Utrwalanie danych – zastosowanie obiektowego modelu danych warstwy biznesowej do generowania schematu relacyjnej bazy danych

Informacje wstępne

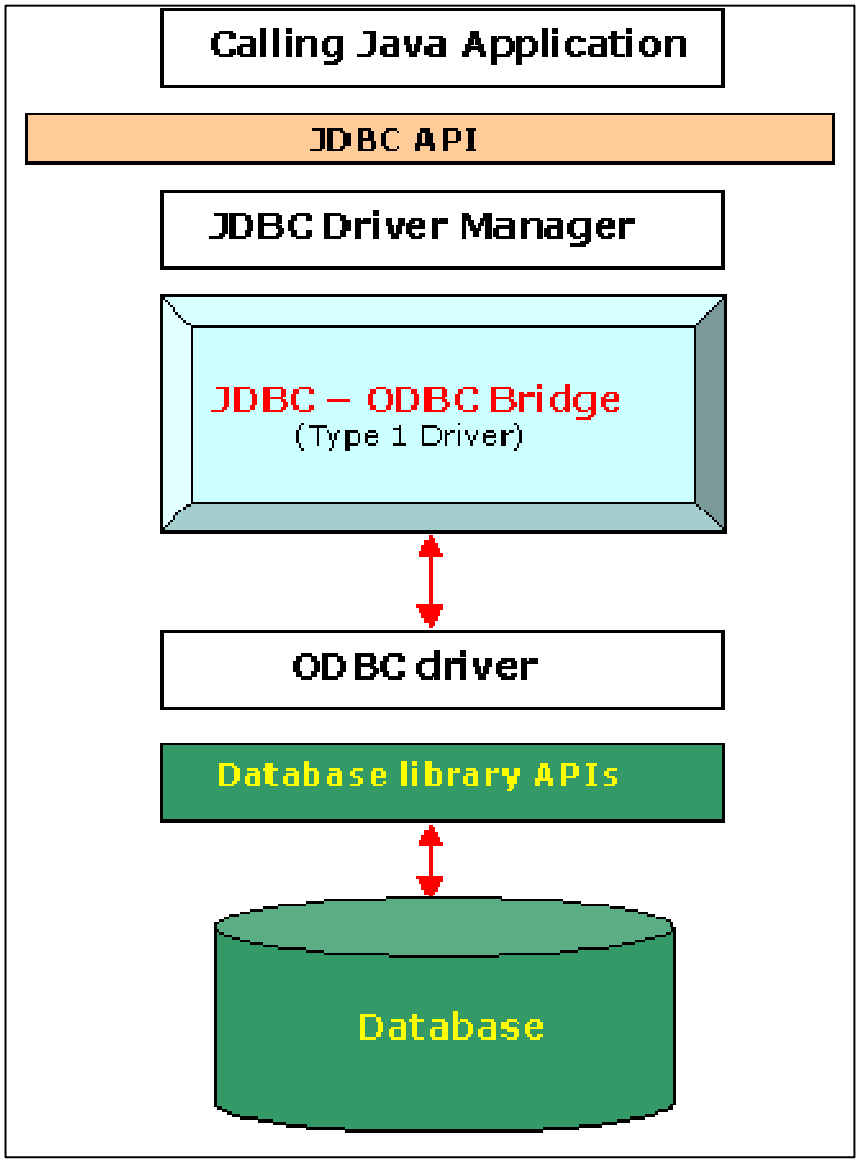
Autor

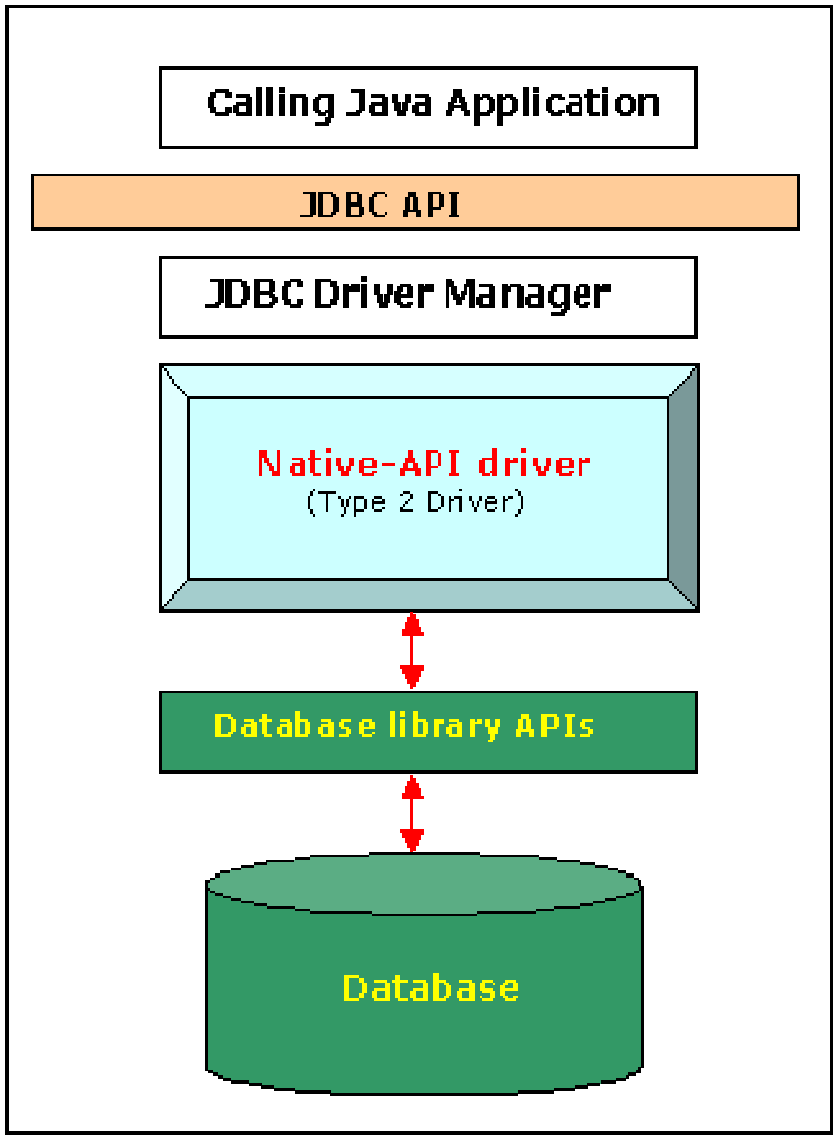
Zofia Kruczkiewicz

Wzorce oprogramowania 4

1. Relacyjne bazy danych udostępniane w programach Javy na platformach SE (Standard Edition) i EE (Enterprise Editon) – sterowniki typu JDBC

Sterowniki JDBC: 1 i 2 typ

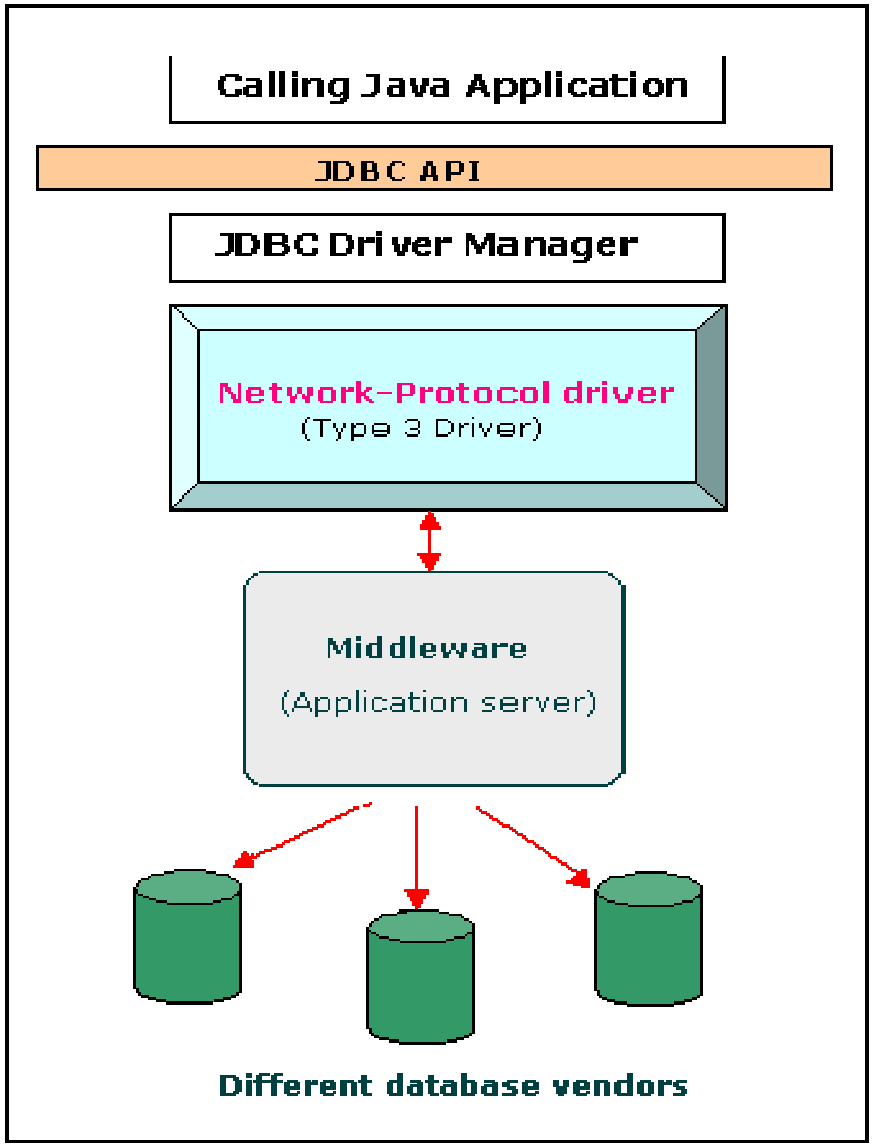


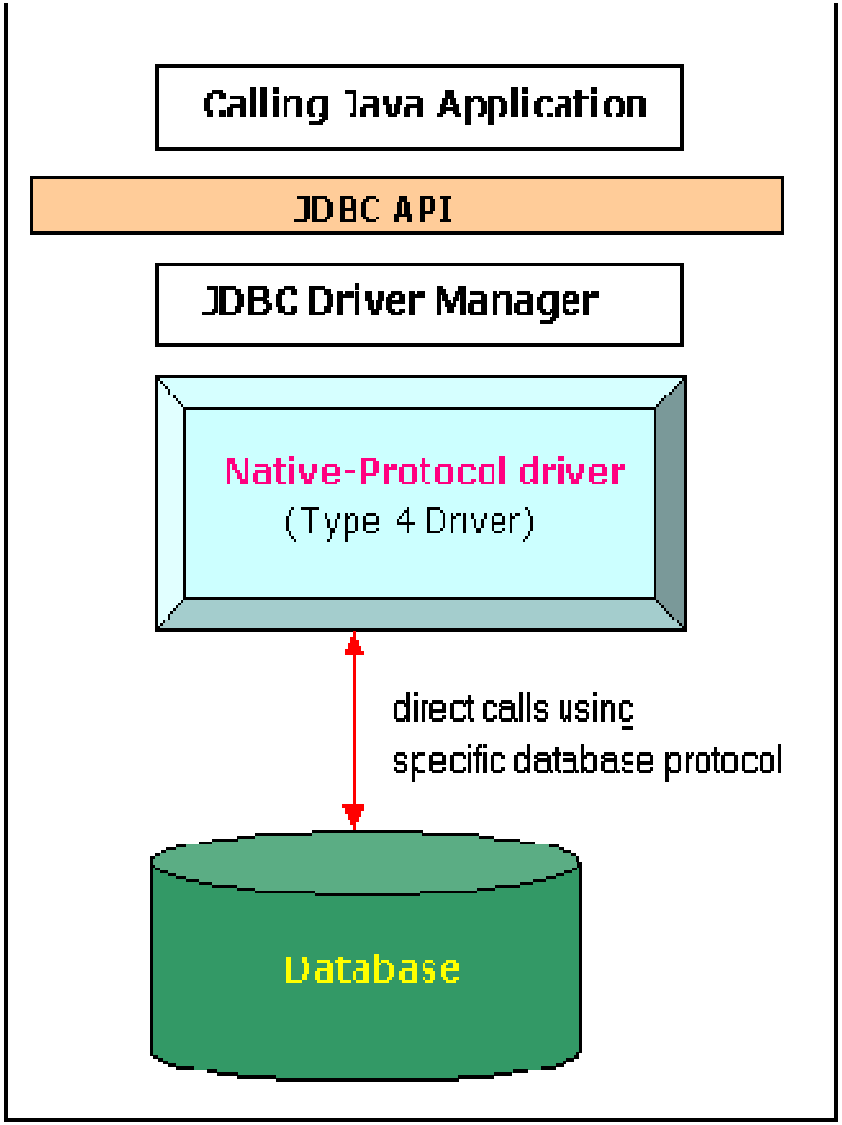


en.wikipedia.org/wiki/JDBC_type_1_driver

en.wikipedia.org/wiki/JDBC_type_2_driver

Sterowniki JDBC: 3 i 4 typ

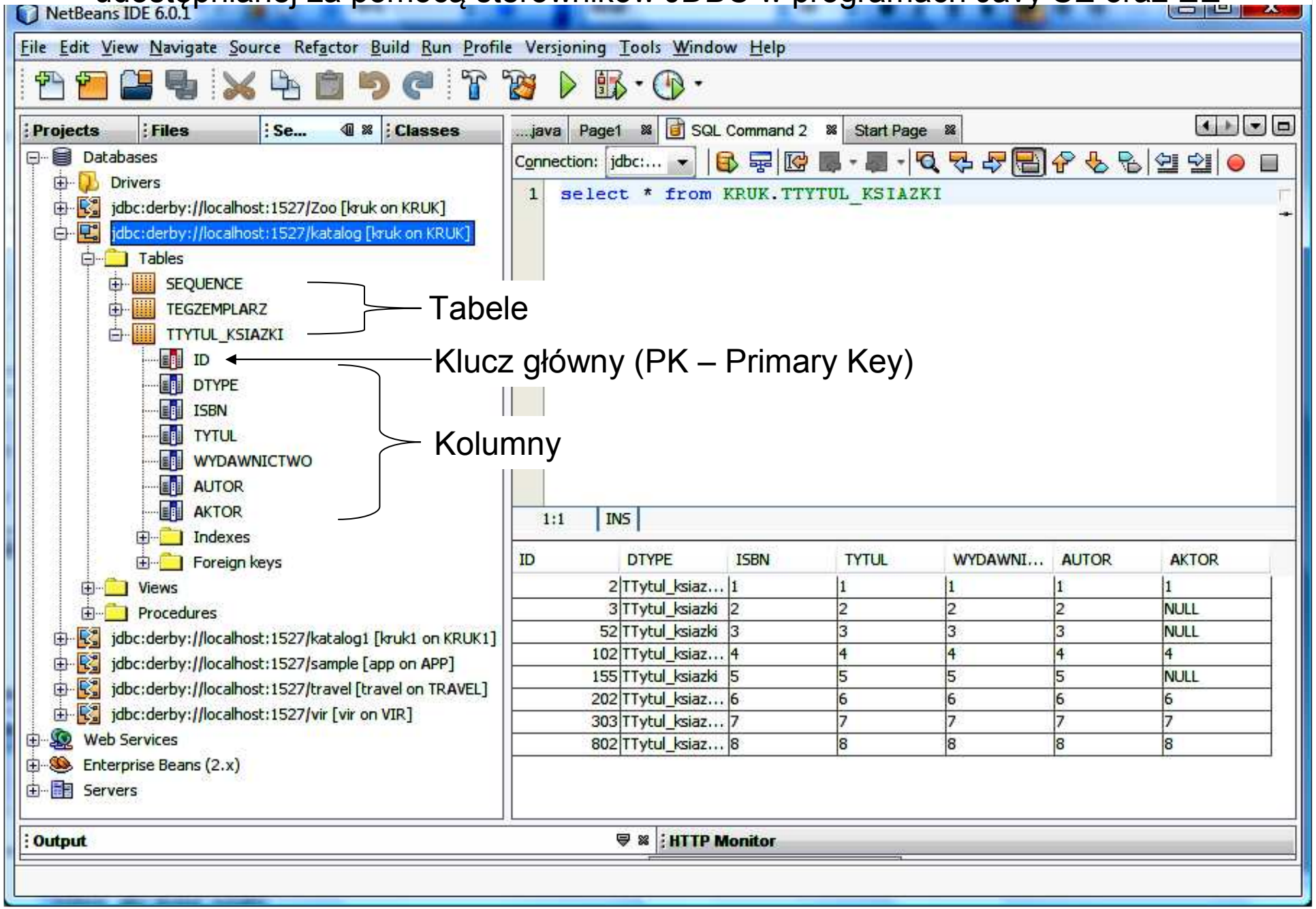




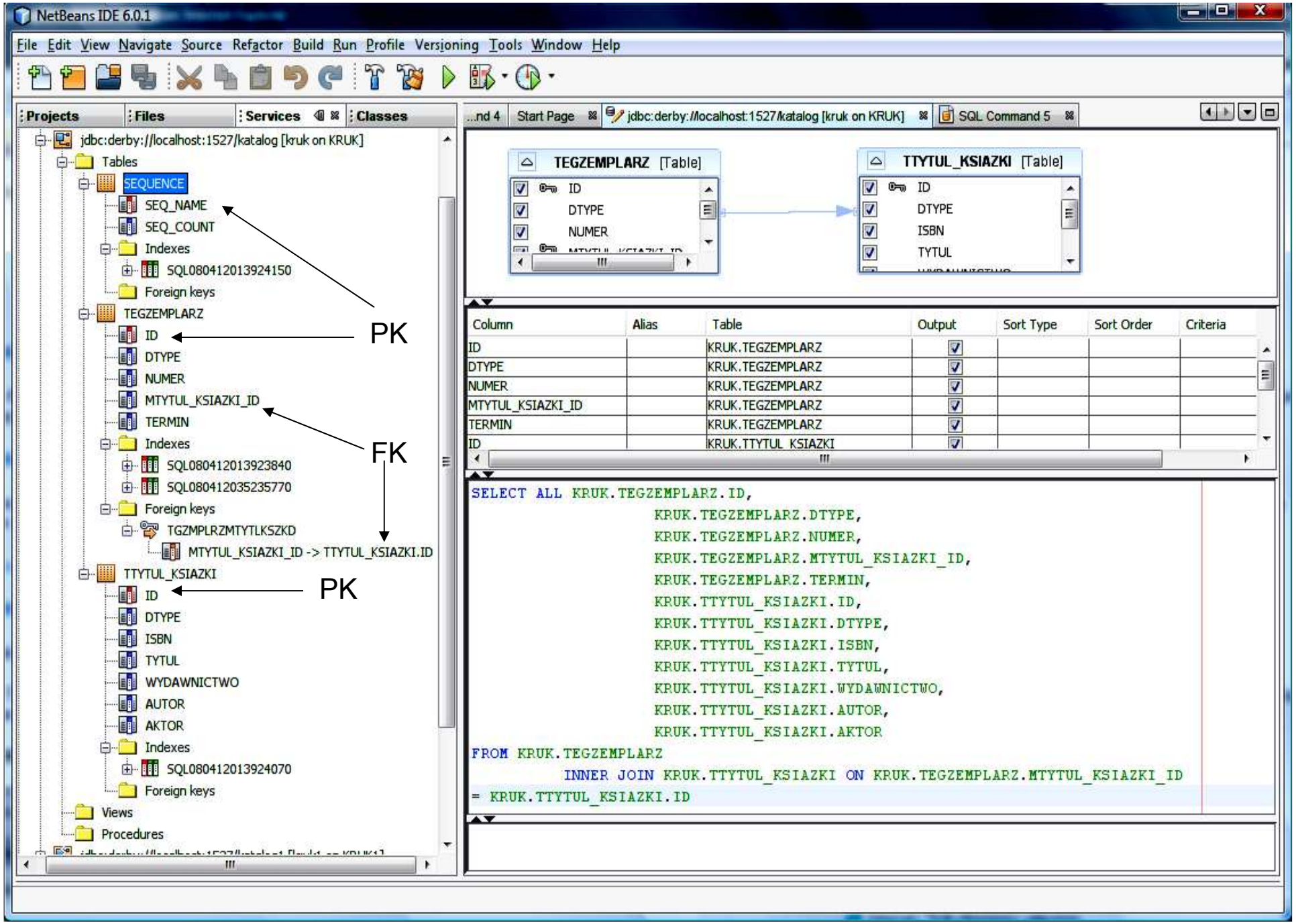
en.wikipedia.org/wiki/JDBC_type_3_driver

en.wikipedia.org/wiki/JDBC_type_4_driver

jdbc:derby://localhost:1527/katalog [kruk on KRUK] – przykład relacyjnej bazy danych w udostępnianej za pomocą sterowników JDBC w programach Javy SE oraz EE



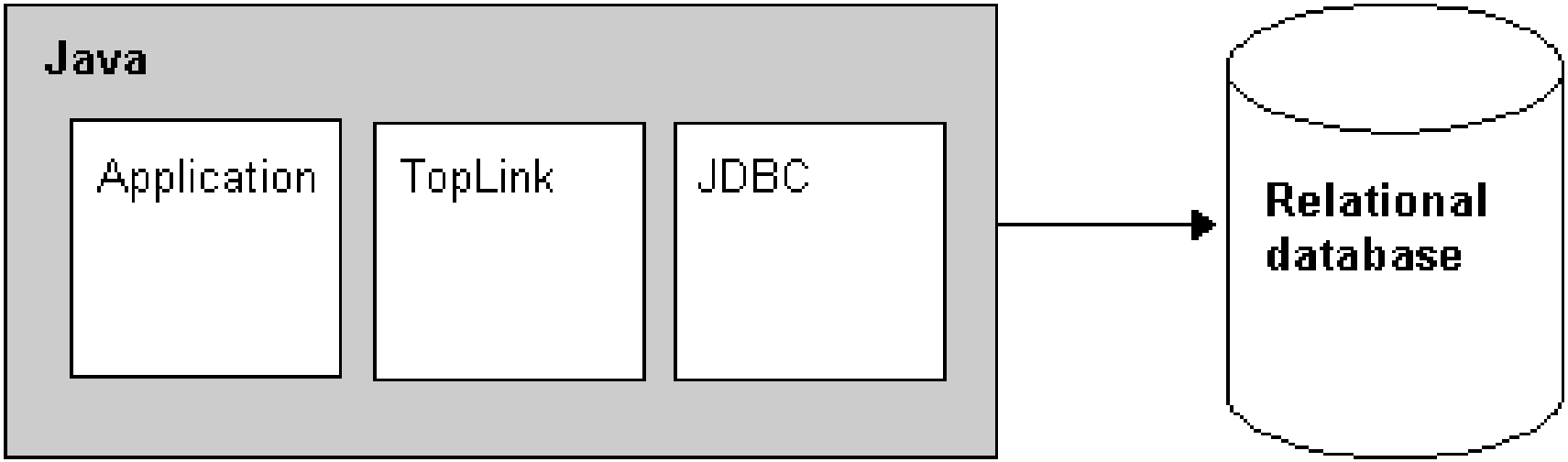
Klucze główne i obce – Primary Key (PK), Foregin Key (FK)



2. Obsługa relacyjnych baz danych za pomocą odwzorowania obiektowego modelu danych na model relacyjny. Zastosowanie technologii TopLink (wprowadzenie)

wg. Oracle9*i*AS TopLink Getting Started Release 2 (9.0.3)

Mapowanie obiektów do relacyjnej lub nierelacyjnej bazy danych



- Wzorzec projektowy typu **Domain Store** zastosowany w technologii TopLink chroni aplikację przed zmianami w bazie danych – ogranicza zmiany w aplikacji
- Wprowadzono zbiór metod przechowywania i przeszukiwania danych. wprowadzono standard metod dostępu w klasach aplikacji – niezależny od typu klasy

Zalety mapowania obiektów

- Redukcja kosztu projektowania aplikacji bazodanowej - projektując model obiektowy w warstwie biznesowej projektuje się jednocześnie schemat relacyjnej lub obiektowej bazy danych
- Uzyskanie znacznej poprawy wydajności i skalowalności aplikacji
- Uzyskuje się niezależny od platformy czysty kod Javy działający w dowolnym środowisku Javy lub serwera aplikacji Javy
- Poziom danych i operowanie na tych danych są niezależne od rodzaju bazy danych
- integracja z najważniejszymi technologiami Javy - EJB, XML, JTS, CORBA, RMI.

Java Object Model

Obiekt Javy zawiera następujące elementy:

- Atrybuty nieobiektowe (np. int, float) i obiektowe (np.String, Date)
- Powiązania jako referencje do innych klas przechowywane przy mapowaniu jako odpowiednie deskryptory
- Metody charakteryzują się zachowaniem, a nie stanem, więc nie są przechowywane w bazie danych.

Podobieństwa między modelem obiektowym i relacyjnym

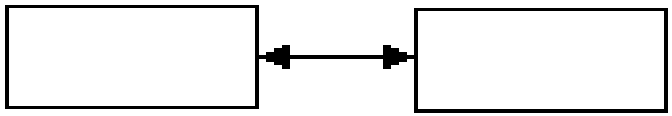
- **Klasy** definiują strukturę obiektów w taki sam sposób, jak tabele definiują strukturę rekordów
- **Obiekty** przechowują dane w atrbutach w taki sam sposób, w jaki rekordy przechowują w polach
- Tak jak rekordy w jednej tabeli są powiązane z rekordami innej tabeli za pomocą **kluczy obcych**, tak obiekty są powiązane za pomocą **referencji** do innych obiektów,
- W językach zorientowanych obiektowo takich jak Java, **atrybuty mają charakter statyczny** - również **pola w rekordach mają też charakter statyczny**

Przechowywanie obiektów w relacyjnej bazie danych

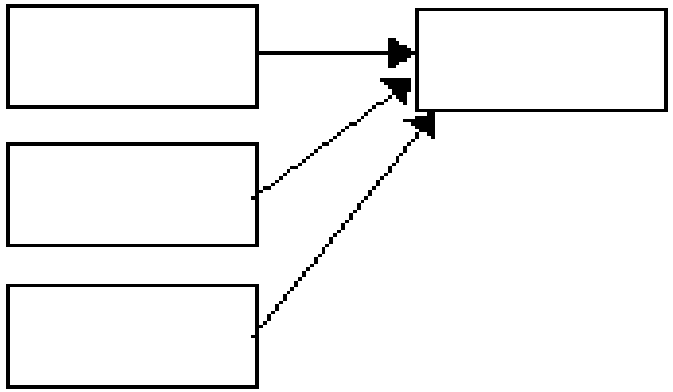
Trudności w wyrażaniu pewnych mechanizmów obiektowych w modelu relacyjnym:

- **Dziedziczenie i polimorfizm** – w obiektowym modelu mechanizm ten pozwala stosować obiekty o różnej liczbie atrybutów (obiekty generyczne). *Często aplikacja nie zna konkretnych wystąpień obiektów. W relacyjnych bazach danych wszystkie rekordy mają zawsze taką sama liczbę pól i klucze obce* zawsze odwołują się tylko do jednego rekordu z jednej tabeli
- **Złożone powiązania** – relacyjne bazy danych bardzo dobrze przedstawiają jednokierunkowe powiązania jeden do jeden (jedno- lub dwukierunkowe) lub wiele do jeden. Jednak powiązania wiele do wiele (jedno- lub dwukierunkowe) oraz jeden do wiele (jednokierunkowe) są trudniejsze do wyrażenia w modelu relacyjnym.

Easy to represent using relational databases:

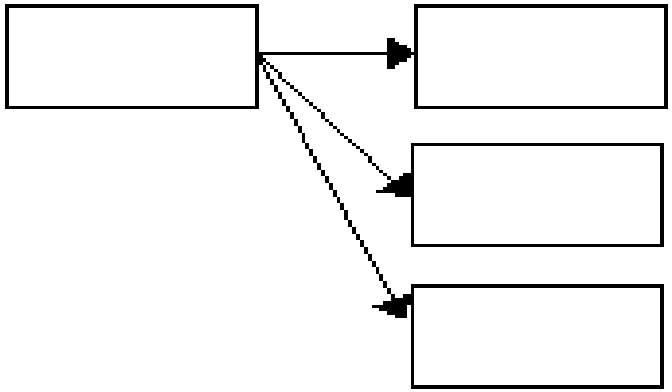


One-to-one
Uni- or bi-directional

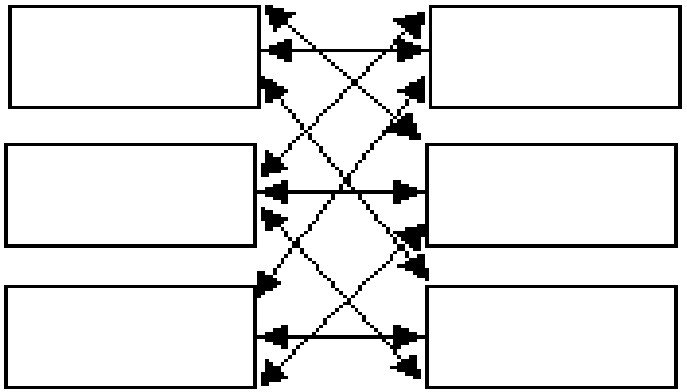


Many-to-one
Uni-directional

Difficult to represent using relational databases:

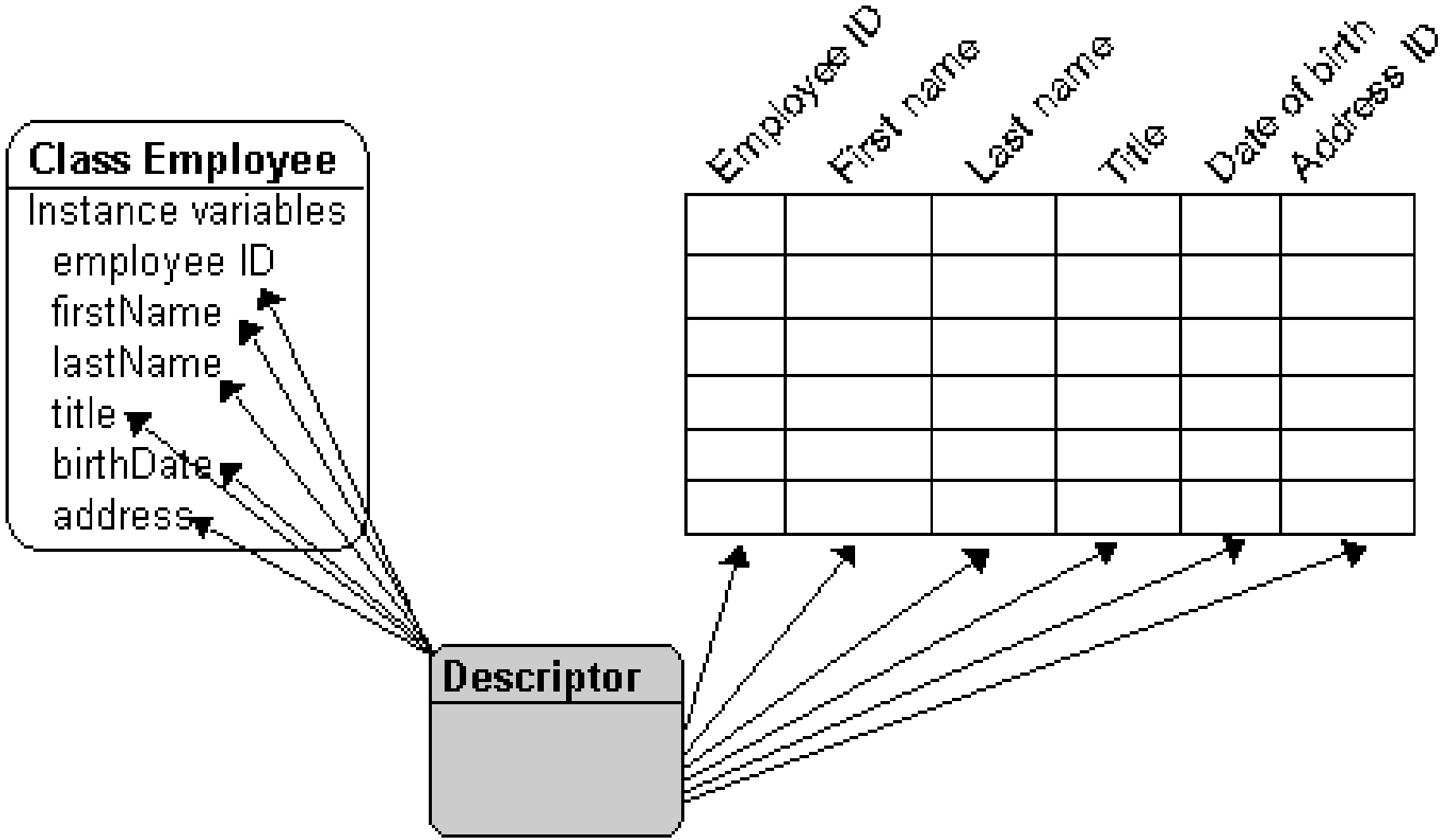


One-to-many
Uni-directional

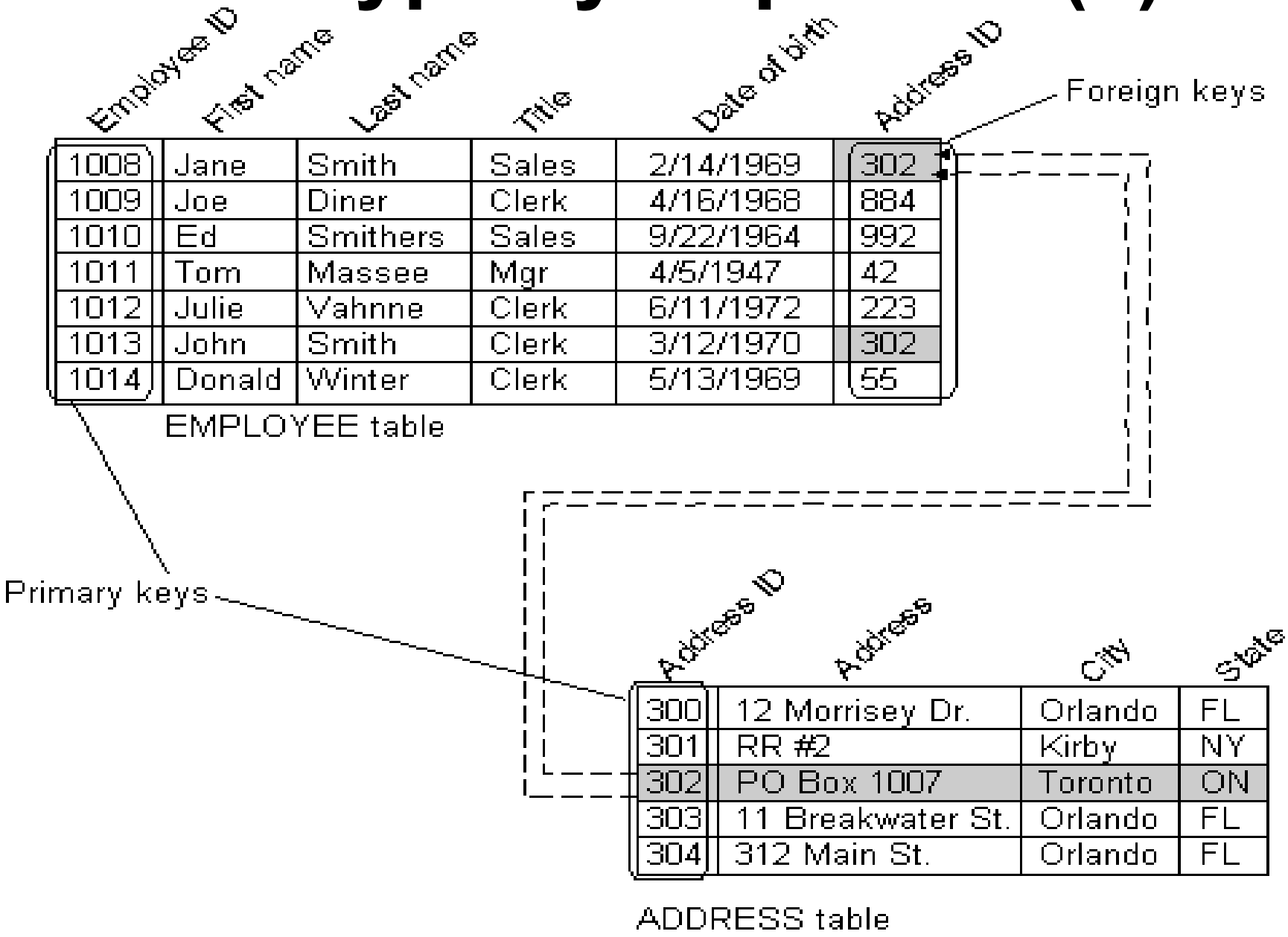


Many-to-many
Uni- or bi-directional

Deskryptory TopLink (1)



Deskryptory TopLink (2)



Deskryptory TopLink (3)

- Deskryptory klas Javy określające odpowiadające im **tabele** przechowujące instancje klas
- Deskryptory opisujące **klucze główne** tabeli
- Deskryptory określające listę kluczy w zapytaniach dla **nazw pól**
- Deskryptory opisujące **atrybuty obiektów i relacji** – ta informacja jest przechowywana w deskryptorach określanych jako „mapowania”
- Zbiór właściwości reprezentujących **zachowanie deskryptorów**

Deskryptory TopLink (4)

Mapowania

Mapowania przechowywane w deskryptorach definiują przechowywanie i odzyskiwanie atrybutów obiektów. TopLink używa dwa typy mapowań: **mapowania powiązań i mapowania bezpośrednie**

1) Mapowanie powiązań

Powiązania te mapują trzy typy powiązań: jeden-do-jeden, jeden-do-wiele, wiele-do-wiele. Jeden-do-jeden przechowują powiązanie między jedną klasą a drugą. Jeden-do-wiele i wiele-do-wiele mapują kolekcje referencji do innych klas.

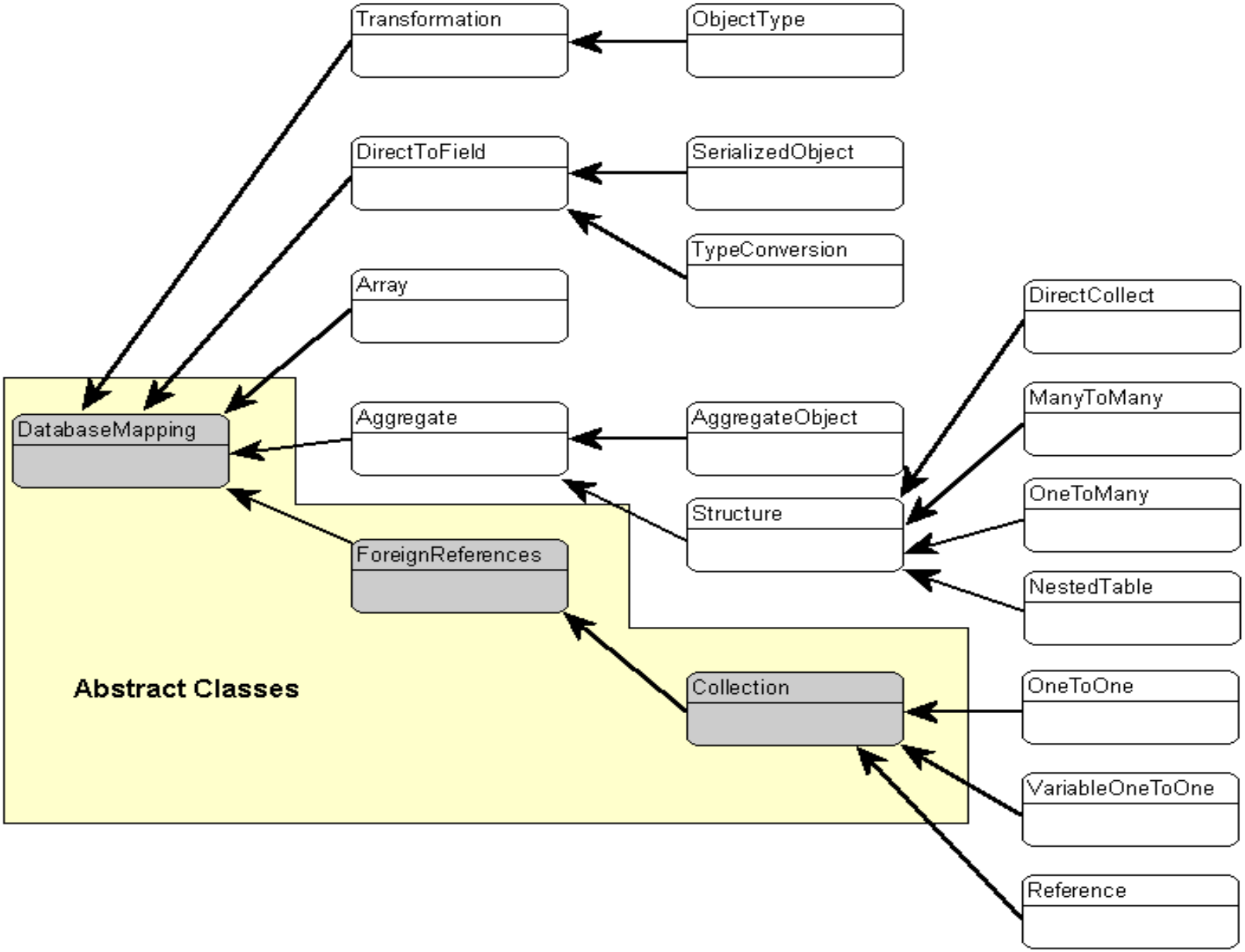
2) Bezpośrednie powiązania

Powiązania te mapują atrybuty. Istnieją dwa typy mapowań bezpośrednich: **bezpośrednie-do-pola** oraz **transformacje typu**.

Bezpośrednie-do-pola są najprostsze do użycia: przechowują atrybuty w bazie danych w polach o natywnym formacie. Mapowanie atrybutów, które nie są wspierane mapowaniem bezpośrednie-do-pola, muszą być wspierane mapowaniem transformacji typu.

Transformacje typu mapują transformacje danych z natywnego formatu Javy wspieranego przez relacyjną bazę danych. Istnieją cztery transformacje typów: mapowanie typu obiektów, mapowanie konwersji typu, mapowanie serializacji obiektów, mapowanie odwzorowań.

Schemat klas użytych do mapowania



Sesje bazodanowe

Klasa typu **DatabaseSession** reprezentuje dialog z relacyjną bazą danych i przechowuje następujące informacje:

- Wystąpienia projektów (informacje o konfiguracji) i logowań do bazy danych
- Instancja dostępu do bazy danych, która umożliwia połączenie typu JDBC i utrzymuje dostęp do bazy danych
- Deskryptory dla każdej aplikacji, która mapuje (utrwala) obiekty
- Identyczność mapowania, która utrzymuje identyczność obiektów i działa jako pamięć podręczna

Użycie TopLink w aplikacjach (1)

Po utworzeniu deskryptorów należy napisać kod Javy do rejestrowania deskryptorów sesji TopLink. Po zarejestrowaniu deskryptorów aplikacja jest gotowa do zapisu i czytania obiektów Javy do/z bazy danych:

- Do czytania obiektów z bazy danych używa się obiektów sesyjnych bazy danych
- Do zapisu obiektów do bazy danych używa się obiektów typu jednostka pracy

Użycie TopLink w aplikacjach(2)

Transakcje

- Są one zbiorem operacji bazodanowych, które muszą być zatwierdzane lub anulowane. Transakcje mogą być proste lub złożone.

Jednostka pracy

- Jest ona obiektem, który unifikuje transakcje bazodanowe związane ze zmianami obiektów Javy, definiując obiektowy poziom transakcji. „Jednostka pracy" poprawia wydajność transakcji określając jej zakres, który opowiada tej części obiektu, który uległ zmianie. „Jednostka pracy" jest preferowaną metodą zapisu w technologii TopLink.

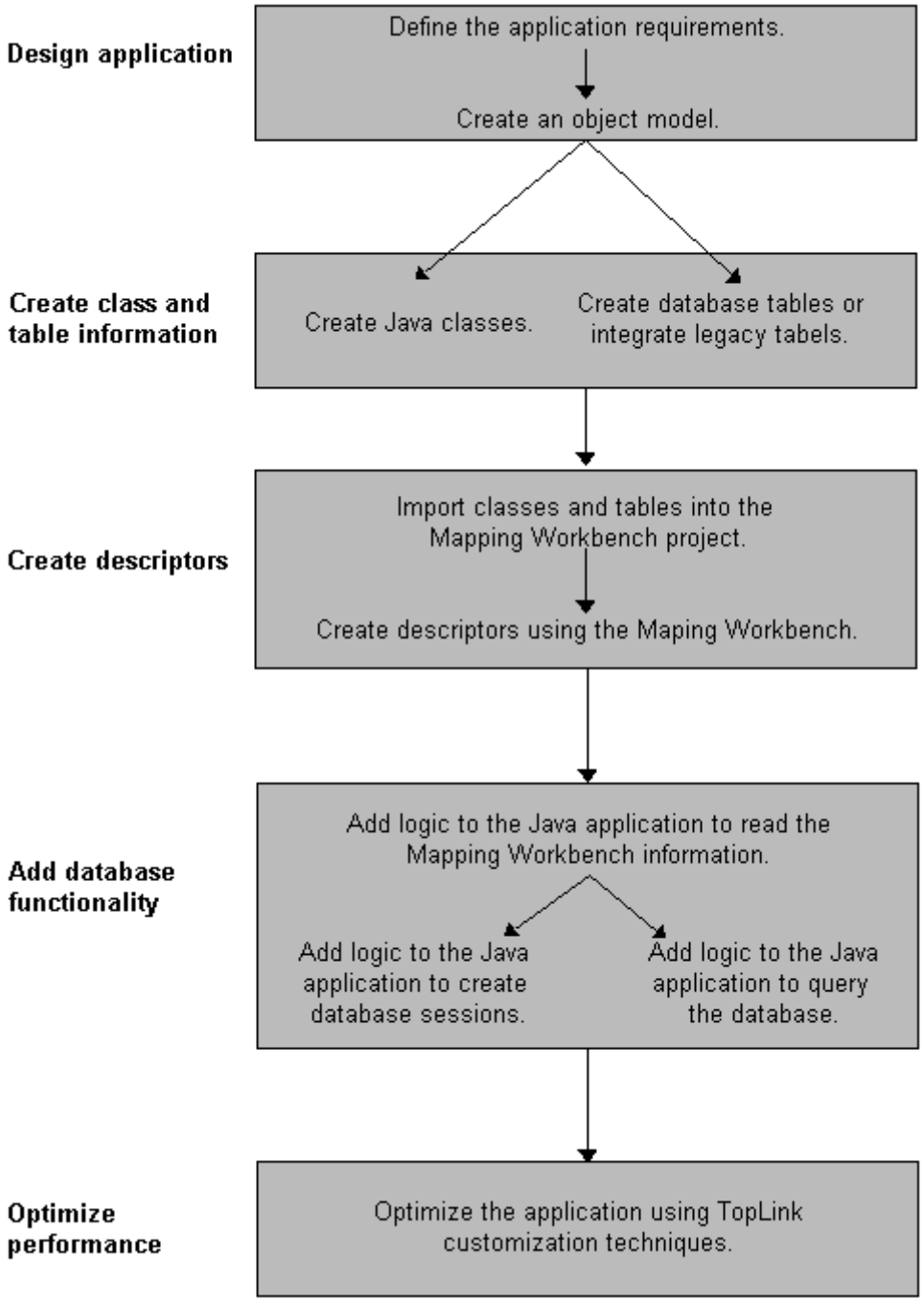
Odczytywanie i zapisywanie obiektów Java

- Podczas odczytywanie obiektów z bazy danych używa się metody sesji typu **readObject()** związaną z serializacją obiektów (**implements Serializable**). Do zapisu (transakcji) obiektów do bazy danych można używać metody sesji typu **writeObject(),** jednak jest ona alternatywą dla „jednostki pracy". Aplikacja, która używa „jednostek pracy" podczas zapisu czyta bazę danych i przez porównanie ze stanem obiektów przeznaczonych do zapisu określa te dane, które muszą być zmienione. Są one zarejestrowane w „jednostkach pracy" i jedynie dane zarejestrowane są zmieniane podczas fazy typu „commit".

- Taki model zapewnia znaczną poprawę wydajności operacji. Wydajność czytania jest optymalizowana za pomocą sesji, ponieważ jednostka pracy pamięta jedynie dostęp do tych obiektów, które ulegają zmianie. Wydajność zapisu jest optymalizowana za pomocą „jednostki pracy", ponieważ pamięta ona jedynie te dane, które uległy zmianie – mogą to być całe obiekty lub ich części.

Strategie i zalecenia projektowania

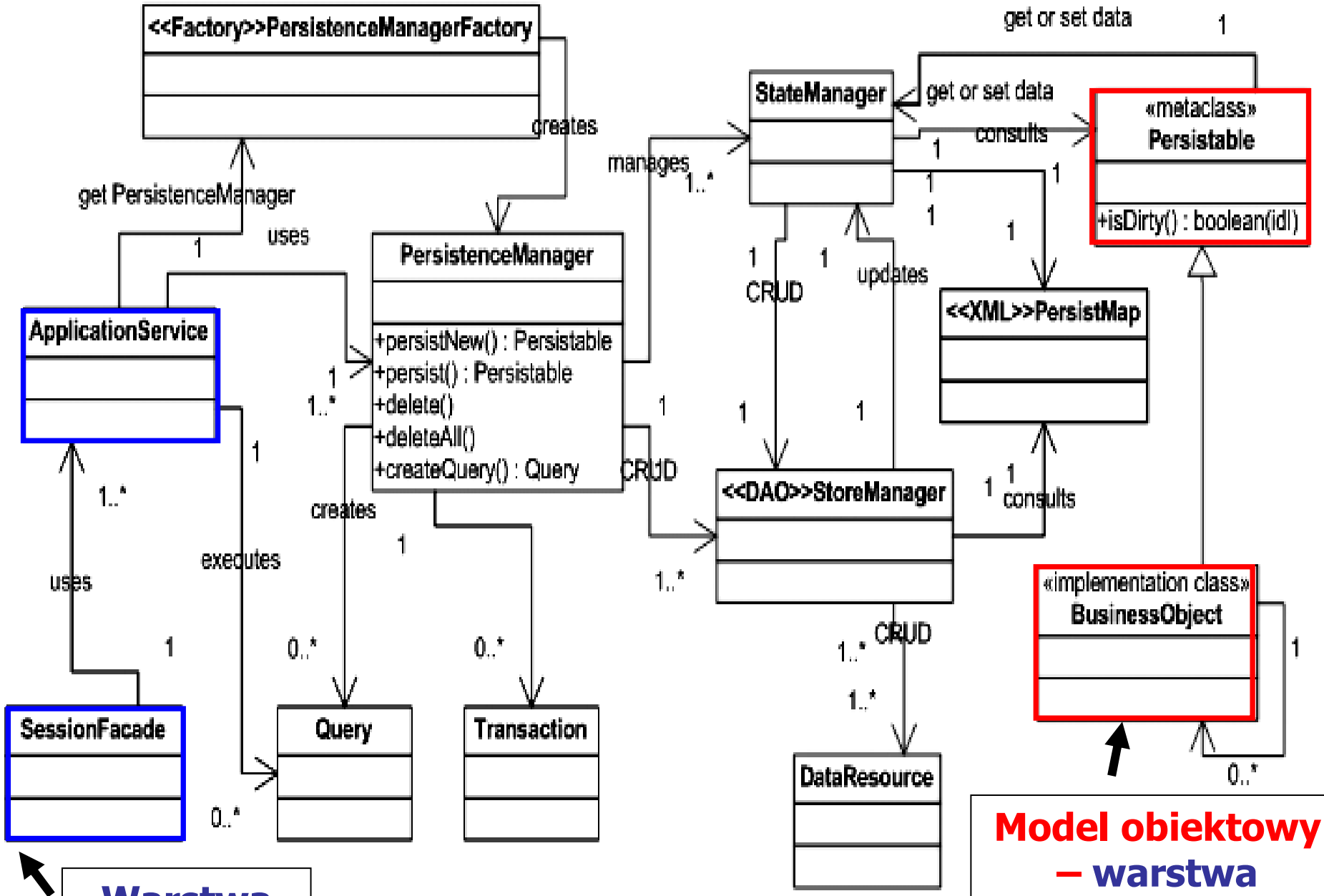
- Wykonaj model obiektowy aplikacji przed zastosowaniem mapowania. Zły model obiektowy powoduje utrudnienia w mapowaniu.
- Zadecyduj, które klasy powinny być przechowywane w bazie danych i ***zadecyduj o schemacie bazie danych***.
- Jeśli aplikacja musi być ***zintegrowana z istniejącą bazą danych***, należy zadecydować, która z tabel musi być mapowana do określonego typu obiektu.
- Napisz kod Javy wykorzystywany w stanie sesji bazy danych. Podczas sesji realizuje się zapytania oraz zapisy (również usuwania i zmiany) obiektów w bazie danych.
- Optymalizuj aplikację dobierając odpowiednie mechanizmy technologii TopLink.


Design application
Define the application requirements.
Create an object model.
Create class and table information
Create Java classes.
Create database tables or integrate legacy tabels.
Create descriptors
Import classes and tables into the Mapping Workbench project.
Create descriptors using the Maping Workbench.
Add database functionality
Add logic to the Java application to read the Mapping Workbench information.
Add logic to the Java application to create database sessions.
Add logic to the Java application to query the database.
Optimize performance
Optimize the application using TopLink customization techniques.

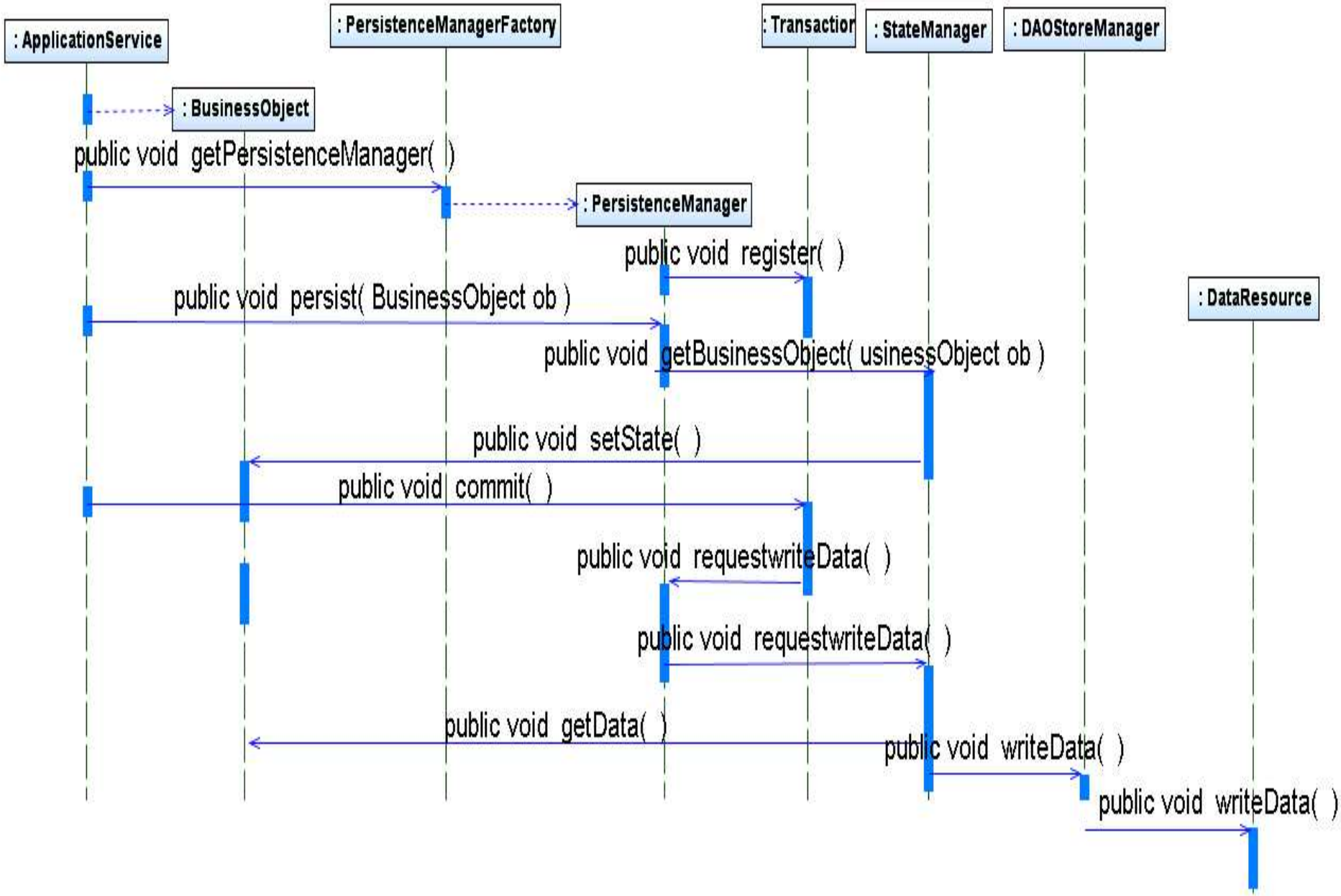
3. Wzorzec projektowy typu **Domain Store** (wzorzec J2EE) pozwalający na **oddzielenie mechanizmów trwałości od modelu obiektowego**

(materiał na podstawie wykładów o warstwie integracji z cyklu „Modelowanie i analiza systemów informatycznych”).

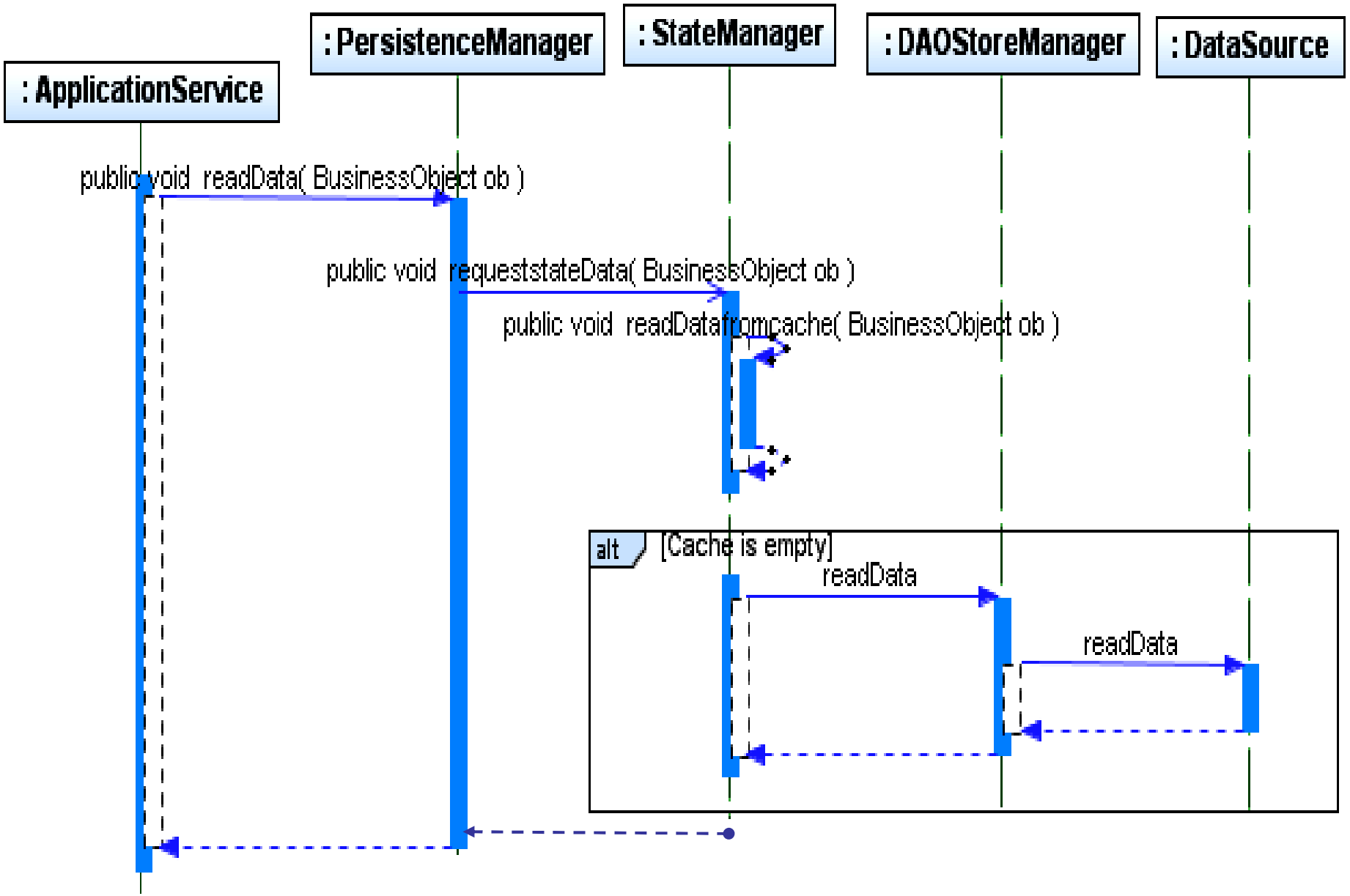
Domain Store (1) - Tworzenie i utrwalanie obiektu biznesowego



Domain Store (2) - Tworzenie i utrwalanie obiektu biznesowego



Domain Store (3) - Pobieranie danych



Domain Store (4) - Tworzenie i wykonanie zapytania