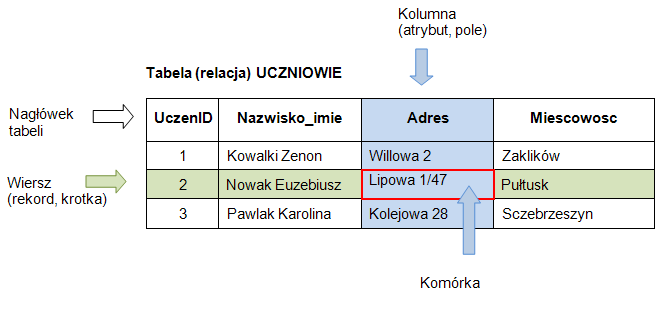
**Lekcja 2**

**Temat:** Relacyjny model danych

**Tabele:**

**Tabele baz danych** nazywane są również relacjami. Relacje zachodzą również między poszczególnymi tabelami.

**Tabele są podstawowymi obiektami baz danych** – przechowują dane. Są tym na co wskazuje nazwa.



* **Kolumny (inaczej atrybuty, pola)** - posiadają wyróżniające je nazwy (pisane bez "polskich ogonków" i spacji) i zawierają różny rodzaj danych, przy czym **w jednej kolumnie znajdują się dane jednego typu**. Możemy się domyśleć, że typ danych w kolumnie UczenID, to liczby całkowite, natomiast w pozostałe kolumny zawierają ciągi (inaczej łańcuchy, stringi) znaków.
* **Wiersze (inaczej krotki, rekordy)** - jeden wiersz tabeli odpowiada jednemu uczniowi. Proszę zwrócić uwagę, że **każdy z wierszy posiada takie same pola**.
* **Wartości** - Każdy wiersz jest zbiorem wartości, z których każda odpowiada odpowiedniej kolumnie. Wartości znajdują się w komórkach.

W **modelu relacyjnym** każda z tabel charakteryzuje się następującymi własnościami:

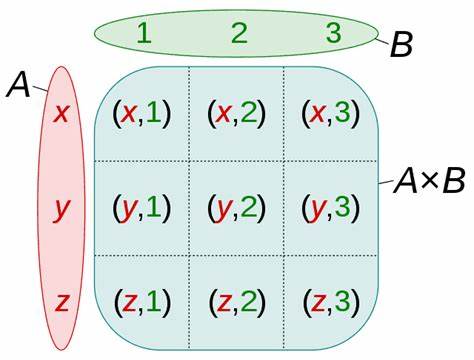
* każda tabela powinna mieć nazwę unikatową w całej relacyjnej bazie danych;
* każda kolumna tabeli powinna posiadać nazwę unikatową w danej tabeli;
* każda komórka, będąca przecięciem wiersza i kolumny, powinna zawierać wartość atomową (niepodzielną);
* wszystkie wartości danej kolumny muszą być określonego typu;
* w tabeli kolejność kolumn jak również kolejność wierszy nie ma znaczenia;
* każda tabela musi się składać z co najmniej z jednej kolumny i może mieć zero, jeden lub większą ilość wierszy;
* wiersze w tabeli nie mogą się powtarzać – muszą być unikatowe, co jest równoznaczne z tym, że muszą się różnić co najmniej jedną wartością;
* każda tabela powinna mieć tzw. **klucz**, a więc jedną lub kilka kolumn, które w sposób jednoznaczny identyfikują każdy wiersz w relacji. Przykładowo w relacji Uczniowie (Rysunek 2\_2\_1\_1) jest to kolumna UczniowieID.

**Klucze**

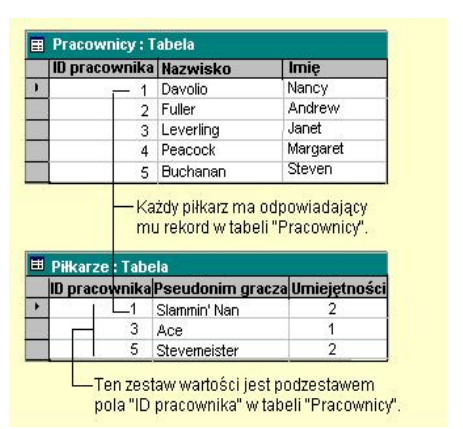
Każda tabela musi posiadać atrybut lub zbiór atrybutów jednoznacznie identyfikujący. Taki atrybut nosi nazwę **klucza głównego (podstawowego).**

* **Klucz główny musi spełniać następujące warunki:**
  + Musi być unikatowy
  + Powinien mieć niepodzielną wartość
  + Nie może przyjmować wartości NULL;
  + Wartość klucza nie może się zmieniać
* **Możemy rozróżnić następujące rodzaje kluczy:**
  + **Klucz prosty** – jednoelementowy złożony z jednej kolumny np. PESEL
  + **Klucz złożony** – kilkuelementowy złożony z kilku kolumn np. Imię, Nazwisko
  + **Klucz sztuczny** – klucz z którego korzysta się tylko w przypadku gdy z cech opisujących obiekt nie można wyodrębnić takiej, która spełnia warunki klucza głównego np. IDPojazdu
  + **Klucz obcy** – służy do łączenia dwóch tabel (np. A i B) za pomocą związków. Klucz pochodzący z obcej tabeli B (w której jest on kluczem głównym) używany do łączenia tej tabeli z tabelą A będzie dla tej tabeli kluczem obcym

**Relacje**



**Relacja jeden do jednego:**

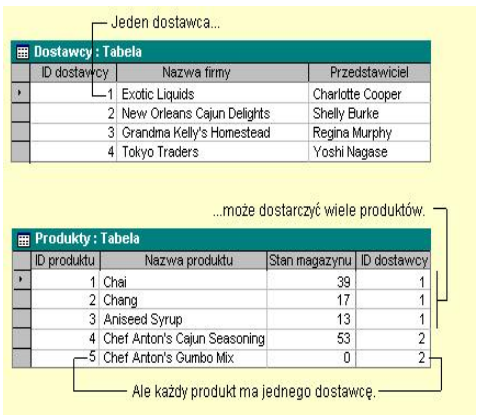
****

• Charakteryzuje się tym, że dla każdej instancji jednej z dwóch encji istnieje dokładnie jedna instancja drugiej encji pozostająca z nią w równoważnym związku.

• Np. czek i opłata (opłata jest realizowana za pomocą jednego czeku i za pomocą jednego czeku można zrealizować tylko jedną opłatę).

• Ten typ relacji spotyka się rzadko, ponieważ większość informacji powiązanych w ten sposób byłoby zawartych w jednej tabeli.

**Relacja jeden do wielu**

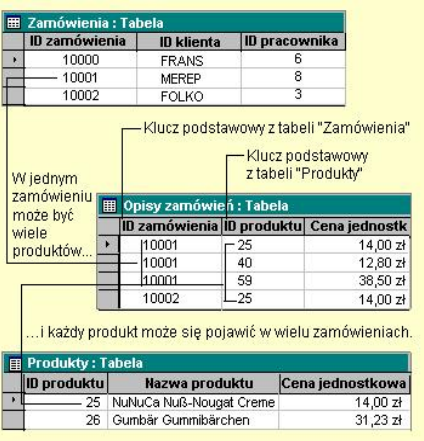
****

• Charakteryzuje się tym, że dla każdej instancji jednej encji istnieje wiele instancji drugiej encji pozostającej z nią w rozważanym związku.

• Relacja jeden-do-wielu jest realizowana poprzez utworzenie atrybutu w encji po stronie wiele aby umieścić w nim klucz encji znajdującej się po stronie jeden.

• Tak utworzony atrybut encji po stronie wiele nosi nazwę klucza obcego ponieważ jest on głównym kluczem w innej tabeli. Relacja jeden-do-wielu jest najbardziej powszechnym typem relacji.

**Relacja wiele do wiele**

****

• W relacji wiele-do-wielu, rekord w tabeli A może mieć wiele dopasowanych do niego rekordów z tabeli B i tak samo rekord w tabeli B może mieć wiele dopasowanych do niego rekordów z tabeli A.

• Jest to możliwe tylko przez zdefiniowanie trzeciej tabeli (nazywanej tabelą łącza), której klucz podstawowy składa się z dwóch pól - kluczy obcych z tabel A i B.

• Relacja wiele-do-wielu jest definiowana jako dwie relacje jeden-do-wielu z trzecią tabelą.

**Typy danych**

Typy liczbowe

W przypadku niektórych typów danych, możliwe jest określenie **maksymalnej liczby wyświetlanych znaków** - będziemy ją zaznaczać literą *M*. Ta liczba może być opcjonalna, wtedy znajduje się w nawiasach kwadratowych: [*M*]. Największa dopuszczalna wartość parametru *M* wynosi 255.

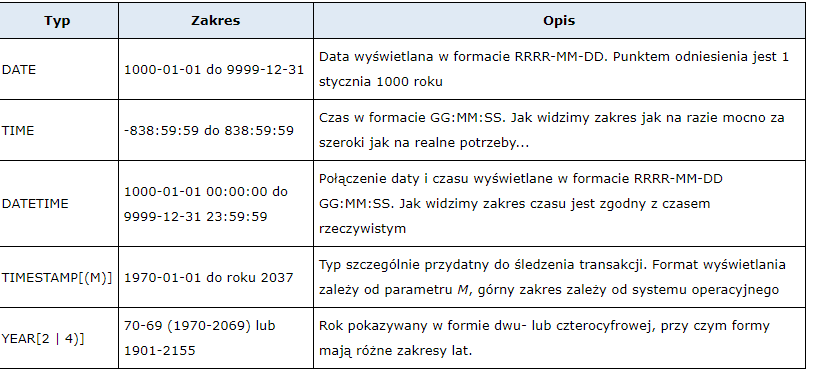
Możemy wyróżnić typy liczbowe:

* **liczby całkowite** - na przykład 2, 5, -2, 233, itp. Ten typ można zawęzić do liczb nieujemnych - **UNSIGNED**. W zestawieniu liczb całkowitych pokazanych w tabeli, zakres po słowie *lub* odpowiada zastosowaniu tego właśnie atrybutu;
* **liczby zmiennoprzecinkowe** - są to liczby rzeczywiste, przy czym jest możliwość zadeklarowania liczby cyfr po przecinku (a dokładnie po kropce, którą będziemy stosować zamiast przecinka). W tabelach poniżej, liczba ta jest oznaczona symbolem *D*. Maksymalnie parametr *D* może przyjąć wartość 30, przy czym nie może być większy od *M* - 2. Skąd wzięła się liczba 2? Stąd, że liczbę wszystkich znaków musimy pomniejszyć o znak kropki i przynajmniej jeden znak liczby całkowitej;

Do wszystkich typów, można zastosować parametr **ZEROFILL** powodujący wypełnienie zerami miejsc poprzedzających liczbę - np. 04. Tak zdefiniowana kolumna, automatycznie przyjmuje typ **UNSIGNED**.

# Typy daty i czasu

Każdy z pokazanych typów, pozwala na wpisanie danych w formie lczbowej lub łańcucha znaków. Typ TIMESTAMP, posiada właściwość automatycznego uzupełnienia o wartość aktualnej daty i czasu w przypadku, jeżeli pole pozostawimy puste.



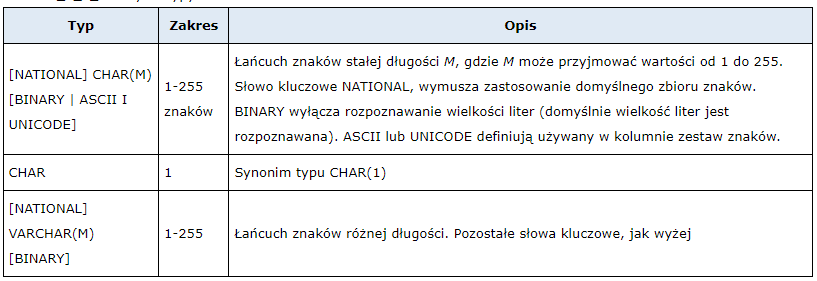
# Typy łańcuchowe

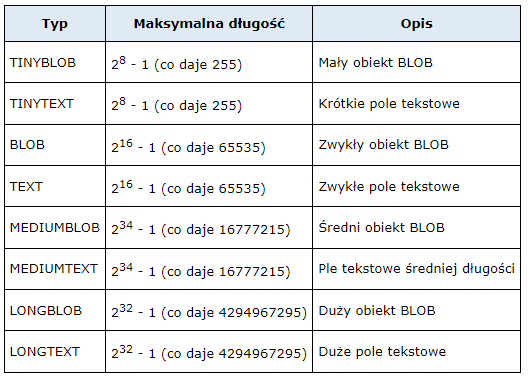
Można je podzielić na 3 grupy:

* klasyczne łańcuchy (stringi) znaków
  + CHAR - łańcuchy o stałej długości
  + VARCHAR - łańcuchy o zmiennej długości
* typy TEXT do przechowywania długich tekstów oraz BLOB dla dużych danych binarnych
* typy specjalne SET do określonego zbioru wartości oraz ENUM - typ wyliczeniowy, gdzie można wybrać jedną wartość spośród określonego zbioru wartości

## **Zwykłe typy łańcuchowe**

Stosowaliśmy już tworząc tabele bazy ksiegarnia\_internetowa, a następnie [**omawiając zastosowane w tabelach typy danych**](http://www.zs2krasnystaw.edu.pl/dla_ucznia/kurs_mysql/k_3_4_1.php#lancuchy_opis). Zapamiętaj, że **długość łańcuchów jest mierzona liczbą znaków je tworzących**. Łańcuchy zapisywane w kolumnach typu CHAR, zostają automatycznie uzupełnione spacjami aż do całkowitego wyczerpania ich długości. W kolumnach typu VARCHAR, łańcuchy zajmują tyle pamięci, ile wynosi ich rzeczywista długość. W przypadku występowania dodatkowych spacji, zostają one automatycznie usunięte. Tak więc w przypadku danych typu CHAR MySQL usunie dodatkowe spacje podczs pobierania danych z bazy, natomiast w przypadku VARCHAR, dodatkowe spacje są eliminowane podczas zapisywania danych.





Transformacja modelu związków encji do modelu relacyjnego

Reguły transformacji sa następujące:

* 1. Encja jest odwzorowywana w relację

Nazwa encji jest odworowywana w nazwę relacji. Uwaga: pryjmuje się że nazwy encji są rzeczownikami w liczbie mnogiej

* 1. Atrybut encji jest odwzorowywany w atrybut rwlacji

Nazwy atrybutów są odwzorowywane w nazwy atrybutów relacji

* 1. Typ danych