Systemy Baz Danych

Model Związków Encji

Bartosz Zieliński

WYDZIAŁ FIZYKI
i INFORMATYKI STOSOWANEJ
Uniwersytet Łódzki

Modelowanie Fizyczne, Logiczne i Pojęciowe

- **Model pojęciowy** — o jakiego rodzaju obiektach chcemy przechowywać informacje w bazie, jakie są ich cechy, jakie są powiązania pomiędzy obiektami różnych typów.
  - Tworząc model pojęciowy staramy się zrozumieć strukturę opisywanego w bazie fragmentu rzeczywistości.
- **Model logiczny** — staramy się tu odwzorować model pojęciowy na konkretny model danych stosowany przez DBMS — np. na model relacyjny.
  - Model logiczny zawiera strukturę zmiennych relacyjnych (włącznie z więzami), także strukturę zmiennych modelujących powiązania wiele-do-wielu pomiędzy zmiennymi, widoki, itp.
- **Model fizyczny** — zawiera szczegóły implementacyjne i poprawiające efektywność wykonywania zapytań takie jak partycjonowanie danych, indeksy, rozmieszczenie plików z danymi po różnych dyskach dla zrównoleglenia dostępu i wiele innych.

Rozdzielenie Modelu Fizycznego od Logicznego w Relacyjnych DBMS

Jedną z największych zalet relacyjnych DBMS jest to że kod aplikacji korzystającej z bazy danych zależy (za pośrednictwem poleceń DML i zapytań) jedynie od modelu logicznego a nie fizycznego danych.

Przykład

Indeksy są specjalnymi strukturami które można założyć na kolumnach tabel w celu przyśpieszenia wyszukiwania po tych kolumnach. Np. index na kolumnie **JobID** przyśpieszy wykonanie zapytań typu

```
SELECT * FROM Employees WHERE JobId=10
```

nie jest jednak wymagany aby takie zapytania były wykonalne, ani jego obecność nie uwidacznia się w żaden sposób w kodzie zapytania

Model Związków Encji

Do modelowania pojęciowego najczęściej korzysta się z **modelu związków encji** (*entity/relationship model*, inaczej modelu **ER**) a przede wszystkim ze związanym z tym modelem graficznym językiem diagramów **ER**.

W modelu związków encji wyróżniamy:

- **Encje** (*Entities*) — modelują typy (klasy) rzeczy, np. **osoba, pracownik, wydział, produkt**, itp.
- **Atrybuty** — cechy encji, np. *kolor* dla **produktu**, *imię, nazwisko* i *PESEL* dla **osoby** itp.
- **Związki** — modelują powiązania pomiędzy encjami, np. **pracownik** zatrudniony na **wydziale**.
  - Związki są powiązaniami pomiędzy instancjami encji, tzn. np. nie pracownik jako typ jest zatrudniony na typie wydziału, ale konkretny pracownik na konkretnym wydziale.

Bibliografia

Wykład o diagramach **ER** opiera się przede wszystkim na

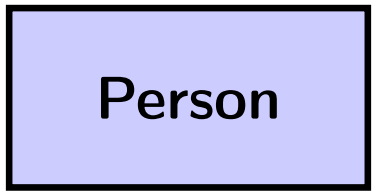
- H. Garcia-Molina, J.D. Ullman, J. Widom „*Systemy Baz Danych. Pełny Wykład*", **Wydawnictwa Naukowo-Techniczne**, Warszawa 2006
- Pável Calado „*The Tikz-er2 Package for Drawing Entity-Relationship Diagrams*" (2010) [Stąd wziąłem część przykładów. Pakiet został też wykorzystany do rysowania diagramów]

Istnieje wiele różnych wariantów notacji diagramów **ER**. Korzystam z wariantu przedstawionego w książce H. Garcia-Moliny.

Interesującą krytykę **ER** można znaleźć w książce

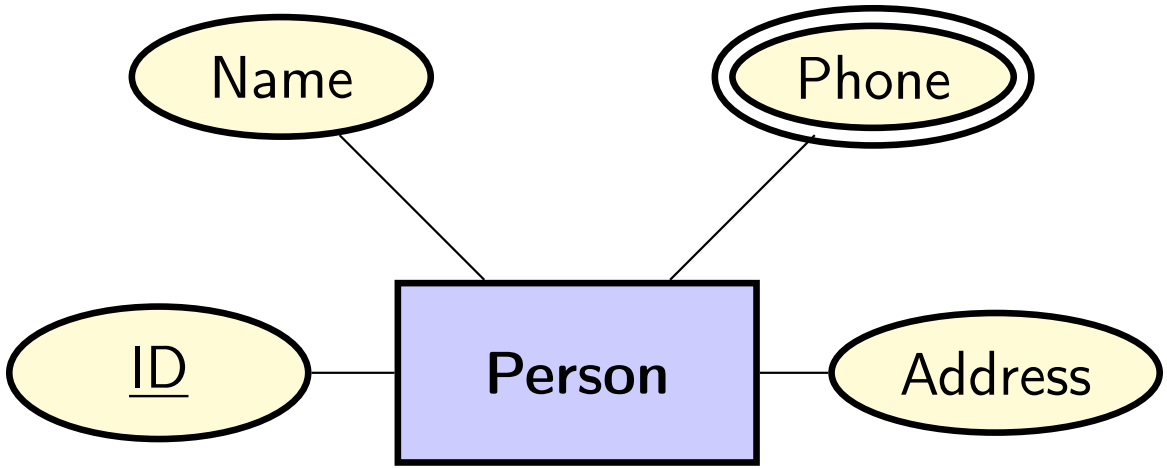
- C.J. Date „*An Introduction to Database Systems*", **Addison Wesley**, 2004

Encje



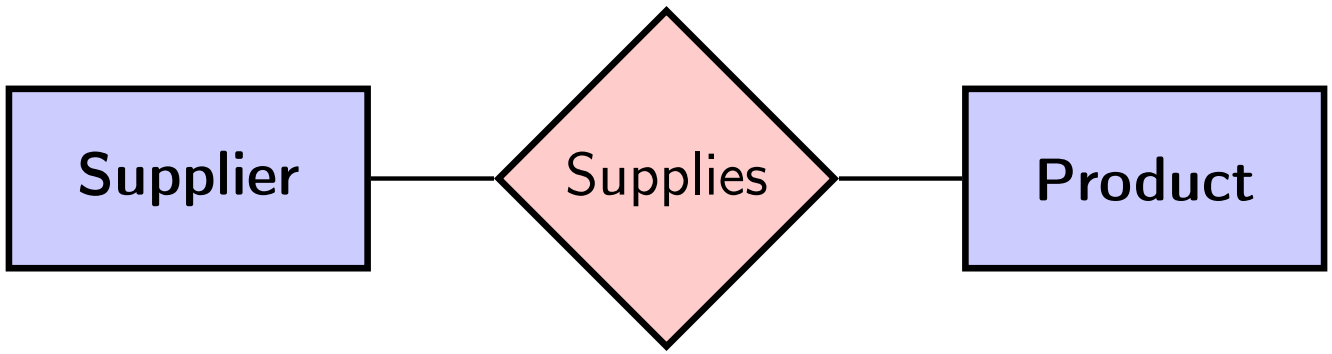
- **Encje** (*Entities*) — modelują typy (klasy) rzeczy, np. **osoba, pracownik, wydział, produkt**, itp.
- Na diagramach są zaznaczane przy pomocy prostokąta z nazwą encji w środku (powyżej: encja **Person**).
- Encje można zinterpretować jako zbiory swoich możliwych instancji — faktycznie istniejących obiektów danego typu.
  - Np. encja **Person** z diagramu powyżej to zbiór wszystkich prawdziwych osób o których informacje chcemy przechowywać w bazie danych.

Atrybuty



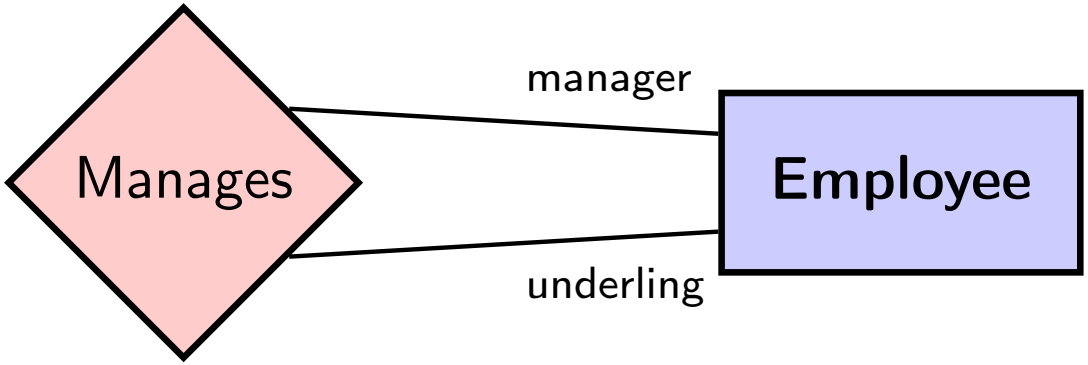
- **Atrybuty** opisują cechy instancji encji, np. *kolor* dla **produktu**, *imię, nazwisko* i *PESEL* dla **osoby** itp.
- **Podkreślamy** (jak ID powyżej) atrybuty jednoznacznie identyfikujące instancję danej encji.
- **Z podwójnym brzegiem** rysujemy te atrybuty które są wielowartościowe. Np. osoba może mieć wiele telefonów.

Związki



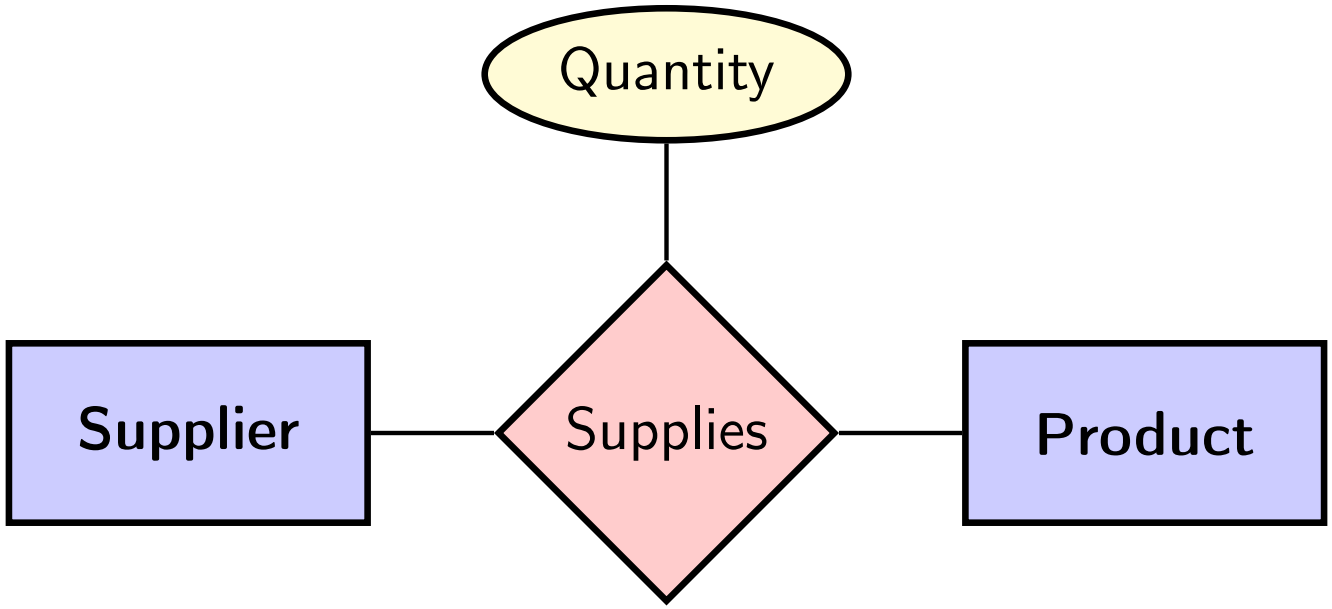
- **Związki** — wyrażają powiązania pomiędzy instancjami encji.
- Powyżej: związek **Supplies** wyrażający fakt że (dany) dostawca **dostarcza** (danego) produktu.
- W większości przypadków **związki** są binarne (łączą dwie, niekoniecznie różne, encje). Model **ER** dopuszcza jednak **związki** łączące więcej niż dwie encje.

Role Uczestników Związku



- Ta sama encja może w **związku** występować wielokrotnie, w różnych rolach (najczęściej oznacza to że związek wyraża powiązanie różnych instancji tej samej encji).
- Na diagramie można nazwać role w jakich występuje w danym związku encja (także gdy występuje jednokrotnie).
- Np. powyżej mamy związek **manages** łączący pracownika będącego podwładnym (**underling**) z jego szefem (**manager**) który oczywiście jest także pracownikiem

Atrybuty Związków



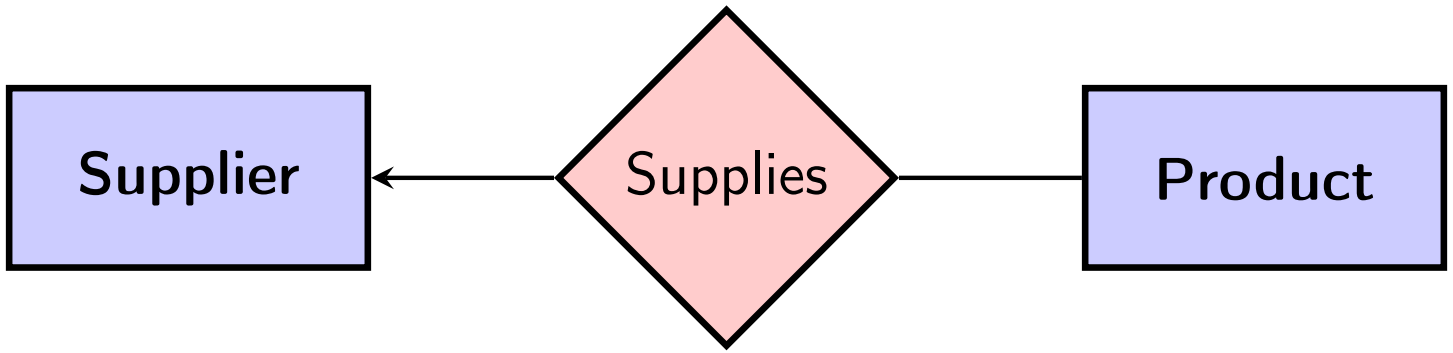
- **Związki** mogą mieć własne **atrybuty** opisujące cechy danego związku.
- Np. w przykładzie powyżej atrybut **Quantity** opisuje ilość danego towaru **dostarczanego** przez danego dostawcę.

Krotność Związków

- Na diagramach **ER** można zaznaczać **krotność związków**.
- Brzeg po stronie „**jeden**” oznaczamy strzałką, brzeg po stronie „**wiele**” pozostawiamy bez oznaczenia.
- Krotność związku najłatwiej zrozumieć dla **związków binarnych** gdzie wyróżniamy:
  - Związek **jeden do wielu** (i **wiele do jeden**)
  - Związek **wiele do wielu**
  - Związek **jeden do jeden**

Krotność Związków

Związek Jeden do Wielu



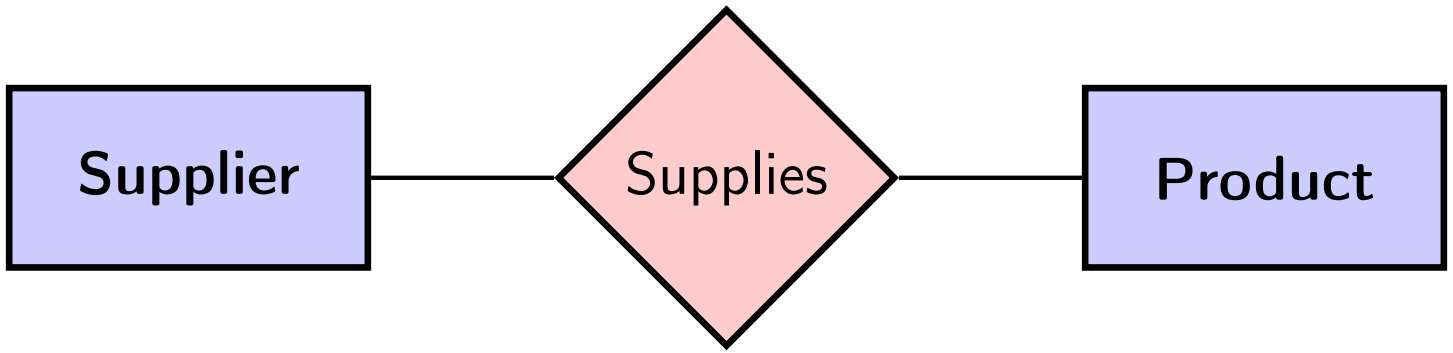
Dany **produkt** dostarczany jest przez **co najwyżej** jednego **dostawcę** (ale dany dostawca może dostarczać dowolną ilość produktów).

Związek jest jeden do wielu (lub wiele do jeden)

gdy instancja encji stojącej po stronie „**wiele**" może być w tym związku z **co najwyżej** jedną instancją encji stojącej po stronie „**jeden**" (oznaczonej strzałką). Instancja po stronie „**jeden**" może być w związku z dowolną ilością instancji (także żadną) po stronie „**wiele**".

Krotność Związków

Związek Wiele do Wielu



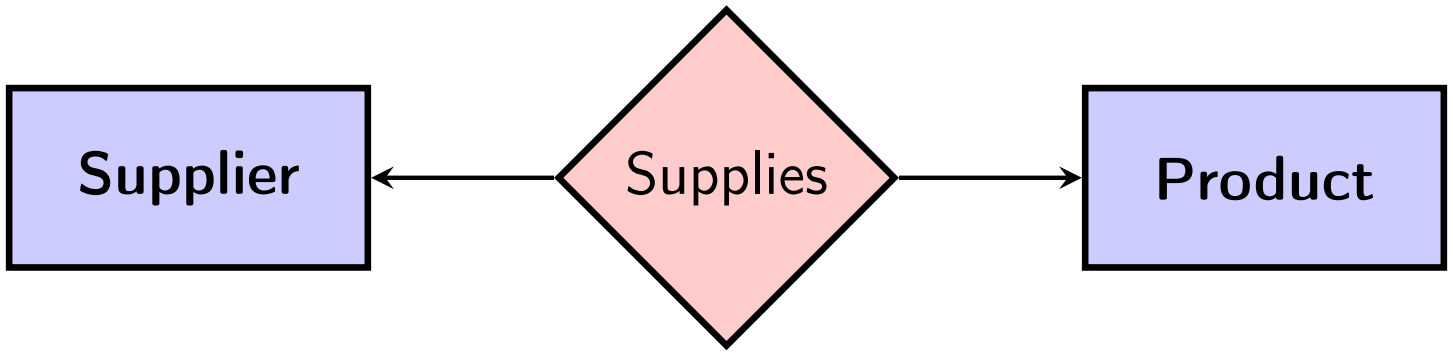
Dany **dostawca** może **dostarczać** dowolną ilość **produktów**, a dany **produkt** może być **dostarczany** przez dowolną ilość **dostawców**.

Związek jest jeden do wielu

kiedy instancje jednej z encji mogą być w związku z dowolną ilością instancji drugiej encji i odwrotnie.

Krotność Związków

Związek Jeden do Jeden



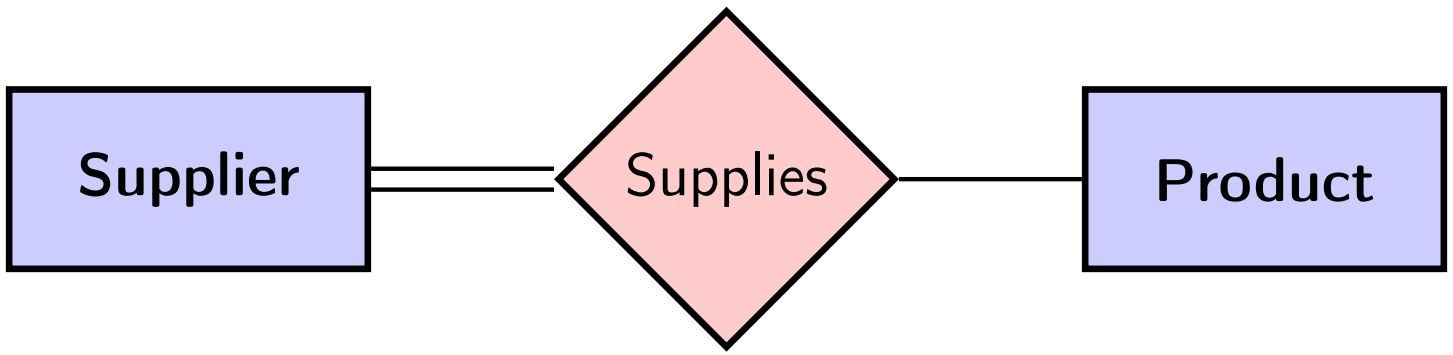
Dany **produkt** może być **dostarczany** przez co najwyżej jednego **dostawcę** a każdy **dostawca dostarcza** co najwyżej jeden **produkt**

Związek jest Jeden do Jeden

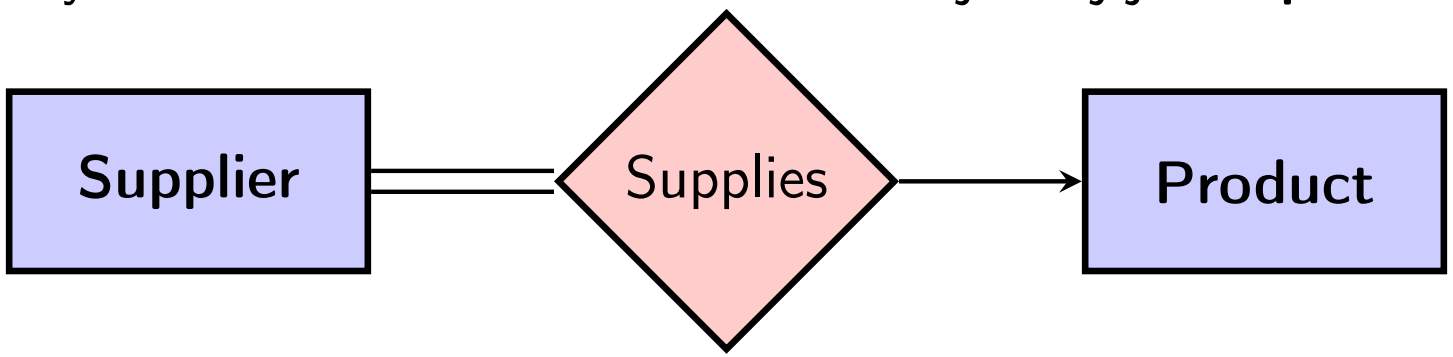
kiedy instancje jednej z encji mogą być w związku z co najwyżej jedną instancją drugiej encji i odwrotnie.

Całkowite Uczestnictwo w Związku

Mówimy że encja **uczestniczy całkowicie w związku** gdy każda instancja encji musi uczestniczyć w danym związku z co najmniej jedną instancją drugiej encji uczestniczącej w związku.

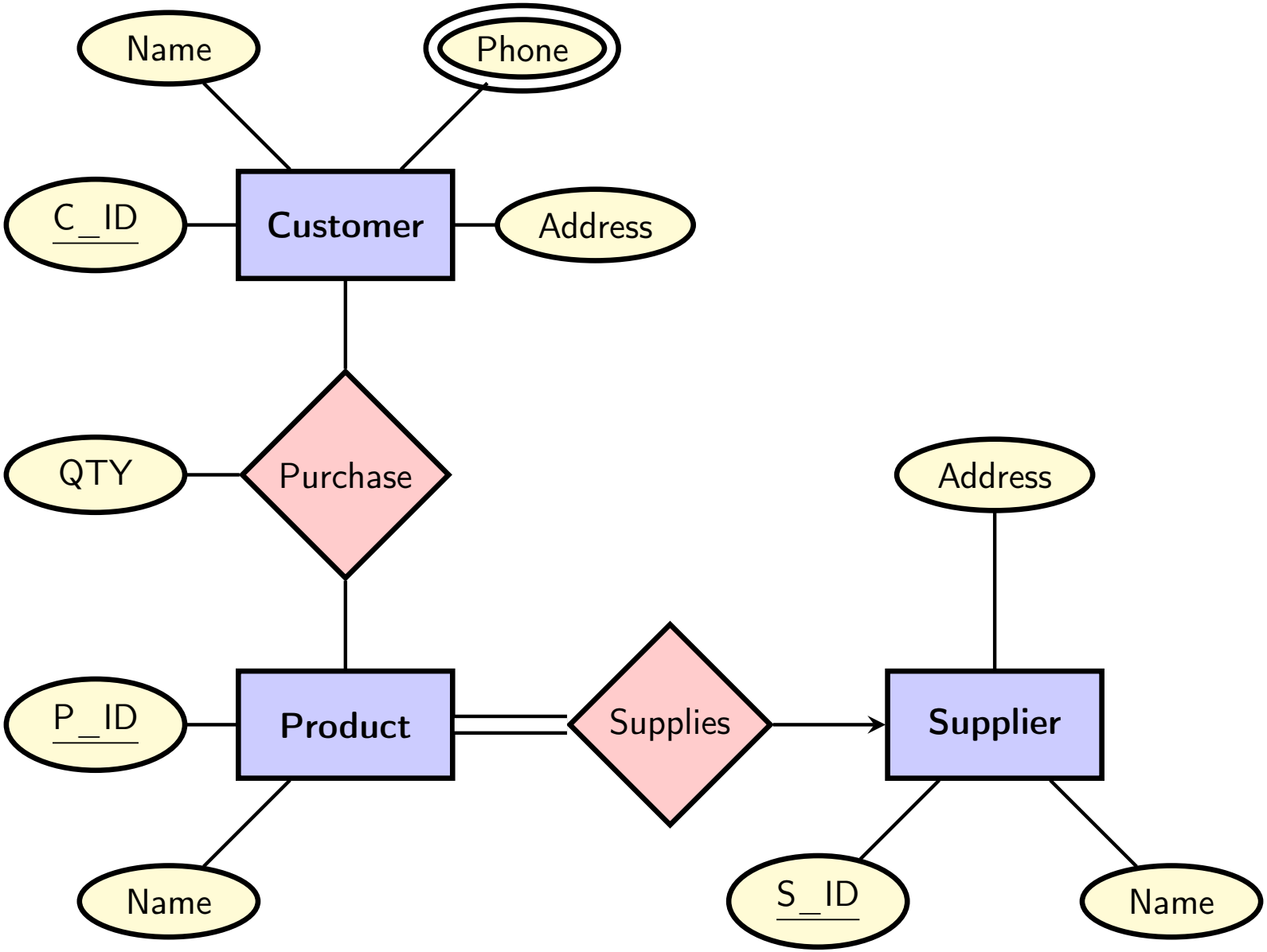


Dany **dostawca** musi **dostarczać** co najmniej jeden **produkt**



Dany **dostawca** musi **dostarczać** **dokładnie** jeden **produkt**

Pełen Przykład



Od Diagramu ER do Schematu Bazy Danych

Część I

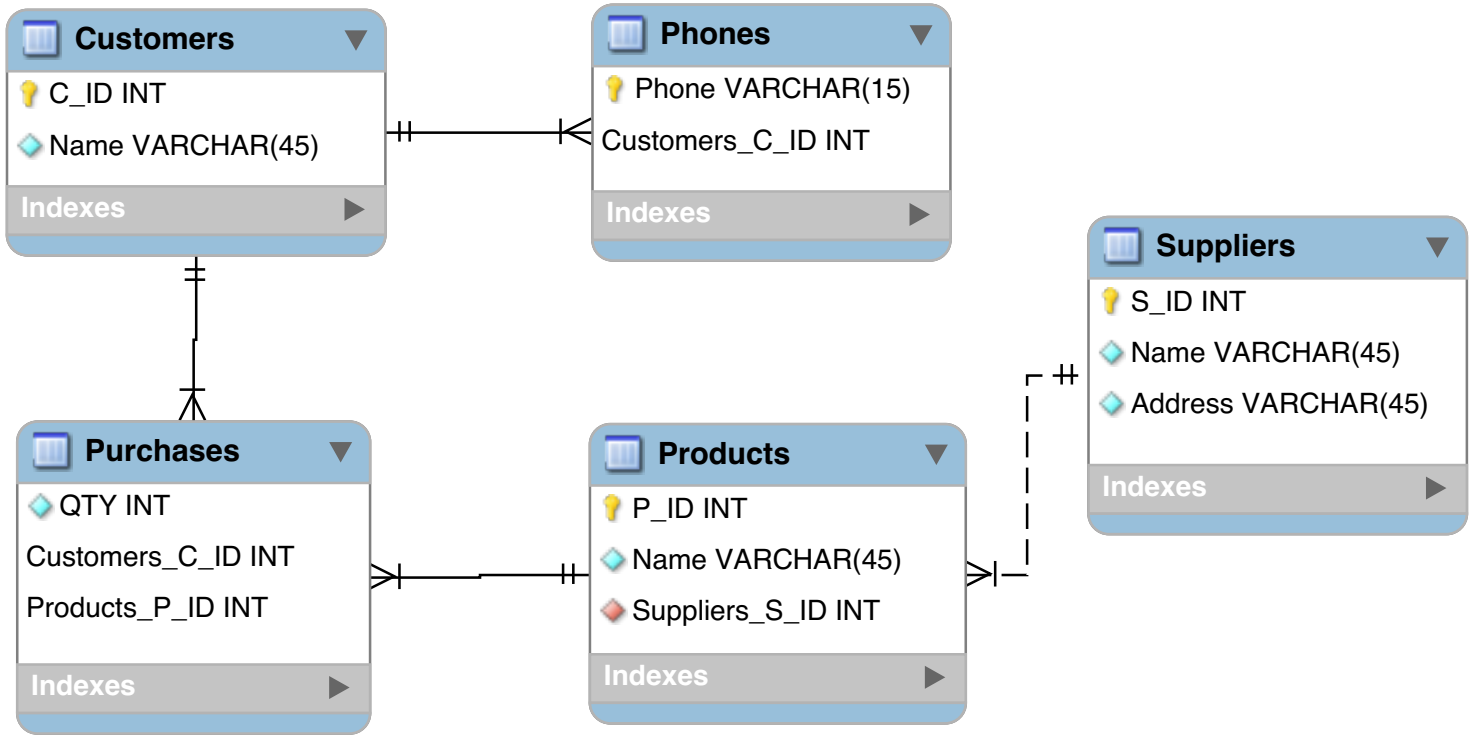
- Każdemu **atrybutowi** musimy przypisać typ wartości.
- Dla każdej **encji** i każdego **związku** (poza związkami binarnymi **jeden do wielu** lub **jeden do jeden** tworzymy zmienną relacyjną.
- **Atrybutami** tych zmiennych będą atrybuty jednowartościowe odpowiadających **encji** i **związków**. Dodatkowo
  - dla każdej tabeli odpowiadającej **związkowi** dodajemy klucze obce do wszystkich **encji** biorących udział w związku. Kombinacja tych kluczy będzie stanowiła zwykle klucz główny,
  - dla każdego **związku wiele do jeden** dodajemy dla tabeli odpowiadającej **encji** stojącej po stronie **wiele** klucz obcy do encji stojącej po stronie **jeden**.
  - W przypadku **związku jeden do jeden** wybieramy jedną z encji biorących udział w związku aby dodać do odpowiadającej tabeli klucz obcy na który dodatkowo nakładamy wiąz **UNIQUE**.

- Dla każdego **atrybutu wielowartościowego** A tworzymy zmienną relacyjną z następującymi dwoma atrybutami:
  1. atrybutem przechowującym pojedyńczą wartość A
  2. kluczem obcym do encji do której przynależy A.

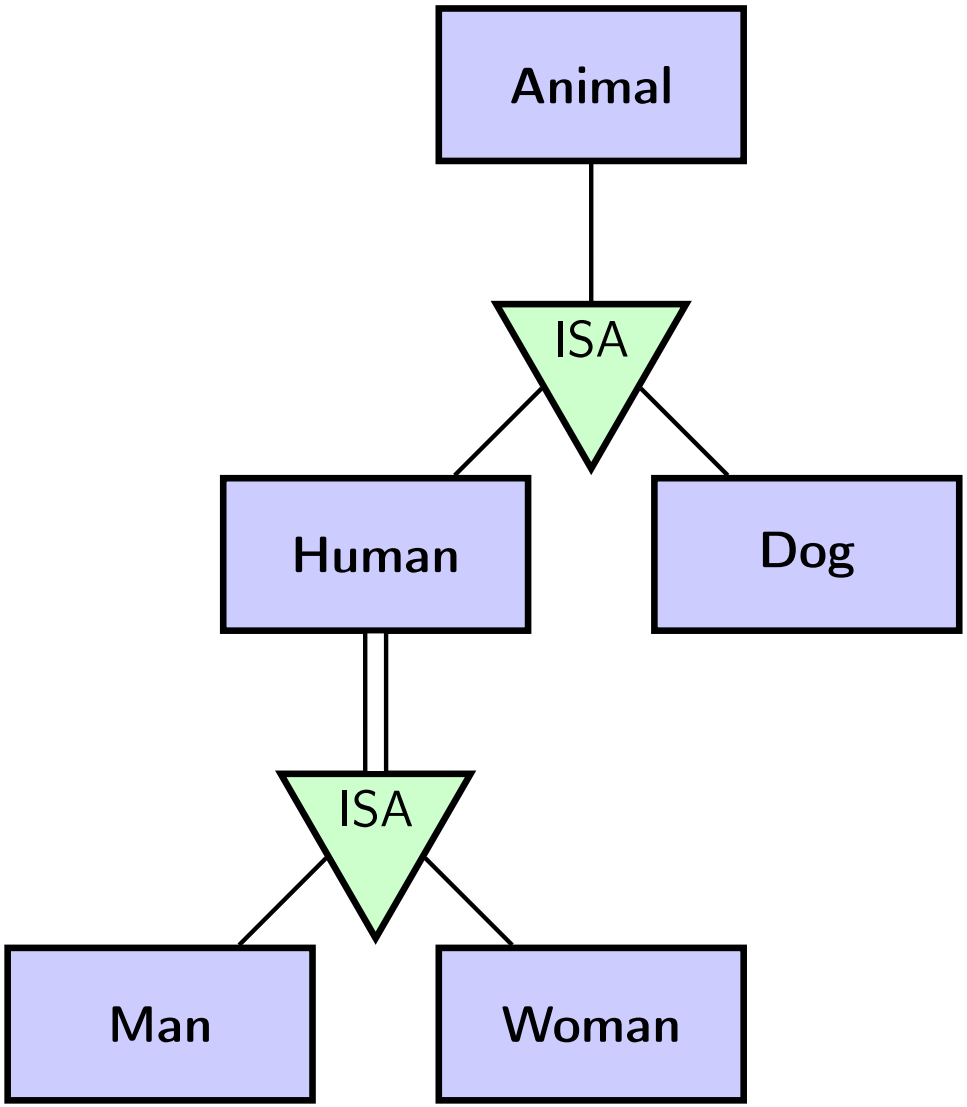
Diagramy ER i Normalizacja

- Na ogół, jeśli dokonamy poprawnego wyboru **encji**, **atrybutów** i **związków** (z odpowiednią krotnością), analizując strukturę danych które chcemy przechowywać w bazie, wówczas otrzymany bazę danych której schemat zawiera znormalizowane (co najmniej do **3NF**) zmienne relacyjne.
- Można to również odwrócić, mówiąc że cechą dobrego schematu **ER** jest to że daje się go bezpośrednio przetłumaczyć na odpowiednio znormalizowany schemat relacyjnej bazy danych.

Przykład


Customers
C_ID INT
Name VARCHAR(45)
Indexes
Phones
Phone VARCHAR(15)
Customers_C_ID INT
Indexes
Suppliers
S_ID INT
Name VARCHAR(45)
Address VARCHAR(45)
Indexes
Purchases
QTY INT
Customers_C_ID INT
Products_P_ID INT
Indexes
Products
P_ID INT
Name VARCHAR(45)
Suppliers_S_ID INT
Indexes

Specjalizacja/Generalizacja



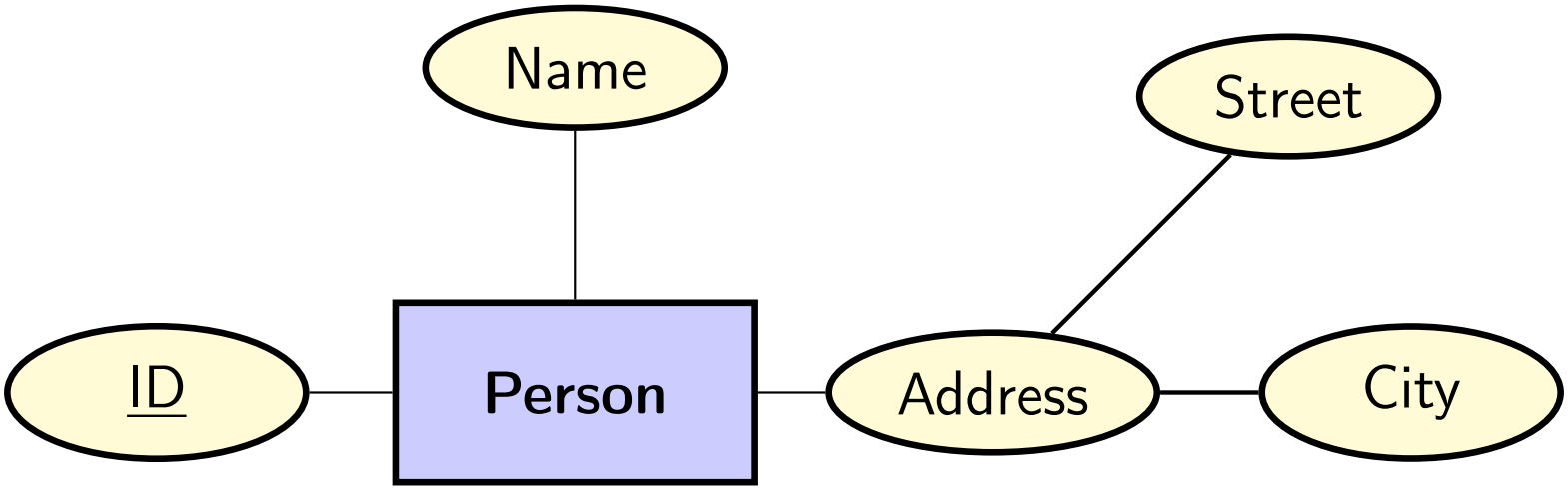
Związek **ISA** jest specjalnym rodzajem **związku** który można porównać do dziedziczenia w programowaniu obiektowym.

- Każdy **człowiek** i **pies** jest **zwierzęciem**
- Każdy człowiek jest **mężczyzną** albo **kobietą**

Encje będące **specjalizacją** innych encji dziedziczą atrybuty po tych encjach ale mogą też dodawać własne atrybuty.

- **Człowiek** jest specjalizacją **zwierzęcia**,
- **zwierzę** generalizuje **człowieka**

Złożone Atrybuty



- Atrybut mający własne atrybuty nazywa się **atrybutem złożonym**
- Można uznać że atrybut **Address** w przykładzie powyżej ma typ rekordowy z dwoma polami **Street** i **City**
- **Atrybutów złożonych** nie można reprezentować bezpośrednio w czystym modelu relacyjnym — encji **Person** będzie odpowiadała zmienna relacyjna z czterema atrybutami: **ID**, **Name**, **Street** i **City**

Obiektowo Relacyjne i Postrelacyjne DBMS-y

Wielowartościowych i złożonych atrybutów ani związków **ISA** pomimo ich naturalności i użyteczności nie da się reprezentować **bezpośrednio** w modelu relacyjnym, choć można to zrobić stosując dodatkowe zmienne relacyjne z nałożonymi więzami referencyjnymi.

Postrelacyjne i Obiektowo-Relacyjne DBMS-y których przykładem są **PostgreSQL** i **Oracle** pozwalają między innymi na

- Definiowanie własnych (także złożonych) typów przez użytkownika razem z operacjami domenowymi na nich wykonywalnymi
- Pozwalają na przechowywanie w atrybutach kolekcji elementów a nie tylko pojedyńczych wartości
- Pozwalają na deklarowanie zmiennych relacyjnych dziedziczących po innych tabelach.
- Pozwalają także na definiowanie kodu proceduralnego wykonywanego i przechowywanego przez DBMS w postaci **podprogramów składowanych** i **triggerów** (wyzwalaczy).