Asocjacje

Modelowanie powiązań bazodanowych za pomocą ActiveRecord

Typy baz danych a ActiveRecord

- Railsy zostały zaprojektowane do pracy z relacyjnymi bazami danych
- ActiveRecord może używać języka SQL tych baz danych do wykonywania kwerend
- Jednym z zadań ActiveRecord jest zarządzanie powiązanymi danymi w różnych tabelach.

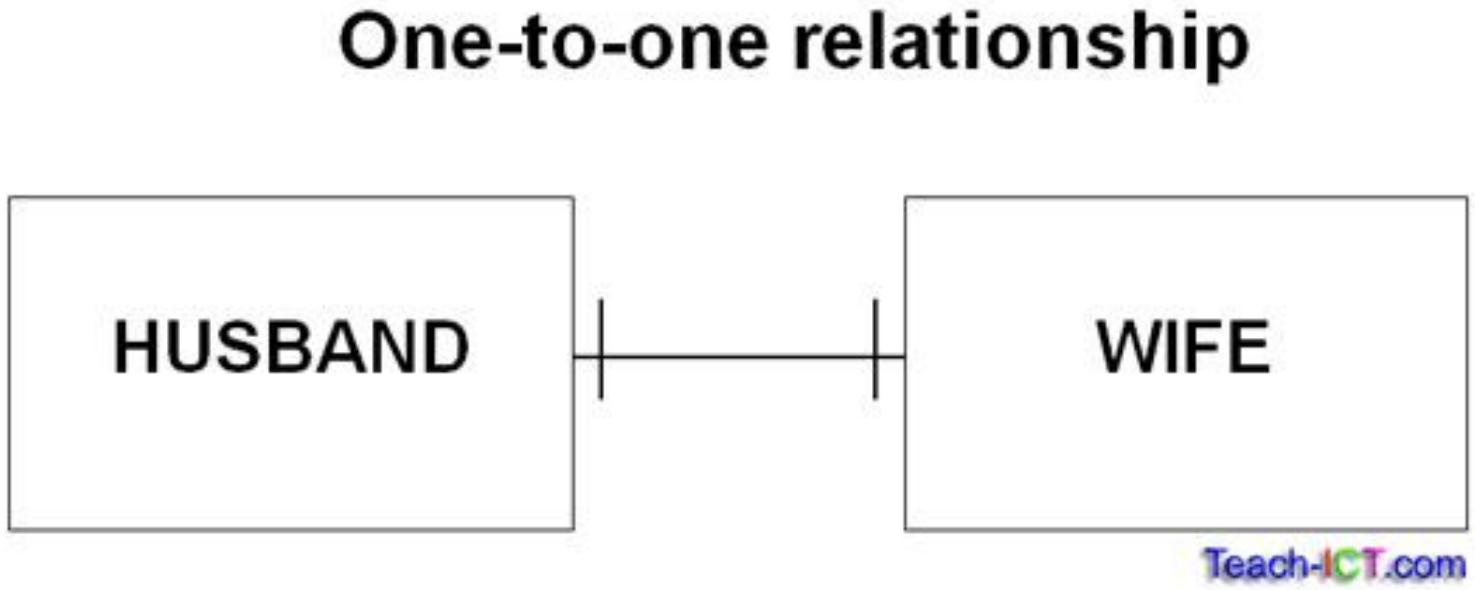
Typy relacji

W SQL instnieją tylko trzy typy powiązań

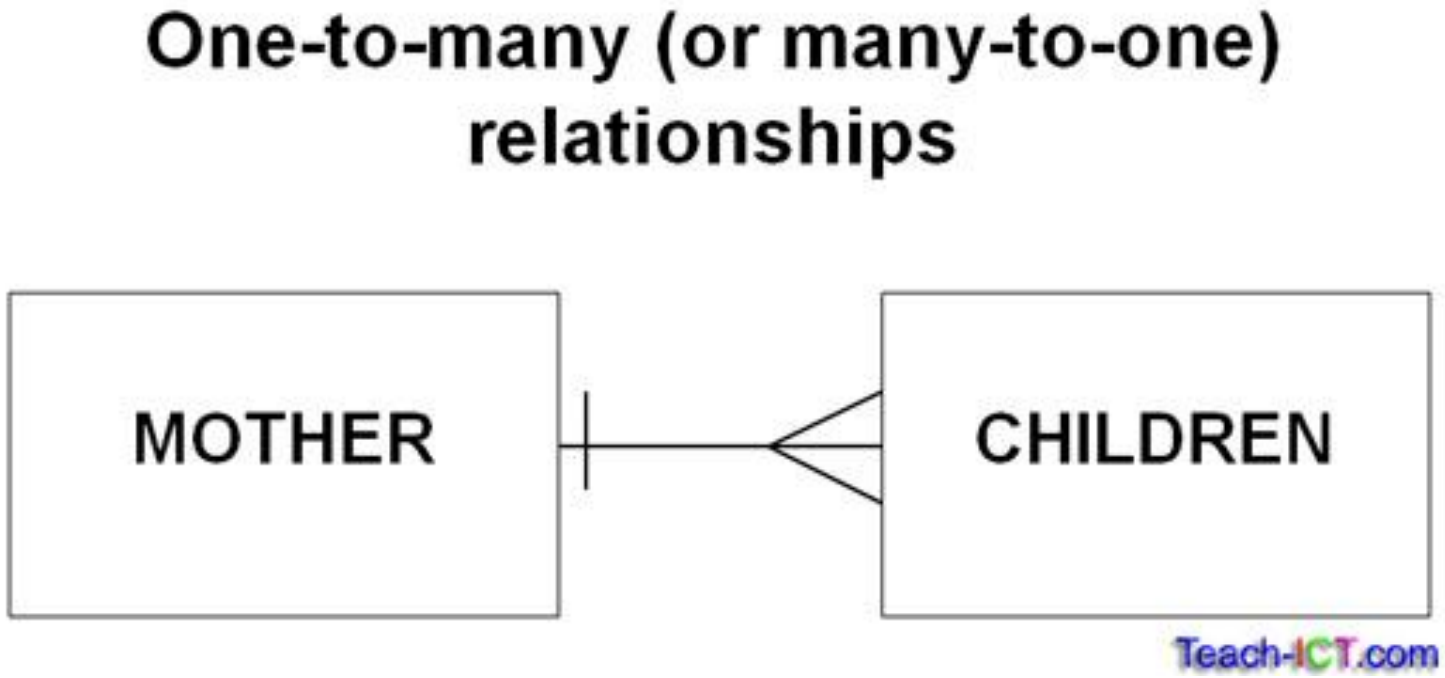
- jeden do jednego (one to one)
- jeden do wielu (one to many)
- wiele do wielu (many to many)

  - w rzeczywistości nie istnieje, ale można je zasymulować

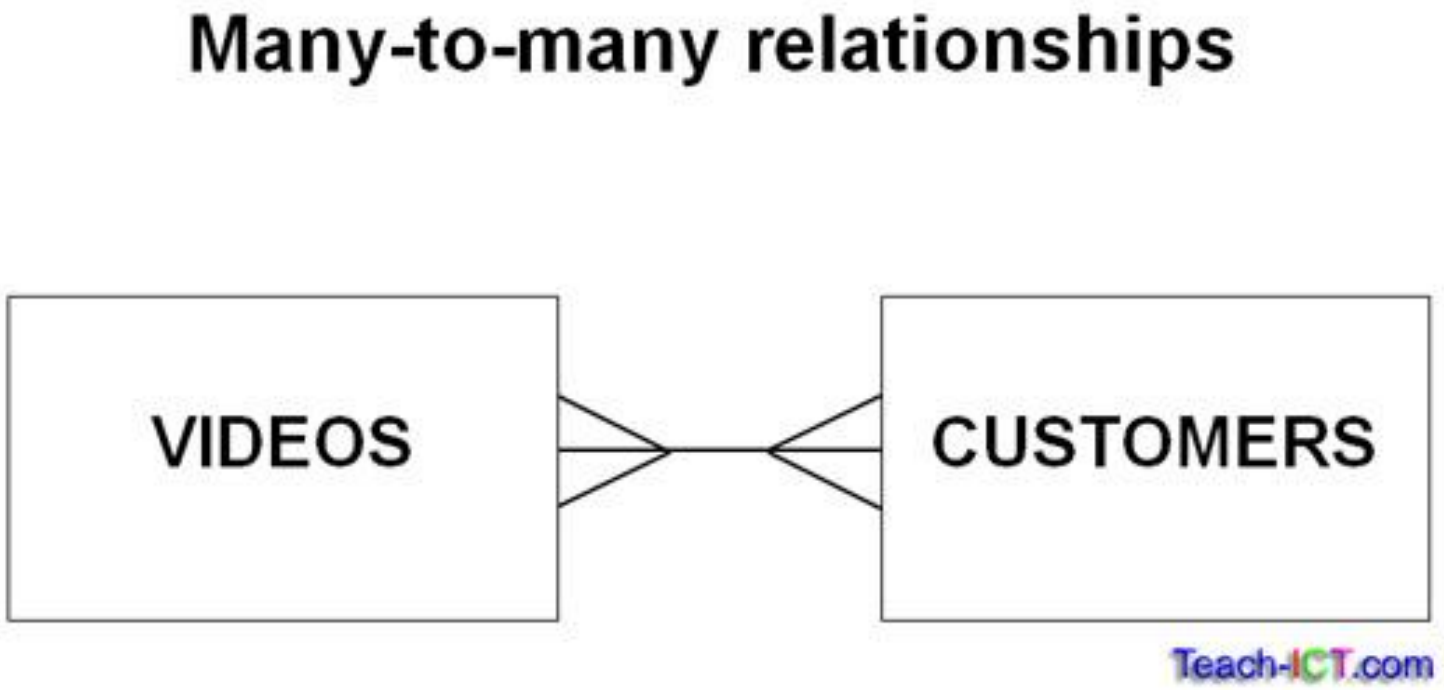
Jeden do jednego (one to one)



Jeden do wielu (one to many)



Wiele do wielu (many to many)



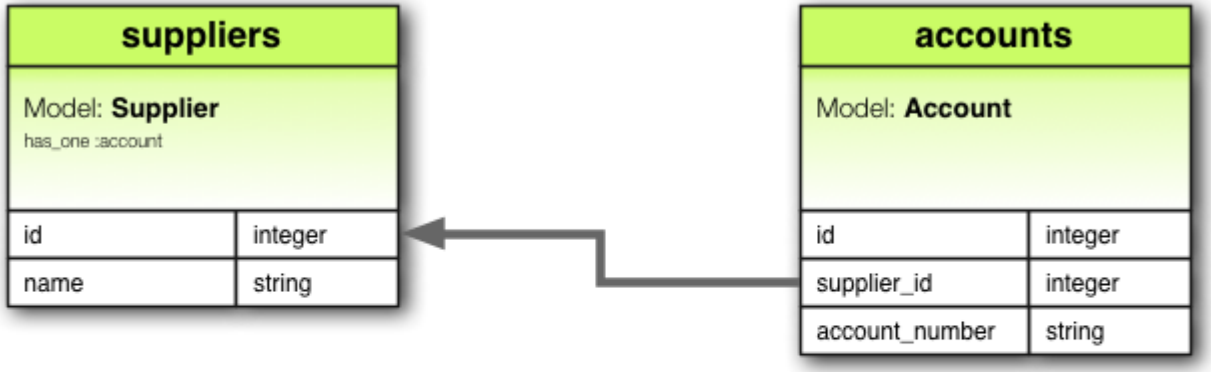
Konwencja asocjacji

- automatycznie zwiększające id (auto_increment)
- nazwa tabeli w bazie jest zapisana w liczbie mnogiej, ponieważ reprezentuje kolekcję. np. products
- klucz obcy jest zapisany w liczbie pojedyńczej np. user_id

Metody asocjacyjne ActiveRecord

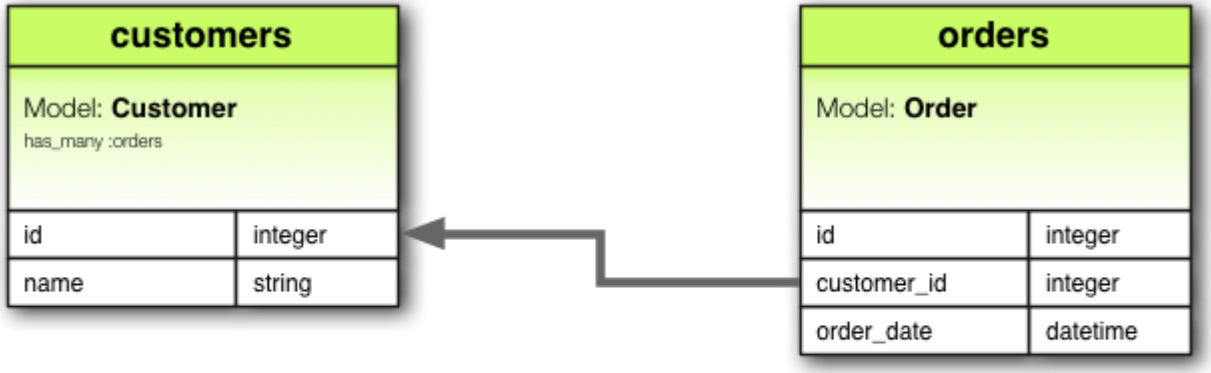
- belongs_to() po stronie tabeli z kluczem obcym w relacji one to one lub one to many
- has_one() po drugiej stronie relacji one to one
- has_many() po drugiej stronie one to many
- podwójny belongs_to() w tabeli łączącej relacji many to many
- has_many() i has_many(..., through: …) dla modeli z relacją many to many

Jeden do jednego (one to one)
Dostawca ma jedno konto



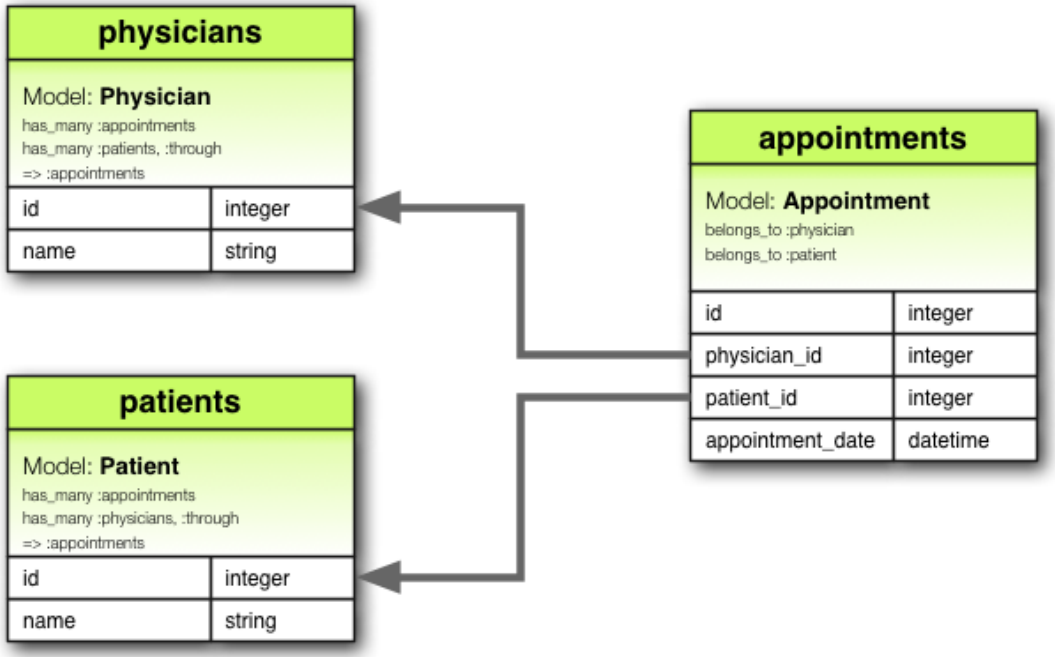

```
class Supplier < ActiveRecord::Base
  has_one :account
end
```

Jeden do wielu (one to many)
Klient ma wiele zamówień




```
class Customer < ActiveRecord::Base
  has_many :orders
end
```

Wiele do wielu (many to many)
Lekarz ma wielu pacjentów

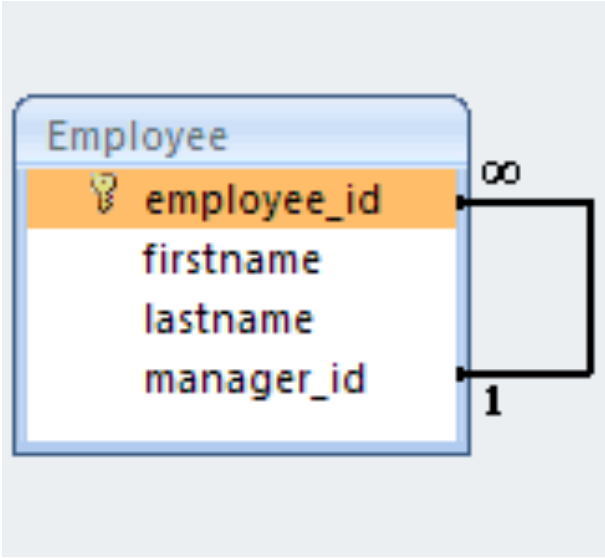


Self Join
Pracownik ma wielu podwładnych

Przypadek w którym model ma relację z samym sobą.

```
class Employee < ActiveRecord::Base
  has_many :subordinates, class_name: "Employee",
                          foreign_key: "manager_id"

  belongs_to :manager, class_name: "Employee"
end
```



Wiele do wielu (many to many)

```
class Physician < ActiveRecord::Base
  has_many :appointments
  has_many :patients, :through => :appointments
end

class Appointment < ActiveRecord::Base
  belongs_to :physician
  belongs_to :patient
end

class Patient < ActiveRecord::Base
  has_many :appointments
  has_many :physicians, :through => :appointments
end
```

Używając asocjacji poprawnie

- unikaj odwołań poprzez ID

  - pozwój Railsom się tym zająć
- bezpieczniej jest zawężać wyszukiwanie poprzez jak najwięcej asocjacji

  - mniejsza szansa na niechciane zmiany w danych

  - dodaje to bezpieczeństwa przed użytkownikami próbującymi podmieniać ID

Opcje asocjacji railsowych

- ustaw :conditions, :order, etc. na asocjacjach
  - przykład: order: “created_at DESC”
- sprecyzuj co stanie się z asocjacją po usunięciu powiązanego rekordu metodą destroy()
  - dependend: :destroy forwarduje destroy()
  - dependend: :nulify przerywa asocjacje

Asocjacje polimorficzne

- Modele są powiązane kluczami _type i _id
  - to pozwala na podłączenie do modelu różnych typów innych modeli
  - na przykład możesz użyć modelu Comment aby pozwolić użytkownikom komentować instacje różnorodnych modeli, np. Artykuły i Autorów.