Bazy danych

Andrzej Grzybowski

Instytut Fizyki, Uniwersytet Śląski

Wykład 6

Model relacyjny danych – projektowanie relacyjnych baz danych, model logiczny i relacyjny, zastosowanie Oracle SQL Developer Data Modeler

Model logiczny – koncepcyjny (konceptualny) oraz pełny na przykładzie modelu związków encji (*ERD – Entity Relationship Diagram)*

W modelu związków encji:

- **ENCJE** reprezentują **obiekty świata rzeczywistego**
- **ZWIĄZKI między ENCJAMI** opisują zależności między obiektami świata rzeczywistego

Stosuje się różne notacje w modelach związków encji, spośród których omówione zostaną dwie:

- notacja Chena (o znaczeniu głównie historycznym i dydaktycznym)
- notacja Barkera (preferowana w narzędziach Oracle)

Model logiczny – koncepcyjny (konceptualny) oraz pełny na przykładzie modelu związków encji (*ERD – Entity Relationship Diagram)*

Charakterystyka encji:

- Encja reprezentuje zbiór obiektów opisanych tymi samymi cechami (atrybutami, własnościami).
- Dany obiekt świata rzeczywistego jest reprezentowany jako wystąpienie encji (instancja encji).
- Dana rzecz lub obiekt świata rzeczywistego powinien być reprezentowany tylko przez jedną encję.
- Każda encja posiada unikalną nazwę.
- Nazwa encji powinna być rzeczownikiem w liczbie pojedynczej (obecnie ten wymóg często nie jest respektowany).
- Encje zwykle wchodzą w związki z innymi encjami z wyjątkiem encji, reprezentujących dane słownika i konfiguracyjne bazy danych

Model logiczny – koncepcyjny (konceptualny) oraz pełny na przykładzie modelu związków encji (*ERD – Entity Relationship Diagram)*

Charakterystyka atrybutu encji:

- Atrybut encji musi posiadać unikalną nazwę, odróżniającą go od innych atrybutów tej encji.
- Atrybutu encji posiada określoną dziedzinę danych.
- Atrybut encji może dozwalać brak wartości (NULL) bądź zawsze wymagać podania wartości (NOT NULL)
- Wartości atrybutu mogą być unikalne bądź nie.

Model logiczny – koncepcyjny (konceptualny) oraz pełny

na przykładzie modelu związków encji (*ERD – Entity Relationship Diagram)*

Kategorie klasyfikacji związków (asocjacji) między encjami:

- Stopień
- Kardynalność
- Przynależność

Klasyfikacja związków (asocjacji) między encjami wg kategorii:

- Stopień związku
  - unarny (binarny rekursywny, binarny rekurencyjny)
  - binarny
  - ternarny
  - *n*-arny
- Kardynalność (*jeden* bądź *wiele* charakteryzuje końcówkę związku !)
  - jeden do jeden
  - jeden do wiele
  - wiele do wiele
- Przynależność (*opcjonalny* bądź *obowiązkowy* także charakteryzuje końcówkę związku !)
  - opcjonalny (obustronnie)
  - obowiązkowy (obustronnie)
  - jednostronnie opcjonalny lub jednostronnie obowiązkowy

Model logiczny – koncepcyjny (konceptualny) oraz pełny

na przykładzie modelu związków encji (*ERD – Entity Relationship Diagram)*

Notacja Chena:

- Symbol encji – prostokąt
- Symbol atrybutu encji – koło
- Symbol związku między encjami – romb
- Symbole kardynalności końcówki związku między encjami:
  - jeden: 1
  - wiele: N bądź M
- Symbole kardynalności związków:
  - jeden do jeden 1 : 1
  - jeden do wiele 1 : N
  - wiele do wiele N : M

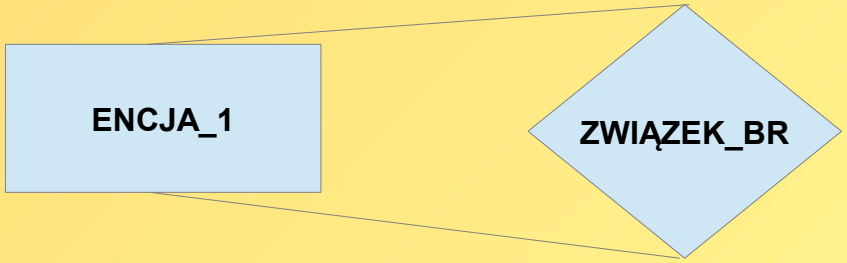
Niekiedy określa się, że **model koncepcyjny (konceptualny) to model związków encji bez atrybutów**. Natomiast **model logiczny (pełny)** można przedstawić jako **model związków encji ze zdefiniowanymi atrybutami encji**.

Model logiczny – koncepcyjny (konceptualny) oraz pełny

na przykładzie modelu związków encji (*ERD – Entity Relationship Diagram)*

- Związek binarny w notacji Chena – model konceptualny

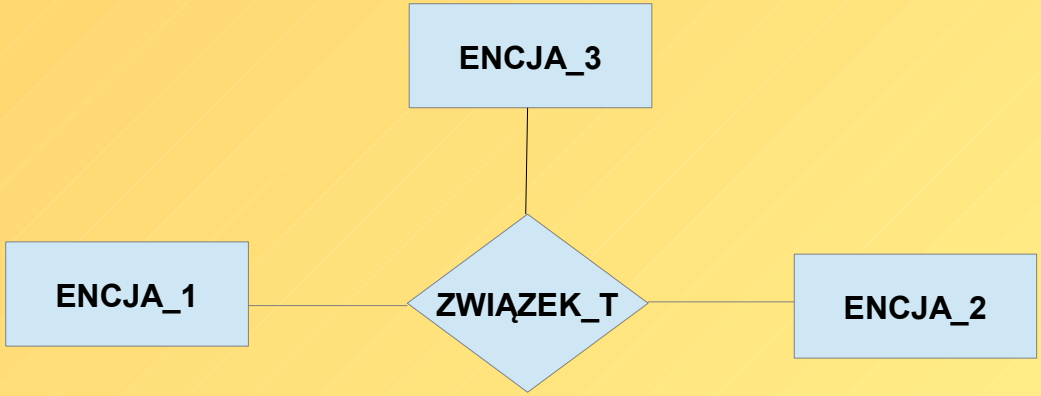
- Rekursywny związek binarny (związek unarny) w notacji Chena – model konceptualny



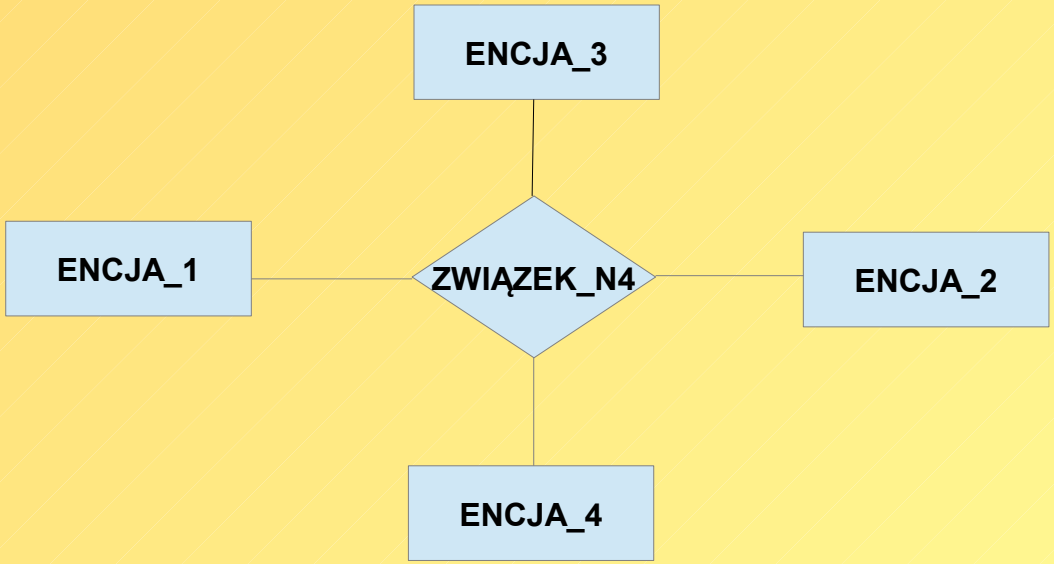
Model logiczny – koncepcyjny (konceptualny) oraz pełny

na przykładzie modelu związków encji (*ERD – Entity Relationship Diagram)*

- Związek ternarny w notacji Chena – model konceptualny



Związek *n*-arnego w notacji Chena dla *n* = *4* – model konceptualny



Model logiczny – koncepcyjny (konceptualny) oraz pełny

na przykładzie modelu związków encji (*ERD – Entity Relationship Diagram)*

- Związek binarny typu 1:1 w notacji Chena – model konceptualny

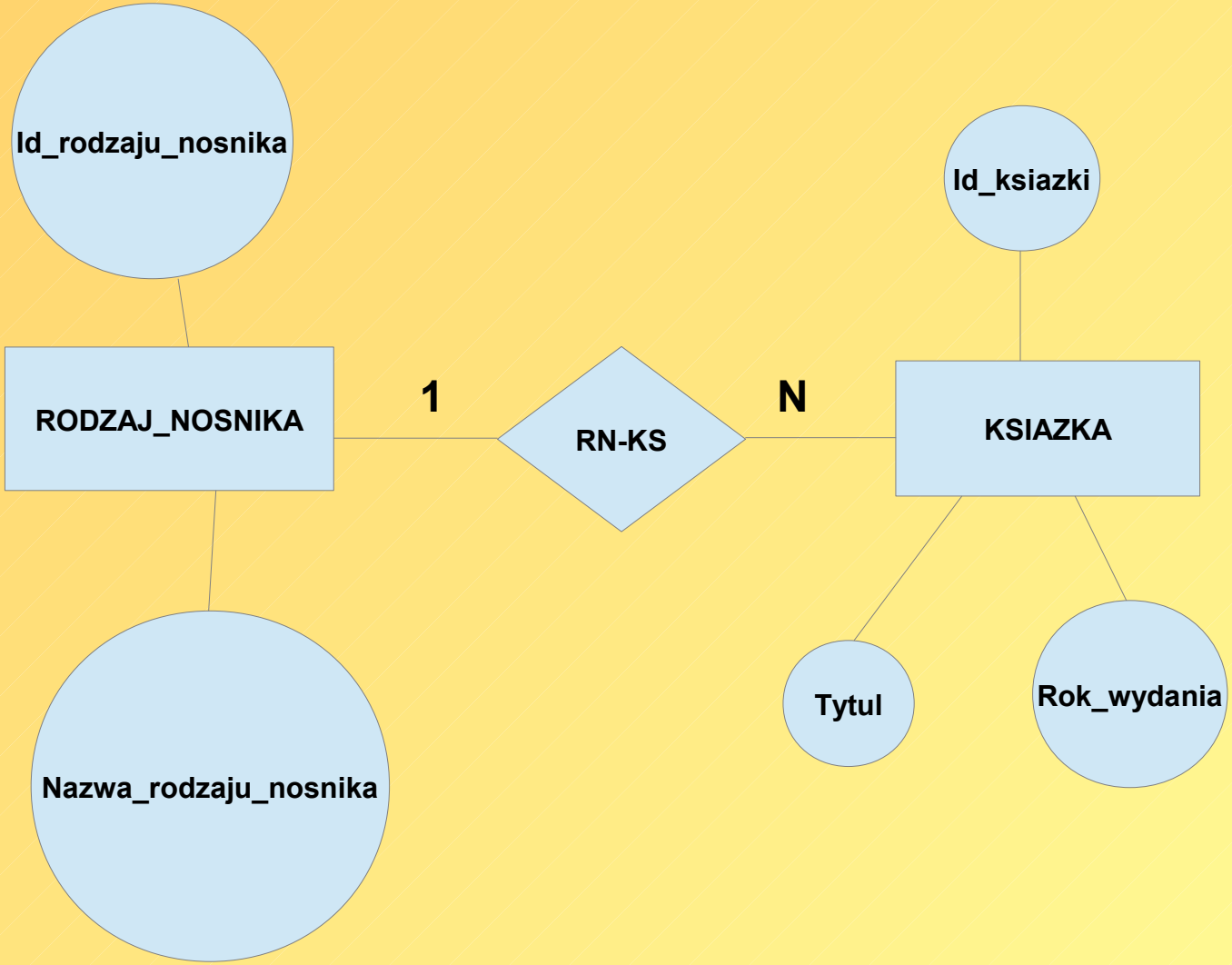
- Związek binarny typu 1:N w notacji Chena – model konceptualny

- Związek binarny typu N:M w notacji Chena – model konceptualny

Model logiczny – koncepcyjny (konceptualny) oraz pełny

na przykładzie modelu związków encji (*ERD – Entity Relationship Diagram)*

- Przykład związku binarnego typu 1:N w notacji Chena – model logiczny pełny



Przypomnienie z poprzedniego wykładu przykładu dekompozycji relacji KSIĘGOZBIÓR

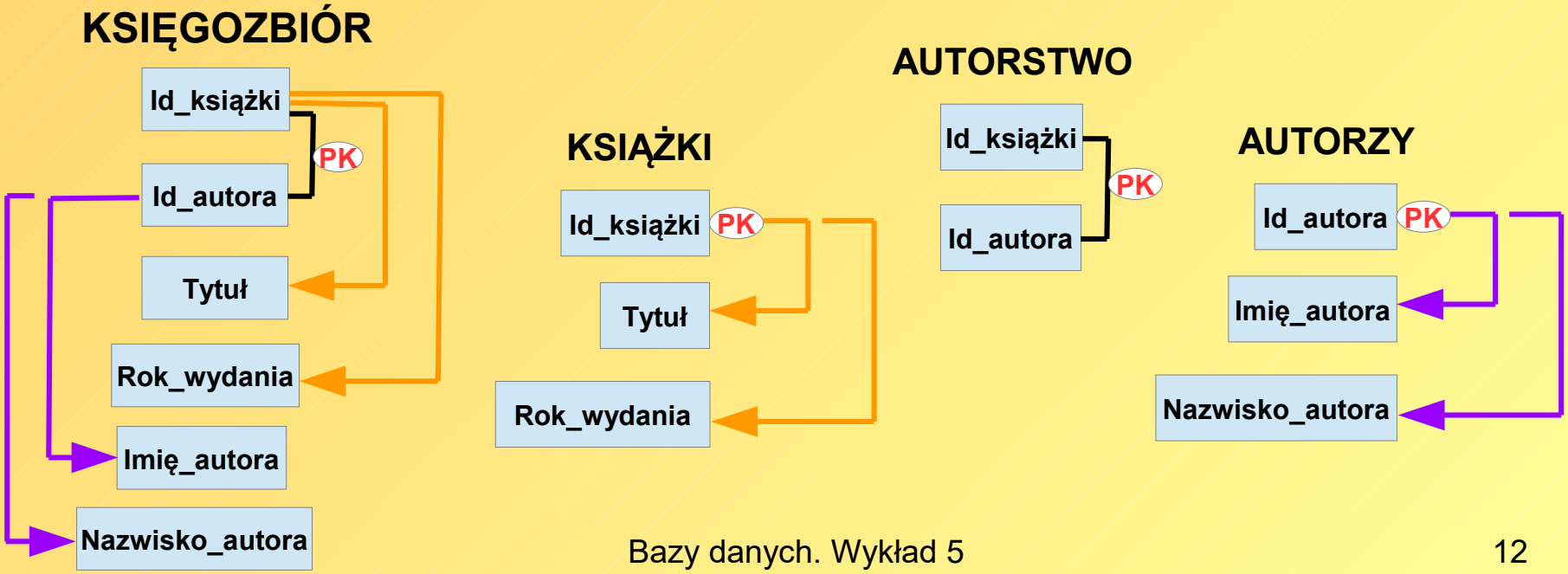
Dla dziedziny rzeczywistości, jaką są książki i ich autorzy, przyjęliśmy założenia:

(1) Relacja KSIĘGOZBIÓR o nagłówku N(KSIĘGOZBIÓR) = {Id_książki, Tytuł, Rok_wydania, Id_autora, Imię_autora, Nazwisko_autora} jest w 1NF.

(2) Książka może mieć wielu autorów.

(3) Autor mógł napisać wiele książek.

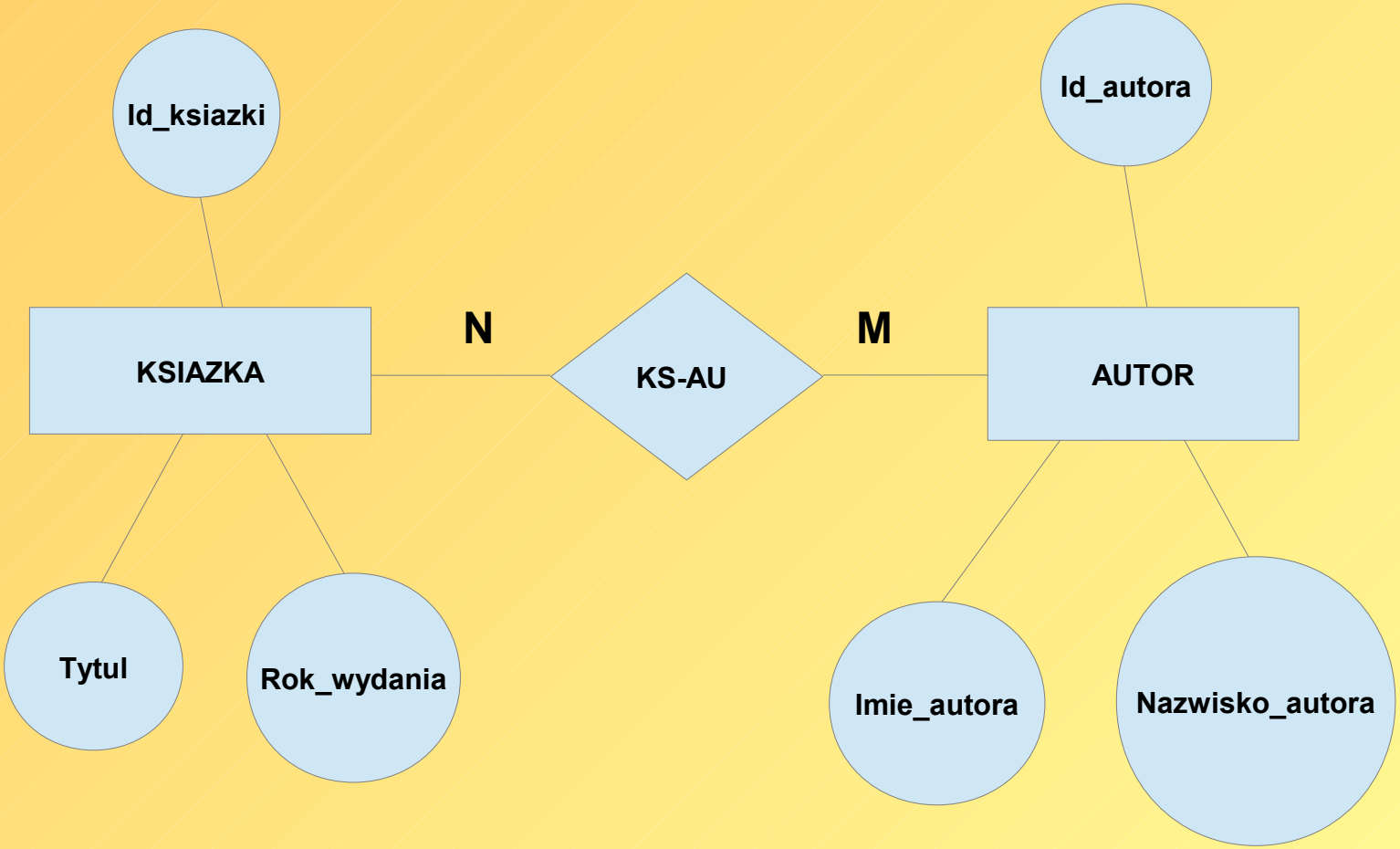
Wówczas PK(KSIĘGOZBIÓR)={ Id_książki, Id_autora }



Model logiczny – koncepcyjny (konceptualny) oraz pełny

na przykładzie modelu związków encji (*ERD – Entity Relationship Diagram)*

- Przykład związku binarnego typu N:M w notacji Chena – model logiczny pełny



Model logiczny – koncepcyjny (konceptualny) oraz pełny

na przykładzie modelu związków encji (*ERD – Entity Relationship Diagram)*

W szkicach projektów wygodnie jest **stosować zmodyfikowaną notację Chena**:

- Symbol encji – prostokąt z nazwą encji nad jego górną krawędzią
- Symbol atrybutu encji – nie stosuje się:

  - atrybuty danej encji są wypisane w prostokącie reprezentującym tę encję

  - dla wygody warto jawnie podać atrybuty kluczy obcych
- Symbol związku między encjami – romb
  jednak opcjonalnie można pominąć romb, poprzestając na linii łączącej encje i/lub strzałką zakończyć końcówkę linii, wskazującą klucz odniesienia (klucz główny) dla klucza obcego definiowanego przez ten związek)
- Symbole kardynalności końcówki związku między encjami:

  - jeden: 1

  - wiele: N bądź M
- Symbole kardynalności związków:

  - jeden do jeden 1 : 1

  - jeden do wiele 1 : N

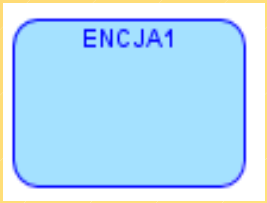
  - wiele do wiele N : M

Model logiczny – koncepcyjny (konceptualny) oraz pełny

na przykładzie modelu związków encji (*ERD – Entity Relationship Diagram)*

Notacja Barkera (stosowana w narzędziach Oracle):

- Symbol encji bez atrybutów



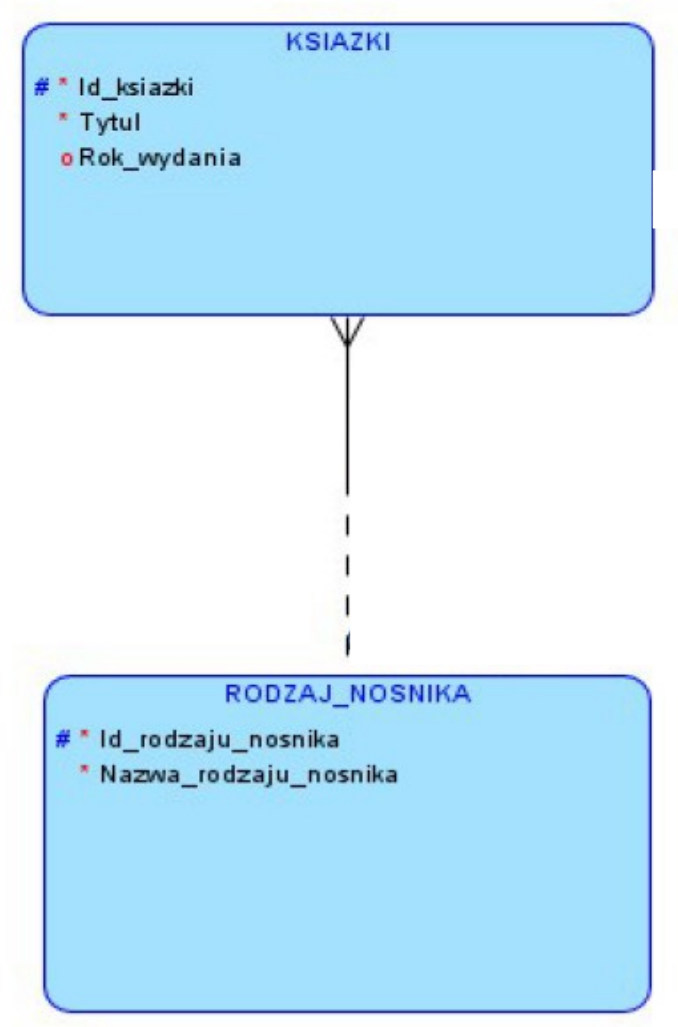
- Symbol encji z atrybutami

- Symbole związków między encjami
- Symbole kardynalności końcówki związku między encjami:
  - jeden: —
  - wiele: ⩽
- Symbole przynależności końcówki związku między encjami:
  - opcjonalny: — — — —
  - obowiązkowy: ————————
- Symbol identyfikującej własności końcówki między encjami: |

Model logiczny – koncepcyjny (konceptualny) oraz pełny

na przykładzie modelu związków encji (*ERD – Entity Relationship Diagram)*

- Przykład związku binarnego typu 1:N w notacji Barkera – model logiczny pełny

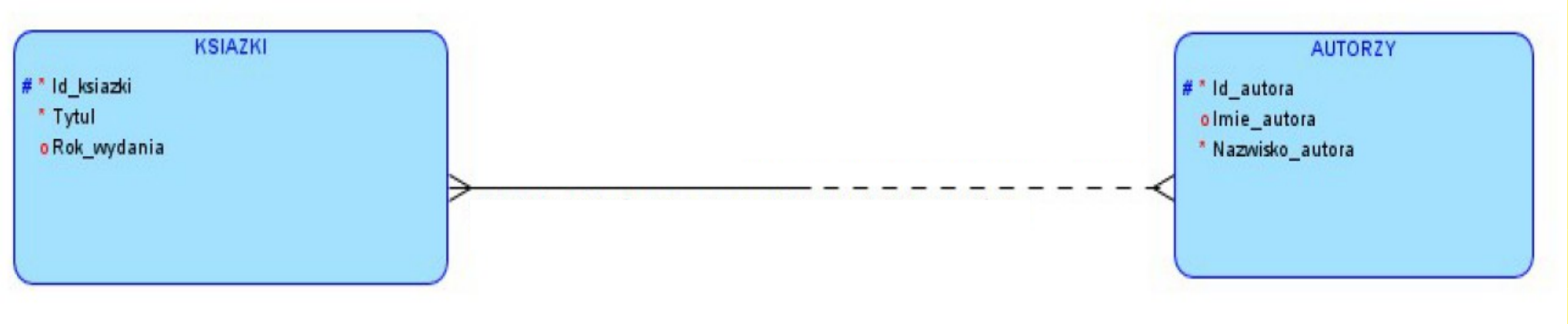


Związek RODZAJ_NOSNIKA – KSIAZKI
jest typu jeden do wiele,
opcjonalny po stronie encji RODZAJ_NOSNIKA
i obowiązkowy po stronie encji KSIAZKI

Model logiczny – koncepcyjny (konceptualny) oraz pełny

na przykładzie modelu związków encji (*ERD – Entity Relationship Diagram)*

- Przykład związku binarnego typu N:M w notacji Barkera – model logiczny pełny



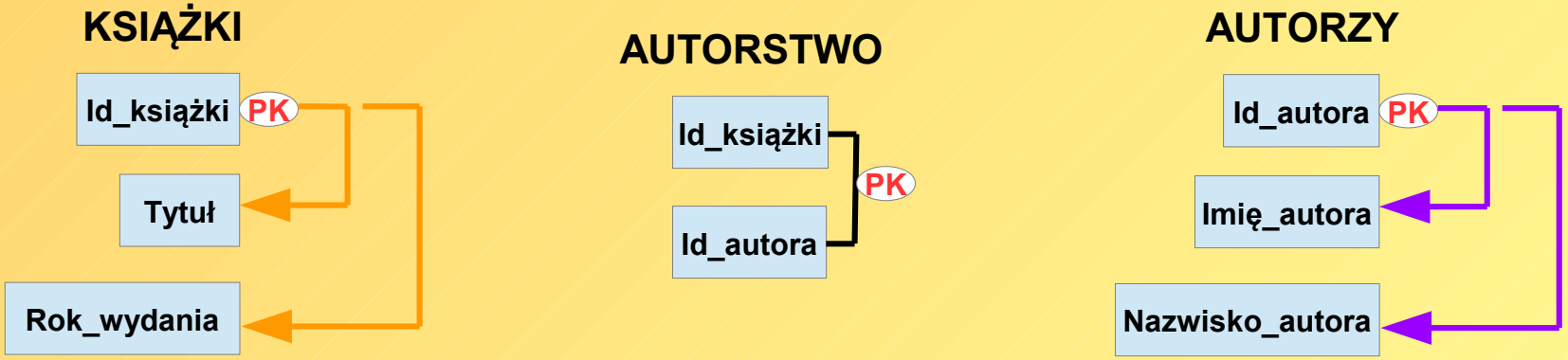
Związek AUTORZY – KSIAZKI jest typu wiele do wiele,
opcjonalny po stronie encji AUTORZY
i obowiązkowy po stronie encji KSIAZKI

Model fizyczny – implementacyjny

na przykładzie modelu relacyjnego wygenerowanego z modelu logicznego w narzędziu Oracle SQL Developer Data Modeler

- **Przykład związku** binarnego KSIĄŻKI – AUTORZY **typu wiele do wiele sprowadzonego do postaci z encją słabą** AUTORSTWO (reprezentującą kardynalność tego związku)

  Przypomnijmy relacje powstałe po dekompozycji relacji KSIĘGOZBIÓR:



Zrzuty ekranu dostępne w pliku ksiegozbiorKSATSAU.pdfpokazują szczegóły pracy w module Data Modeler od modelu logicznego poprzez model relacyjny (fizyczny) do wygenerowania skryptu DDL, a następnie jego implementacji na serwerze bazy danych.

Model fizyczny – implementacyjny

na przykładzie modelu relacyjnego wygenerowanego z modelu logicznego w narzędziu Oracle SQL Developer Data Modeler

- **Przykład związku** binarnego RODZAJ_NOŚNIKA – KSIĄŻKI **typu jeden do wiele** przedstawionego wcześniej w notacji Chena

Zrzuty ekranu dostępne w pliku ksiegozbiorPELNY.pdf pokazują szczegóły pracy w module Data Modeler od modelu logicznego poprzez model relacyjny (fizyczny) do wygenerowania skryptu DDL.

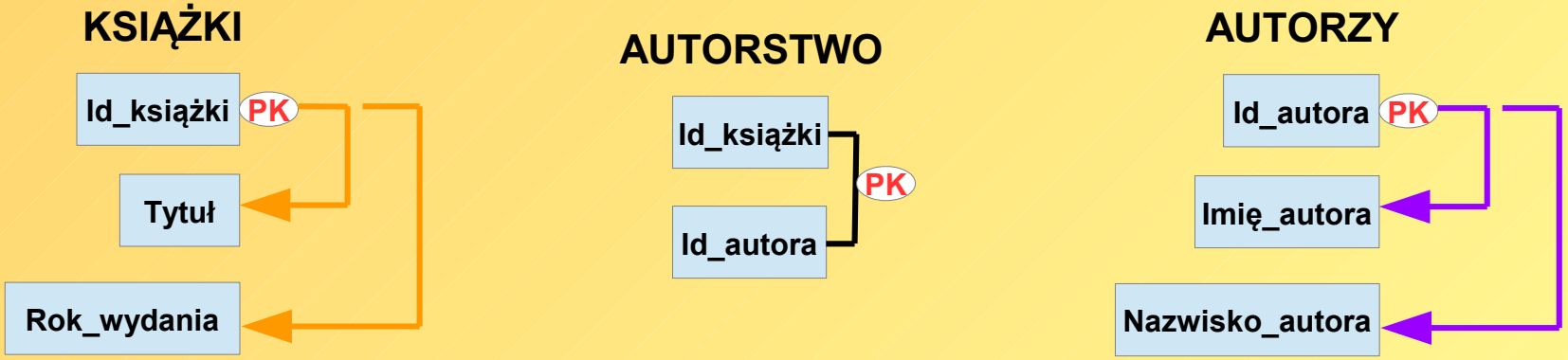
Projektowanie jest kontynuowane w modelu logicznym, zawierającym już związek KSIĄŻKI – AUTORZY z encją słabą AUTORSTWO.

Model fizyczny – implementacyjny

na przykładzie modelu relacyjnego wygenerowanego z modelu logicznego w narzędziu Oracle SQL Developer Data Modeler

- **Przykład związku** binarnego KSIĄŻKI – AUTORZY **typu wiele do wiele w czystej formie**, tj. **bez jawnego wystąpienia encji słabej** AUTORSTWO w modelu logicznym

  Przypomnijmy relacje powstałe po dekompozycji relacji KSIĘGOZBIÓR:



Zrzuty ekranu dostępne w pliku ksiegozbiorRB.pdf
pokazują szczegóły pracy w module Data Modeler od modelu logicznego poprzez model relacyjny (fizyczny) do wygenerowania skryptu DDL.
Na zrzutach przedstawiających model logiczny pokazane zostały w pełnej formie **reguły biznesowe** jako etykiety końcówek związków między encjami.

Model fizyczny – implementacyjny

na przykładzie modelu relacyjnego wygenerowanego z modelu logicznego w narzędziu Oracle SQL Developer Data Modeler

- Uzupełnienie modelu logicznego zawierającego już związek binarny KSIĄŻKI – AUTORZY typu wiele do wiele (bez encji słabej) o związek binarny RODZAJ_NOŚNIKA – KSIĄŻKI typu jeden do wiele z podaniem pełnych reguł biznesowych jako etykiet końcówek związku.

Zrzuty ekranu dostępne w pliku ksiegozbiorRBkontynuacja.pdf pokazują szczegóły pracy w module Data Modeler od uzupełnienia modelu logicznego poprzez aktualizację (ponowne utworzenie) modelu relacyjnego (fizycznego) do wygenerowania skryptu DDL.

Inżynieria odwrotna w relacyjnym modelowaniu danych

za pomocą narzędzia Oracle SQL Developer Data Modeler

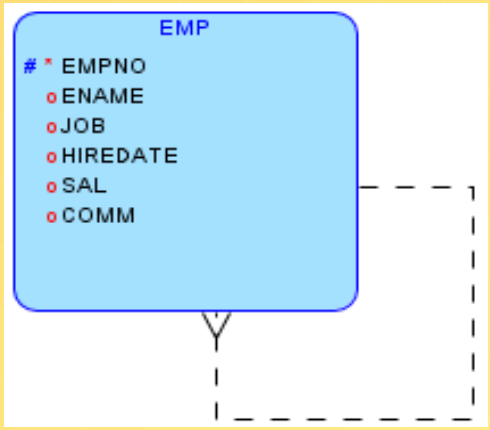
- Utworzenie modelu relacyjnego a następnie logicznego na podstawie schematu użytkownika istniejącego w bazie danych

Zrzuty ekranu dostępne w pliku scott_inzynieria_odwrotna.pdf pokazują szczegóły pracy w module Data Modeler od importu ze słownika danych bazy danych schematu użytkownika a15 i utworzenie na jego podstawie modelu relacyjnego, a następnie utworzenie modelu logicznego na podstawie modelu relacyjnego (fizycznego).

Związek unarny (binarny rekursywny) w relacyjnym modelowaniu danych

Zastosowanie w Oracle SQL Developer Data Modeler

- Uzupełnienie modelu logicznego wygenerowanego poprzednio na podstawie schematu użytkownika istniejącego w bazie danych o rekursywny związek binarny typu 1:N zastosowany dla encji EMP.

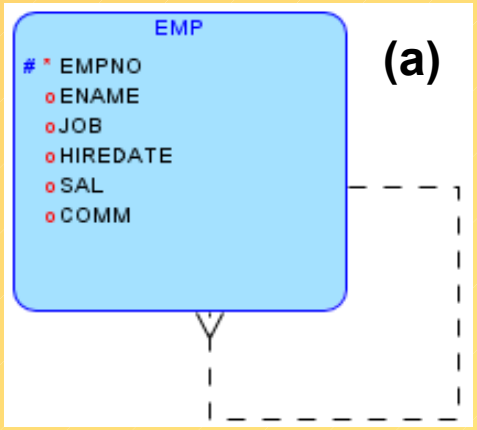


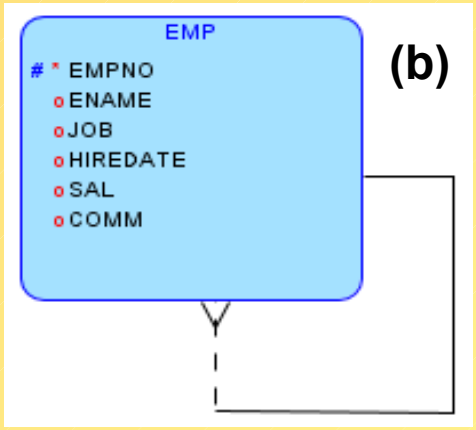
Zrzuty ekranu dostępne w pliku scottBR.pdf
pokazują szczegóły pracy w module Data Modeler od importu utworzone wcześniej projektu na podstawie istniejącego w bazie danych schematu użytkownika a15 i uzupełnienia modelu logicznego o rekursywny związek binarny EMP-EMP typu 1:N po wygenerowanie z rozbudowanego modelu relacyjnego nowego modelu relacyjnego (fizycznego), a z niego nowego skryptu DDL.

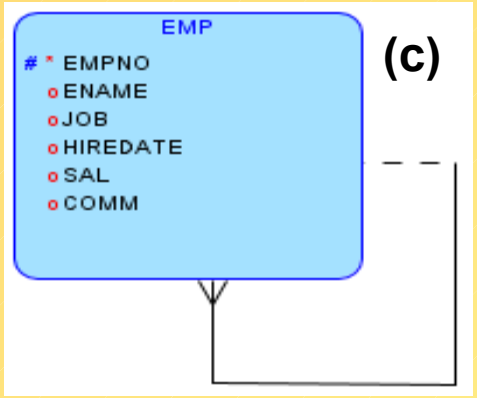
Związek unarny (binarny rekursywny) w relacyjnym modelowaniu danych

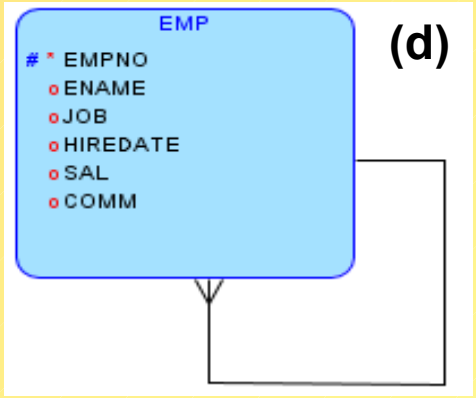
Zastosowanie w Oracle SQL Developer Data Modeler

- Problem do rozważenia wraz z uzasadnieniem odpowiedzi:
  Które z pokazanych poniżej związków są dozwolone w modelu logicznym?









Związki typu 1:1 w relacyjnym modelowaniu danych

Zastosowanie w Oracle SQL Developer Data Modeler

- Uzupełnienie poprzedniego modelu logicznego o **związek binarny typu 1:1** między encjami EMP i DEPT.

Zrzuty ekranu dostępne w pliku scottBRSZD.pdf
pokazują szczegóły pracy w module Data Modeler od importu utworzone poprzednio projektu i uzupełnienia modelu logicznego o związek binarny EMP-DEPT typu 1:1 po wygenerowanie z rozbudowanego modelu logicznego nowego modelu relacyjnego (fizycznego), a z niego nowego skryptu DDL.

Związki typu 1:1 w relacyjnym modelowaniu danych

Zastosowanie w Oracle SQL Developer Data Modeler

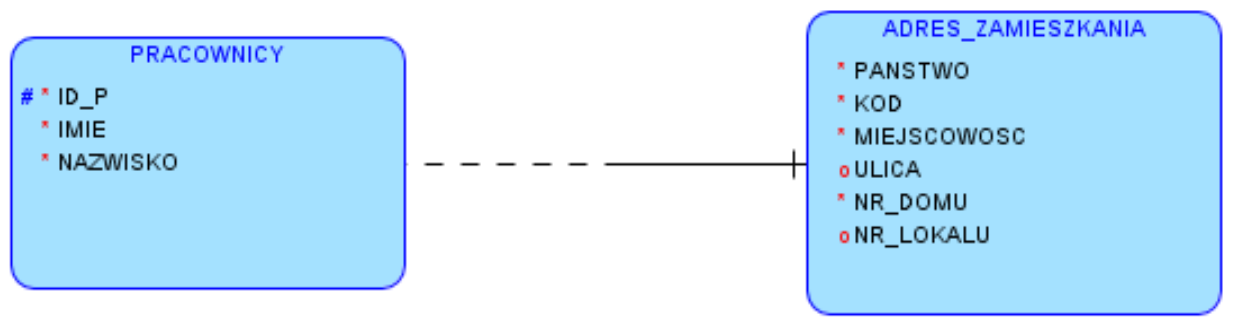
- Prawidłowy w modelu logicznym z koncepcyjnego punktu widzenia związek binarny typu 1:1 między encjami EMP i DEPT, który jest obowiązkowy po stronie encji DEPT i opcjonalny po stronie encji EMP, generuje jednak więzy w modelu relacyjnym (fizycznym), jakie stwarzają utrudnienia w obsłudze tabel EMP i DEPT zaimplementowanych na serwerze bazy danych, jeśli jednocześnie występuje między encjami DEPT i EMP związek binarny 1:N, który oryginalnie jest implementowany w edukacyjnym schemacie użytkownika SCOTT.

  Proszę sprawdzić, co daje poniżej przedstawiony dla modelu logicznego układ związków między encjami EMP i DEPT po wygenerowaniu na jego podstawie modelu relacyjnego (fizycznego).

Związki typu 1:1 w relacyjnym modelowaniu danych

Zastosowanie w Oracle SQL Developer Data Modeler

- Zastosowanie **identyfikującego związku binarnego typu 1:1** między encjami PRACOWNICY i ADRESY_ZAMIESZKANIA.



Zrzuty ekranu dostępne w pliku pracownicy_adresy.pdf
pokazują szczegóły pracy w module Data Modeler od modelu logicznego z encjami PRACOWNICY i ADRESY_ZAMIESZKANIA, między którymi zdefiniowano identyfikujący związek typu 1:1, do wygenerowania z modelu logicznego modelu relacyjnego (fizycznego), a z niego skryptu DDL.

Hierarchie encji w relacyjnym modelowaniu danych

Zastosowanie w Oracle SQL Developer Data Modeler

- Hierarchia encji na przykładzie poprzednio omówionych encji PRACOWNICY i ADRESY_ZAMIESZKANIA, przedstawionych **w** nowym **modelu logicznym** jako **nadencja** (nadtyp, supertyp) **PRACOWNICY** i jej **podencja** (podtyp) **ADRESY_ZAMIESZKANIA**, które **w modelu relacyjnym prowadzą do identyfikującego związku binarnego typu 1:1** między relacjami PRACOWNICY i ADRESY_ZAMIESZKANIA.

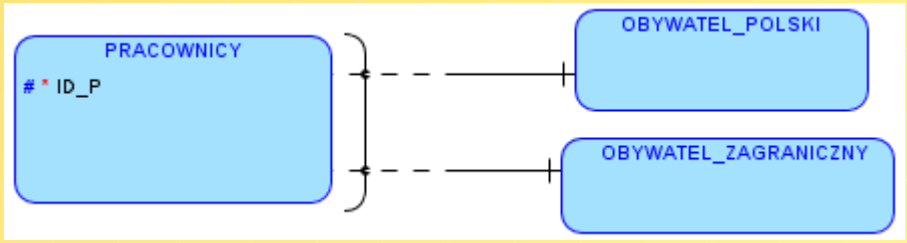
Zrzuty ekranu dostępne w pliku pracownicy_adresy_hierarchia.pdf pokazują szczegóły pracy w module Data Modeler od modelu logicznego z nadencją PRACOWNICY i jej podencją ADRESY_ZAMIESZKANIA poprzez wygenerowanie z modelu logicznego modelu relacyjnego (fizycznego), a z niego skryptu DDL.

Hierarchie encji w relacyjnym modelowaniu danych

Zastosowanie w Oracle SQL Developer Data Modeler

- Hierarchia encji rozpatrywana
  w modelu logicznym z punktu widzenia
  encji generalizacji – nadencji (nadtypu, supertypu)
  PRACOWNICY o wspólnych atrybutach
  dla dwóch **encji specjalizacji – podencji** (podtypów)
  OBYWATEL_POLSKI i OBYWATEL_ZAGRANICZNY
  o atrybutach specyficznych tylko dla siebie.
  Ta hierarchia prowadzi w modelu relacyjnym
  do **związków wyłącznych** (*exclusive relationships*)
  między relacjami specjalizacji a relacją generalizacji.

- **Związki wyłączne** w modelu logicznym charakteryzują się tym, że **dane wystąpienie encji może wchodzić tylko w jeden ze związków**.



Hierarchie encji w relacyjnym modelowaniu danych

Zastosowanie w Oracle SQL Developer Data Modeler

- Związki wyłączne znajdują interpretację w przypadku zdefiniowanej hierarchii encji znajdują w tym, że dane wystąpienie encji generalizacji może wchodzić w związek tylko z jedną z encji specjalizacji.

W ogólności hierarchia encji posiada następujące własności:

(1) Każde wystąpienie encji generalizacji – nadencji wiąże się zawsze z jednym wystąpieniem encji specjalizacji – podencji.

(2) Encje specjalizacji – podencje dziedziczą wszystkie atrybuty swojej encji generalizacji – nadencji.

(3) Klucz encji generalizacji – nadencji jest wspólny dla jej wszystkich encji specjalizacji – podencji.

(4) Zatem każde wystąpienie encji specjalizacji – podencji jest wystąpieniem także jej encji generalizacji – nadencji.

Zrzuty ekranu dostępne w pliku pracownicy_obywatelstwo.pdf pokazują szczegóły pracy w module Data Modeler od modelu logicznego z encją generalizacji – nadencją PRACOWNICY oraz jej encjami specjalizacji – podencjami OBYWATEL_POLSKI i OBYWATEL_ZAGRANICZNY poprzez wygenerowanie z modelu logicznego modelu relacyjnego (fizycznego), a z niego skryptu DDL.

Reguły Codda relacyjnego modelowania danych (w formie 12 zasad + zerowa zasada podstawowa)

(0) Reguły 1 – 12 dotyczą dowolnego systemu baz danych, w którym dane składowane są wyłącznie zgodnie z relacyjnym modelem danych.

(1) Reguła informacyjna: Dane składowane w bazie danych, zarówno dane użytkowników, jak i metadane, muszą być wartościami pojedynczych komórek tabel (tzn. jako atomowe wartości atrybutów w każdej krotce relacji, stanowiącej matematyczną reprezentację danej tabeli). Upraszczając, wszystko w bazie danych musi być składowane w tabelach.

(2) Reguła gwarantowanego dostępu: Każda pojedyncza wartość atrybutu (w każdej krotce) ma być rozpoznawana (dostępna) logicznie za pomocą kombinacji: nazwa tabeli, wartość podkrotki klucza głównego i nazwa atrybutu, którego wartość chcemy zidentyfikować.

(3) Reguła systematycznego traktowania wartości pustej (NULL): Wartość NULL musi być traktowana systematycznie i jednakowo (we wszystkich bazach relacyjnych) jako brak wartości, nieznana wartość lub wartość nie znajdująca zastosowania (nieadekwatna).

Reguły Codda relacyjnego modelowania danych (w formie 12 zasad + zerowa zasada podstawowa)

(4) Reguła dotycząca aktywnego online katalogu (tzn. słownika bazy danych): Opis struktury całej bazy danych musi być składowany w katalogu, zwanym słownikiem (bazy) danych, który jest dostępny online dla autoryzowanych użytkowników, korzystających do wydobycia informacji z katalogu tego samego języka zapytań, jaki umożliwia wydobycie (eksplorację) wszystkich innych danych z bazy danych.

(5) Kompleksowa reguła dotycząca języka danych: Dane składowane w bazie danych mogą być dostępne jedynie za pomocą języka o liniowej składni, który umożliwia definiowanie danych (tzn. struktur, w których są składowane), manipulowanie danymi i operacje zarządzania transakcjami w bazie danych. Język danych może być stosowany zarówno bezpośrednio w bazie danych, jak i za pomocą aplikacji. Gdy baza danych zezwala na dostęp do danych bez wykorzystania tego języka, traktowane jest to jako naruszenie dostępu do bazy danych.

Uwaga dotycząca języków o liniowej składni: Składnia liniowa oznacza możliwość pisania kodu bez użycia znaków końca linii lub znaku powrotu karetki. Chociaż użycie takich znaków jest zalecane dla czytelności kodu,

Reguły Codda relacyjnego modelowania danych (w formie 12 zasad + zerowa zasada podstawowa)

są one opcjonalne, ponieważ kompilatory pomijają je w analizie i kompilacji kodu. HTML, C i SQL to przykłady języków, które posiadają składnię liniową; wszystkie one wykorzystują przecinki, średniki i nawiasy w celu oddzielenia bloków kodu. Składnię liniową można analizować od lewej do prawej.

(6) Reguła aktualizacji perspektyw: System bazy danych musi umożliwiać aktualizację wszystkich perspektyw w bazie danych, które zgodnie z teorią mogą być aktualizowane.

(7) Reguła modyfikowalności danych (wysokopoziomowych operacji *insert*, *update* i *delete*): Baza danych musi umożliwiać (wysokopoziomowe) wstawianie, aktualizację i usuwanie danych. Te operacje (transakcje) nie mogą być ograniczone do pojedynczych wierszy, tzn. muszą także wspierać unię, przecięcie i różnicę, aby dawać zbiory rekordów danych.

Reguły Codda relacyjnego modelowania danych (w formie 12 zasad + zerowa zasada podstawowa)

(8) Reguła fizycznej niezależności danych: Dane składowane w bazie danych muszą być niezależne od aplikacji, która ma dostęp do bazy danych. Żadna zmiana w fizycznej strukturze bazy danych (tj. w fizycznych modelach danych, w organizacji struktury składowania czy w urządzeniach składowania danych) nie może mieć jakiegokolwiek wpływu na modele logiczne (koncepcyjne) schematów bazy danych, a także na dostępność danych dla zewnętrznych aplikacji.

(9) Reguła logicznej niezależności danych: Logika (model logiczny) danych w określonym schemacie bazy danych musi być niezależny z punktu widzenia użytkownika aplikacji. Jakakolwiek zmiana logiki (modelu logicznego) danych (określonego schematu) w bazie danych nie może wpływać na aplikacje, korzystające z tych danych. *To jedna z najtrudniejszych reguł do spełnienia!*

Reguły Codda relacyjnego modelowania danych (w formie 12 zasad + zerowa zasada podstawowa)

(10) Reguła niezależności integralności danych: W bazie danych wszystkie więzy integralności danych mogą być modyfikowane niezależnie od aplikacji, korzystającej z danych składowanych w bazie danych, tzn. bez potrzeby jakichkolwiek zmian w aplikacji. Ta reguła czyni bazę danych niezależną od *fasady* aplikacji (tzw. części *front-end* aplikacji) i jej interfejsu.

(11) Reguła niezależności dystrybucyjnej: Użytkownik końcowy nie może dostrzegać dystrybucji (rozproszenia, rozłożenia) danych w różnych lokalizacjach. Użytkownicy powinni zawsze mieć wrażenie, że dane są składowane tylko w jednej lokalizacji. Ta reguła została uznana za podstawę **rozproszonych systemów baz danych**.

(12) Reguła zakazująca *działalności wywrotowej* (*non-subversion rule*): Jeżeli system baz danych posiada interfejs, dający dostęp niskopoziomowy do rekordów danych (tzn. do operacji na pojedynczych rekordach), to taki interfejs bazy danych nie może dopuszczać do uszkodzenia systemu bazy danych, obejścia mechanizmów zabezpieczeń ani do ominięcia więzów integralności danych.