LEFT JOIN.

Ich znajomość będzie sprawdzona na przyszłych zajęciach, gdzie będziesz implementować swój schemat bazy w języku SQL oraz pisać do niej zapytania.

Teraz idź korzystać z dobrej/złej pogody, bo na weekendzie ma kropić.

*“Uczmy się kochać papier, tak szybko odchodzi”*

---

Copyright (c) 2017 Piotr Maślanka. Niektóre prawa zastrzeżone.

Kod źródłowy dostępny na [http://git.dms-serwis.com.pl/henrietta/inf2\\_eedi](http://git.dms-serwis.com.pl/henrietta/inf2_eedi).