### Bazy Danych 2

### Laboratorium 5

Hibernate & JPA

Mateusz Łopaciński

### Kod po wprowadzeniu

#### **Zaimplementowane klasy**

#### **Klasa Main**

package com**.**matipl01**;**

**import** org**.**hibernate**.**HibernateException**;**

**import** org**.**hibernate**.**Session**;**

**import** org**.**hibernate**.**SessionFactory**;**

**import** org**.**hibernate**.**Transaction**;**

**import** org**.**hibernate**.**cfg**.**Configuration**;**

public class Main **{**

private static final SessionFactory ourSessionFactory**;**

static **{**

**try** **{**

Configuration configuration **=** **new** Configuration**();**

configuration**.**configure**();**

ourSessionFactory **=** configuration**.**buildSessionFactory**();**

**}** **catch** **(**Throwable ex**)** **{**

**throw** **new** ExceptionInInitializerError**(**ex**);**

**}**

**}**

public static Session getSession**()** **throws** HibernateException **{**

**return** ourSessionFactory**.**openSession**();**

**}**

public static void main**(**final String**[]** args**)** **throws** Exception **{**

final Session session **=** getSession**();**

Product product **=** **new** Product**(**"Krzesło"**,** 111**);**

**try** **{**

Transaction tx **=** session**.**beginTransaction**();**

session**.**save**(**product**);**

tx**.**commit**();**

**}** **finally** **{**

session**.**close**();**

**}**

**}**

**}**

#### **Klasa Product**

package com**.**matipl01**;**

**import** javax**.**persistence**.**Entity**;**

**import** javax**.**persistence**.**GeneratedValue**;**

**import** javax**.**persistence**.**GenerationType**;**

**import** javax**.**persistence**.**Id**;**

@Entity

public class Product **{**

@Id

@GeneratedValue**(**strategy **=** GenerationType**.**AUTO**)**

public int productID**;**

public String productName**;**

public int unitsInStock**;**

public Product**()** **{}**

public Product**(**String productName**,** int unitsInStock**)** **{**

**this.**productName **=** productName**;**

**this.**unitsInStock **=** unitsInStock**;**

**}**

**}**

#### **Plik konfiguracyjny hibernate.cfg.xml**

<?xml version=**"1.0"** encoding=**"UTF-8"**?>

<!DOCTYPE hibernate-configuration PUBLIC

"-//Hibernate/Hibernate Configuration DTD//EN"

"https://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<hibernate-configuration>

<session-factory>

<property name=**"connection.driver\_class"**>**org.apache.derby.jdbc.ClientDriver**</property>

<property name=**"connection.url"**>**jdbc:derby://127.0.0.1/LopacinskiMateuszJPA**</property>

<property name=**"dialect"**>**org.hibernate.dialect.DerbyTenSevenDialect**</property>

<property name=**"show\_sql"**>**true**</property>

<property name=**"format\_sql"**>**true**</property>

<property name=**"use\_sql\_comments"**>**true**</property>

<property name=**"hbm2ddl.auto"**>**create-drop**</property>

<mapping class=**"com.matipl01.Product"**/>

</session-factory>

</hibernate-configuration>

#### **Uzyskana tabela**

Obraz zawierający tekst

Opis wygenerowany automatycznie

### Wprowadzenie pojęcia Dostawcy

#### **Zaimplementowane klasy**

#### **Metoda main z klasy Main**

W implementacji klasy **Main** zmieniła się jedynie metoda public static void main, dlatego tylko kod tej metody umieściłem poniżej. Aby móc odczytać poprzednio dodany do bazy danych produkt, zmieniłem wartość właściwości **hbm2ddl.auto** w pliku konfiguracyjnym **hibernate.cfg.xml** na <property name=**"hbm2ddl.auto"**>**update**</property>.

public static void main**(**final String**[]** args**)** **{**

**try** **(**Session session **=** getSession**())** **{**

Transaction tx **=** session**.**beginTransaction**();**

// Create the new supplier

Supplier supplier **=** **new** Supplier**(**"Super dostawca"**,** "Malinowa"**,** "Poznań"**);**

// Get the previously added product

Product product **=** session**.**get**(**Product**.**class**,** 1**);**

product**.**setSupplier**(**supplier**);**

session**.**save**(**supplier**);**

tx**.**commit**();**

// Testowanie

Query query **=** session**.**createQuery**(**"from Product"**);**

query**.**getResultList**().**forEach**(**p **->** **{**

System**.**out**.**println**(**"Produkt '" **+** p **+** "' jest dostarczany przez '" **+** **((**Product**)** p**).**getSupplier**()** **+** "'"**);**

**});**

**}**

**}**

#### **Klasa Supplier**

Aby móc skorzystać z nowo dodanej klasy dostawcy, konieczne było jej dodanie do pliku **hibernate.cfg.xml**. Umieściłem więc w pliku konfiguracyjnym linijkę: <mapping class=**"com.matipl01.Supplier"**/>.

package com**.**matipl01**;**

**import** javax**.**persistence**.\*;**

@Entity

public class Supplier **{**

@Id

@GeneratedValue**(**strategy **=** GenerationType**.**AUTO**)**

public int supplierID**;**

private String companyName**;**

private String street**;**

private String city**;**

public Supplier**()** **{}**

public Supplier**(**String companyName**,** String street**,** String city**)** **{**

**this.**companyName **=** companyName**;**

**this.**street **=** street**;**

**this.**city **=** city**;**

**}**

@Override

public String toString**()** **{**

**return** companyName**;**

**}**

**}**

#### **Klasa Product**

package com**.**matipl01**;**

**import** javax**.**persistence**.\*;**

@Entity

public class Product **{**

@Id

@GeneratedValue**(**strategy **=** GenerationType**.**AUTO**)**

private int productID**;**

private String productName**;**

private int unitsInStock**;**

@ManyToOne

@JoinColumn**(**name **=** "supplierID"**)**

private Supplier supplier**;**

public Product**()** **{}**

public Product**(**String productName**,** int unitsInStock**)** **{**

**this.**productName **=** productName**;**

**this.**unitsInStock **=** unitsInStock**;**

**}**

@Override

public String toString**()** **{**

**return** productName **+** " (" **+** unitsInStock **+** " szt.)"**;**

**}**

public void setSupplier**(**Supplier supplier**)** **{**

**this.**supplier **=** supplier**;**

**}**

public Supplier getSupplier**()** **{**

**return** supplier**;**

**}**

**}**

#### **Logi SQL**

alter table Product

add column supplierID integer

create table Supplier (

supplierID integer not null,

city varchar(255),

companyName varchar(255),

street varchar(255),

primary key (supplierID)

)

alter table Product

add constraint FKj0x097f8xajoy9j9ryct9pf3o

foreign key (supplierID)

references Supplier

select

product0\_.productID as producti1\_0\_0\_,

product0\_.productName as productn2\_0\_0\_,

product0\_.supplierID as supplier4\_0\_0\_,

product0\_.unitsInStock as unitsins3\_0\_0\_,

supplier1\_.supplierID as supplier1\_1\_1\_,

supplier1\_.city as city2\_1\_1\_,

supplier1\_.companyName as companyn3\_1\_1\_,

supplier1\_.street as street4\_1\_1\_

from

Product product0\_

left outer join

Supplier supplier1\_

on product0\_.supplierID=supplier1\_.supplierID

where

product0\_.productID=?

values

next value for Supplier\_SEQ

/\* insert com.matipl01.Supplier

\*/ insert

into

Supplier

(city, companyName, street, supplierID)

values

(?, ?, ?, ?)

/\* update

com.matipl01.Product \*/ update

Product

set

productName=?,

supplierID=?,

unitsInStock=?

where

productID=?

/\*

from

Product \*/ select

product0\_.productID as producti1\_0\_,

product0\_.productName as productn2\_0\_,

product0\_.supplierID as supplier4\_0\_,

product0\_.unitsInStock as unitsins3\_0\_

from

Product product0\_

#### **Rezultat wykonania kodu**



#### **Diagram bazy danych**

Obraz zawierający tekst, tablica wyników, zrzut ekranu

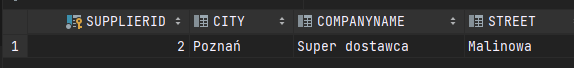
Opis wygenerowany automatycznie

#### **Utworzone tabele**

#### **Tabela Product**



#### **Tabela Supplier**



### Odwrócenie relacji

W pliku **hibernate.cfg.xml** zmieniłem z powrotem wartość **hbm2ddl.auto** na **create-drop**, ponieważ łatwiej jest mi tworzyć wszystkie encje od nowa, a następnie umieszczać je w bazie danych.

#### **3a. Z tabelą łącznikową**

#### **Zaimplementowane klasy**

#### **Metoda main z klasy Main**

W implementacji tej klasy zmieniła się jedynie metoda public static void main, dlatego ponownie załączam tylko tę metodę.

public static void main**(**final String**[]** args**)** **{**

**try** **(**Session session **=** getSession**())** **{**

Transaction tx **=** session**.**beginTransaction**();**

Product product1 **=** **new** Product**(**"Krzesło"**,** 111**);**

Product product2 **=** **new** Product**(**"Stół"**,** 23**);**

Product product3 **=** **new** Product**(**"Szafa"**,** 44**);**

Product product4 **=** **new** Product**(**"Komoda"**,** 53**);**

Supplier supplier1 **=** **new** Supplier**(**"Dostawca 1"**,** "Malinowa"**,** "Poznań"**);**

Supplier supplier2 **=** **new** Supplier**(**"Dostawca 2"**,** "Konwaliowa"**,** "Kraków"**);**

supplier1**.**addProducts**(**product1**,** product3**,** product4**);**

supplier2**.**addProducts**(**product2**);**

session**.**save**(**product1**);**

session**.**save**(**product2**);**

session**.**save**(**product3**);**

session**.**save**(**product4**);**

session**.**save**(**supplier1**);**

session**.**save**(**supplier2**);**

tx**.**commit**();**

// Testowanie

Query query **=** session**.**createQuery**(**"from Supplier"**);**

query**.**getResultList**().**forEach**(**s **->** **{**

**((**Supplier**)** s**).**getProducts**().**forEach**(**p **->** System**.**out**.**println**(**s **+** " dostarcza " **+** p**));**

**});**

**}**

**}**

#### **Klasa Supplier**

Tym razem sprzedawca zawiera zbiór produktów jako private final Collection**<**Product**>** products **=** **new** ArrayList**<>();**.

package com**.**matipl01**;**

**import** javax**.**persistence**.\*;**

**import** java**.**util**.**ArrayList**;**

**import** java**.**util**.**Arrays**;**

**import** java**.**util**.**Collection**;**

@Table**(**name **=** "Suppliers"**)**

@Entity

@SequenceGenerator**(**name **=** "Supplier\_SEQ"**)**

public class Supplier **{**

@Id

@GeneratedValue**(**strategy **=** GenerationType**.**SEQUENCE**,** generator **=** "Supplier\_SEQ"**)**

public int supplierID**;**

private String companyName**;**

private String street**;**

private String city**;**

@OneToMany

@JoinTable**(**

name **=** "SupplierProducts"**,**

joinColumns **=** @JoinColumn**(**name **=** "supplierID"**),**

inverseJoinColumns **=** @JoinColumn**(**name **=** "productID"**)**

**)**

private final Collection**<**Product**>** products **=** **new** ArrayList**<>();**

public Supplier**()** **{}**

public Supplier**(**String companyName**,** String street**,** String city**)** **{**

**this.**companyName **=** companyName**;**

**this.**street **=** street**;**

**this.**city **=** city**;**

**}**

@Override

public String toString**()** **{**

**return** companyName**;**

**}**

public Collection**<**Product**>** getProducts**()** **{**

**return** products**;**

**}**

public void addProducts**(**Product **...**products**)** **{**

**this.**products**.**addAll**(**Arrays**.**asList**(**products**));**

**}**

**}**

#### **Klasa Product**

package com**.**matipl01**;**

**import** javax**.**persistence**.\*;**

@Table**(**name **=** "Products"**)**

@Entity

@SequenceGenerator**(**name **=** "Product\_SEQ"**)**

public class Product **{**

@Id

@GeneratedValue**(**strategy**=**GenerationType**.**SEQUENCE**,** generator **=** "Product\_SEQ"**)**

private int productID**;**

private String productName**;**

private int unitsInStock**;**

public Product**()** **{}**

public Product**(**String productName**,** int unitsInStock**)** **{**

**this.**productName **=** productName**;**

**this.**unitsInStock **=** unitsInStock**;**

**}**

@Override

public String toString**()** **{**

**return** productName **+** " (" **+** unitsInStock **+** " szt.)"**;**

**}**

**}**

#### **Logi SQL**

Pomijam **DROP TABLE** oraz kod odpowiedzialny za tworzenie i korzystanie z sekwencji (używam ich po to, aby dla każdej tabeli id było generowane niezależnie, startując od 1).

create table Products (

productID integer not null,

productName varchar(255),

unitsInStock integer not null,

primary key (productID)

)

create table SupplierProducts (

supplierID integer not null,

productID integer not null

)

create table Suppliers (

supplierID integer not null,

city varchar(255),

companyName varchar(255),

street varchar(255),

primary key (supplierID)

)

alter table SupplierProducts

add constraint UK\_9nc9hk63pkcj735l1lw563bh5 unique (productID)

alter table SupplierProducts

add constraint FKnvospn0k2a1ldi72ui92wv0wg

foreign key (productID)

references Products

alter table SupplierProducts

add constraint FKrglebkocbp0c6faljji6kkind

foreign key (supplierID)

references Suppliers

/\* insert com.matipl01.Product

\*/ insert

into

Products

(productName, unitsInStock, productID)

values

(?, ?, ?)

/\* insert com.matipl01.Product

\*/ insert

into

Products

(productName, unitsInStock, productID)

values

(?, ?, ?)

/\* insert com.matipl01.Product

\*/ insert

into

Products

(productName, unitsInStock, productID)

values

(?, ?, ?)

/\* insert com.matipl01.Product

\*/ insert

into

Products

(productName, unitsInStock, productID)

values

(?, ?, ?)

/\* insert com.matipl01.Supplier

\*/ insert

into

Suppliers

(city, companyName, street, supplierID)

values

(?, ?, ?, ?)

/\* insert com.matipl01.Supplier

\*/ insert

into

Suppliers

(city, companyName, street, supplierID)

values

(?, ?, ?, ?)

/\* insert collection

row com.matipl01.Supplier.products \*/ insert

into

SupplierProducts

(supplierID, productID)

values

(?, ?)

/\* insert collection

row com.matipl01.Supplier.products \*/ insert

into

SupplierProducts

(supplierID, productID)

values

(?, ?)

/\* insert collection

row com.matipl01.Supplier.products \*/ insert

into

SupplierProducts

(supplierID, productID)

values

(?, ?)

/\* insert collection

row com.matipl01.Supplier.products \*/ insert

into

SupplierProducts

(supplierID, productID)

values

(?, ?)

/\*

from

Supplier \*/ select

supplier0\_.supplierID as supplier1\_2\_,

supplier0\_.city as city2\_2\_,

supplier0\_.companyName as companyn3\_2\_,

supplier0\_.street as street4\_2\_

from

Suppliers supplier0\_

#### **Rezultat wykonania kodu**

#### **Obraz zawierający tekst Opis wygenerowany automatycznie**

#### **Diagram bazy danych**

Obraz zawierający tekst, tablica wyników

Opis wygenerowany automatycznie

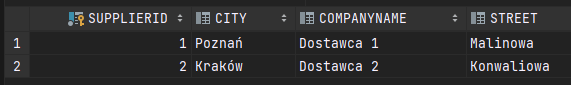
#### **Utworzone tabele**

#### **Tabela Products**

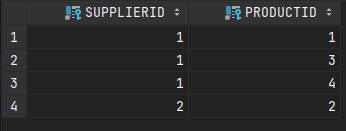
Obraz zawierający tekst, sprzęt elektroniczny, tablica wyników, wyświetlanie

Opis wygenerowany automatycznie

#### **Tabela Suppliers**



#### **Tabela SupplierProducts**



#### **3b. Bez tabeli łącznikowej**

#### **Zaimplementowane klasy**

#### **Metoda main z klasy Main**

Bez zmian

#### **Klasa Supplier**

Zmiana dotyczy jedynie dekoratorów dekorujących kolekcję private final Collection**<**Product**>** products **=** **new** ArrayList**<>();**. Zmodyfikowany został jedynie dekorator @JoinTable. Otrzymujemy więc finalnie:

@OneToMany

@JoinColumn**(**name **=** "supplierID"**)**

private final Collection**<**Product**>** products **=** **new** ArrayList**<>();**

#### **Klasa Product**

Bez zmian

#### **Logi SQL**

Pomijam **DROP TABLE** (usuwanie tabel z poprzedniego zadania) oraz kod odpowiedzialny za tworzenie i korzystanie z sekwencji (używam ich po to, aby dla każdej tabeli id było generowane niezależnie, startując od 1).

create table Products (

productID integer not null,

productName varchar(255),

unitsInStock integer not null,

supplierID integer,

primary key (productID)

)

create table Suppliers (

supplierID integer not null,

city varchar(255),

companyName varchar(255),

street varchar(255),

primary key (supplierID)

)

alter table Products

add constraint FKbjx75exi25f1c48i92gu8rvlx

foreign key (supplierID)

references Suppliers

/\* insert com.matipl01.Product

\*/ insert

into

Products

(productName, unitsInStock, productID)

values

(?, ?, ?)

/\* insert com.matipl01.Product

\*/ insert

into

Products

(productName, unitsInStock, productID)

values

(?, ?, ?)

/\* insert com.matipl01.Product

\*/ insert

into

Products

(productName, unitsInStock, productID)

values

(?, ?, ?)

/\* insert com.matipl01.Product

\*/ insert

into

Products

(productName, unitsInStock, productID)

values

(?, ?, ?)

/\* insert com.matipl01.Supplier

\*/ insert

into

Suppliers

(city, companyName, street, supplierID)

values

(?, ?, ?, ?)

/\* insert com.matipl01.Supplier

\*/ insert

into

Suppliers

(city, companyName, street, supplierID)

values

(?, ?, ?, ?)

/\* create one-to-many row com.matipl01.Supplier.products \*/ update

Products

set

supplierID=?

where

productID=?

/\* create one-to-many row com.matipl01.Supplier.products \*/ update

Products

set

supplierID=?

where

productID=?

/\* create one-to-many row com.matipl01.Supplier.products \*/ update

Products

set

supplierID=?

where

productID=?

/\* create one-to-many row com.matipl01.Supplier.products \*/ update

Products

set

supplierID=?

where

productID=?

/\*

from

Supplier \*/ select

supplier0\_.supplierID as supplier1\_1\_,

supplier0\_.city as city2\_1\_,

supplier0\_.companyName as companyn3\_1\_,

supplier0\_.street as street4\_1\_

from

Suppliers supplier0\_

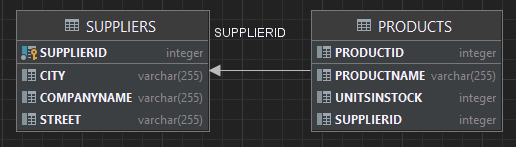
#### **Rezultat wykonania kodu**

Taki sam jak poprzednio.

Obraz zawierający tekst

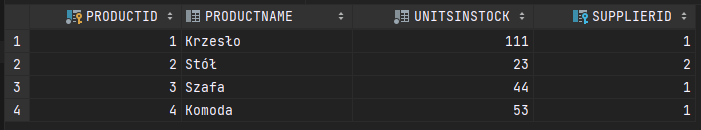
Opis wygenerowany automatycznie

#### **Diagram bazy danych**

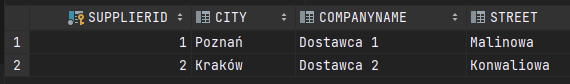


#### **Utworzone tabele**

#### **Tabela Products**



#### **Tabela Suppliers**



### Relacja dwustronna

Łączymy poprzednie rozwiązania

#### **Zaimplementowane klasy**

#### **Matoda main z klasy Main**

public static void main**(**final String**[]** args**)** **{**

**try** **(**Session session **=** getSession**())** **{**

Transaction tx **=** session**.**beginTransaction**();**

Product product1 **=** **new** Product**(**"Krzesło"**,** 111**);**

Product product2 **=** **new** Product**(**"Stół"**,** 23**);**

Product product3 **=** **new** Product**(**"Szafa"**,** 44**);**

Product product4 **=** **new** Product**(**"Komoda"**,** 53**);**

Supplier supplier1 **=** **new** Supplier**(**"Dostawca 1"**,** "Malinowa"**,** "Poznań"**);**

Supplier supplier2 **=** **new** Supplier**(**"Dostawca 2"**,** "Konwaliowa"**,** "Kraków"**);**

supplier1**.**addProducts**(**product1**,** product3**,** product4**);**

product1**.**setSupplier**(**supplier1**);**

product3**.**setSupplier**(**supplier1**);**

product4**.**setSupplier**(**supplier1**);**

supplier2**.**addProducts**(**product2**);**

product2**.**setSupplier**(**supplier2**);**

session**.**save**(**product1**);**

session**.**save**(**product2**);**

session**.**save**(**product3**);**

session**.**save**(**product4**);**

session**.**save**(**supplier1**);**

session**.**save**(**supplier2**);**

tx**.**commit**();**

// Testowanie

Query query **=** session**.**createQuery**(**"from Supplier"**);**

query**.**getResultList**().**forEach**(**s **->** **{**

**((**Supplier**)** s**).**getProducts**().**forEach**(**p **->** System**.**out**.**println**(**s **+** " dostarcza " **+** p**));**

**});**

query **=** session**.**createQuery**(**"from Product"**);**

query**.**getResultList**().**forEach**(**p **->** **{**

System**.**out**.**println**(**p **+** " jest dostarczany/e/a przez " **+** **((**Product**)** p**).**getSupplier**());**

**});**

**}**

**}**

#### **Klasa Supplier**

package com**.**matipl01**;**

**import** javax**.**persistence**.\*;**

**import** java**.**util**.**ArrayList**;**

**import** java**.**util**.**Arrays**;**

**import** java**.**util**.**Collection**;**

@Table**(**name **=** "Suppliers"**)**

@Entity

@SequenceGenerator**(**name **=** "Supplier\_SEQ"**)**

public class Supplier **{**

@Id

@GeneratedValue**(**strategy **=** GenerationType**.**SEQUENCE**,** generator **=** "Supplier\_SEQ"**)**

public int supplierID**;**

private String companyName**;**

private String street**;**

private String city**;**

@OneToMany

@JoinColumn**(**name**=**"supplierID"**)**

private final Collection**<**Product**>** products **=** **new** ArrayList**<>();**

public Supplier**()** **{}**

public Supplier**(**String companyName**,** String street**,** String city**)** **{**

**this.**companyName **=** companyName**;**

**this.**street **=** street**;**

**this.**city **=** city**;**

**}**

@Override

public String toString**()** **{**

**return** companyName**;**

**}**

public Collection**<**Product**>** getProducts**()** **{**

**return** products**;**

**}**

public void addProducts**(**Product **...**products**)** **{**

**this.**products**.**addAll**(**Arrays**.**asList**(**products**));**

**}**

**}**

#### **Klasa Product**

package com**.**matipl01**;**

**import** javax**.**persistence**.\*;**

@Table**(**name **=** "Products"**)**

@Entity

@SequenceGenerator**(**name **=** "Product\_SEQ"**)**

public class Product **{**

@Id

@GeneratedValue**(**strategy**=**GenerationType**.**SEQUENCE**,** generator **=** "Product\_SEQ"**)**

private int productID**;**

private String productName**;**

private int unitsInStock**;**

@ManyToOne

@JoinColumn**(**name **=** "supplierID"**)**

Supplier supplier**;**

public Product**()** **{}**

public Product**(**String productName**,** int unitsInStock**)** **{**

**this.**productName **=** productName**;**

**this.**unitsInStock **=** unitsInStock**;**

**}**

@Override

public String toString**()** **{**

**return** productName **+** " (" **+** unitsInStock **+** " szt.)"**;**

**}**

public void setSupplier**(**Supplier supplier**)** **{**

**this.**supplier **=** supplier**;**

**}**

public Supplier getSupplier**()** **{**

**return** supplier**;**

**}**

**}**

#### **Logi SQL**

Pomijam **DROP TABLE** oraz kod odpowiedzialny za tworzenie i korzystanie z sekwencji (używam ich po to, aby dla każdej tabeli id było generowane niezależnie, startując od 1).

create table Products (

productID integer not null,

productName varchar(255),

unitsInStock integer not null,

supplierID integer,

primary key (productID)

)

create table Suppliers (

supplierID integer not null,

city varchar(255),

companyName varchar(255),

street varchar(255),

primary key (supplierID)

)

alter table Products

add constraint FKbjx75exi25f1c48i92gu8rvlx

foreign key (supplierID)

references Suppliers

/\* insert com.matipl01.Product

\*/ insert

into

Products

(productName, supplierID, unitsInStock, productID)

values

(?, ?, ?, ?)

/\* insert com.matipl01.Product

\*/ insert

into

Products

(productName, supplierID, unitsInStock, productID)

values

(?, ?, ?, ?)

/\* insert com.matipl01.Product

\*/ insert

into

Products

(productName, supplierID, unitsInStock, productID)

values

(?, ?, ?, ?)

/\* insert com.matipl01.Product

\*/ insert

into

Products

(productName, supplierID, unitsInStock, productID)

values

(?, ?, ?, ?)

/\* insert com.matipl01.Supplier

\*/ insert

into

Suppliers

(city, companyName, street, supplierID)

values

(?, ?, ?, ?)

/\* insert com.matipl01.Supplier

\*/ insert

into

Suppliers

(city, companyName, street, supplierID)

values

(?, ?, ?, ?)

/\* update

com.matipl01.Product \*/ update

Products

set

productName=?,

supplierID=?,

unitsInStock=?

where

productID=?

/\* update

com.matipl01.Product \*/ update

Products

set

productName=?,

supplierID=?,

unitsInStock=?

where

productID=?

/\* update

com.matipl01.Product \*/ update

Products

set

productName=?,

supplierID=?,

unitsInStock=?

where

productID=?

/\* update

com.matipl01.Product \*/ update

Products

set

productName=?,

supplierID=?,

unitsInStock=?

where

productID=?

/\* create one-to-many row com.matipl01.Supplier.products \*/ update

Products

set

supplierID=?

where

productID=?

/\* create one-to-many row com.matipl01.Supplier.products \*/ update

Products

set

supplierID=?

where

productID=?

/\* create one-to-many row com.matipl01.Supplier.products \*/ update

Products

set

supplierID=?

where

productID=?

/\* create one-to-many row com.matipl01.Supplier.products \*/ update

Products

set

supplierID=?

where

productID=?

/\*

from

Supplier \*/ select

supplier0\_.supplierID as supplier1\_1\_,

supplier0\_.city as city2\_1\_,

supplier0\_.companyName as companyn3\_1\_,

supplier0\_.street as street4\_1\_

from

Suppliers supplier0\_

/\*

from

Product \*/ select

product0\_.productID as producti1\_0\_,

product0\_.productName as productn2\_0\_,

product0\_.supplierID as supplier4\_0\_,

product0\_.unitsInStock as unitsins3\_0\_

from

Products product0\_

#### **Rezultat wykonania kodu**

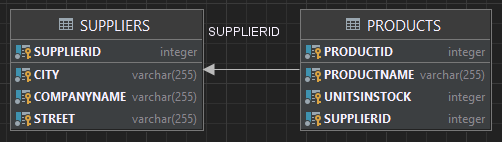
Obraz zawierający tekst

Opis wygenerowany automatycznie

Obraz zawierający tekst

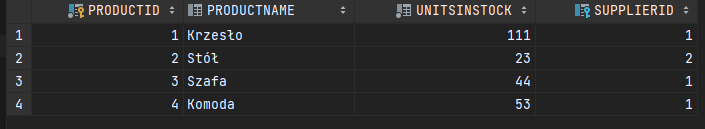
Opis wygenerowany automatycznie

#### **Diagram bazy danych**

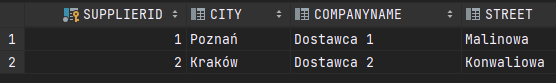


#### **Utworzone tabele**

#### **Tabela Products**



#### **Tabela Suppliers**



### Dodanie klasy Category

Ponieważ znów konieczne jest zmodyfikowanie istniejących już produktów, ponownie zmieniłem wartość **hbm2ddl.auto** w pliku **hibarnate.cfg.xml** na **update**.

#### **Zaimplementowane klasy**

Poniżej umieściłem jedynie kod klas, które używałem podczas realizacji tego podpunktu.

#### **Matoda main z klasy Main**

public static void main**(**final String**[]** args**)** **{**

Query query**;**

**try** **(**Session session **=** getSession**())** **{**

Transaction tx **=** session**.**beginTransaction**();**

Category furniture **=** **new** Category**(**"meble"**);**

Category food **=** **new** Category**(**"food"**);**

// Assign category to the existing products

query **=** session**.**createQuery**(**"from Product"**);**

List**<**Product**>** products **=** query**.**getResultList**();**

products**.**forEach**(**furniture**::**addProduct**);**

products**.**forEach**(**p **->** p**.**setCategory**(**furniture**));**

// Create new products and assign a category to them

Product apple **=** **new** Product**(**"Jabłko"**,** food**,** 59**);**

Product bread **=** **new** Product**(**"Chleb"**,** food**,** 232**);**

food**.**addProduct**(**apple**);**

food**.**addProduct**(**bread**);**

session**.**save**(**furniture**);**

session**.**save**(**food**);**

session**.**save**(**apple**);**

session**.**save**(**bread**);**

// Tests

// Produkty należące do kategorii

query **=** session**.**createQuery**(**"from Category"**);**

query**.**getResultList**().**forEach**(**c **->** **{**

System**.**out**.**println**(**"Kategoria: " **+** c **+** ": "**);**

**((**Category**)** c**).**getProducts**().**forEach**(**p **->** System**.**out**.**println**(**"\t- " **+** p **+** ","**));**

**});**

tx**.**commit**();**

// Kategoria, do której należy produkt

System**.**out**.**println**(**apple **+** " należy do kategorii: " **+** apple**.**getCategory**());**

**if** **(**products**.**size**()** **>=** 2**)** **{**

Product product **=** products**.**get**(**1**);**

System**.**out**.**println**(**product **+** " należy do kategorii: " **+** product**.**getCategory**());**

**}**

**}**

#### **Klasa Product**

package com**.**matipl01**;**

**import** javax**.**persistence**.\*;**

@Entity

@Table**(**name **=** "Products"**)**

public class Product **{**

@Id

@GeneratedValue**(**strategy **=** GenerationType**.**AUTO**)**

private int productID**;**

private String productName**;**

private int unitsInStock**;**

@ManyToOne

@JoinColumn**(**name **=** "supplierID"**)**

private Supplier supplier**;**

@ManyToOne

@JoinColumn**(**name **=** "categoryID"**)**

private Category category**;**

public Product**()** **{}**

public Product**(**String productName**,** Category category**,** int unitsInStock**)** **{**

**this.**productName **=** productName**;**

**this.**category **=** category**;**

**this.**unitsInStock **=** unitsInStock**;**

**}**

@Override

public String toString**()** **{**

**return** productName **+** " (" **+** unitsInStock **+** " szt.)"**;**

**}**

public void setSupplier**(**Supplier supplier**)** **{**

**this.**supplier **=** supplier**;**

**}**

public Supplier getSupplier**()** **{**

**return** supplier**;**

**}**

public void setCategory**(**Category category**)** **{**

**this.**category **=** category**;**

**}**

public Category getCategory**()** **{**

**return** category**;**

**}**

**}**

#### **Klasa Category**

package com**.**matipl01**;**

**import** javax**.**persistence**.\*;**

**import** java**.**util**.**ArrayList**;**

**import** java**.**util**.**Collection**;**

@Table**(**name **=** "Categories"**)**

@Entity

public class Category **{**

@Id

@GeneratedValue**(**strategy **=** GenerationType**.**AUTO**)**

private int categoryID**;**

private String name**;**

@OneToMany

private final Collection**<**Product**>** products **=** **new** ArrayList**<>();**

public Category**()** **{}**

public Category**(**String name**)** **{**

**this.**name **=** name**;**

**}**

@Override

public String toString**()** **{**

**return** name**;**

**}**

public Collection**<**Product**>** getProducts**()** **{**

**return** products**;**

**}**

public void addProduct**(**Product product**)** **{**

products**.**add**(**product**);**

**}**

**}**

#### **Logi SQL**

Pomijam **DROP TABLE** oraz kod odpowiedzialny za tworzenie i korzystanie z sekwencji (używam ich po to, aby dla każdej tabeli id było generowane niezależnie, startując od 1).

create table Categories (

categoryID integer not null,

name varchar(255),

primary key (categoryID)

)

create table CategoryProducts (

categoryID integer not null,

productID integer not null

)

alter table Products

add column categoryID integer

alter table CategoryProducts

drop constraint UK\_mow7rlswv9jtxx0tcbp28l6wo

alter table CategoryProducts

add constraint UK\_mow7rlswv9jtxx0tcbp28l6wo unique (productID)

alter table CategoryProducts

add constraint FKgiwpkxte1nl9np07vexa39fmj

foreign key (productID)

references Products

alter table CategoryProducts

add constraint FK35n26gmakywq04wr4tswhk2hm

foreign key (categoryID)

references Categories

alter table Products

add constraint FKn4dvny5ajgqgw20l5nb7imd5t

foreign key (categoryID)

references Categories

alter table Products

add constraint FKd0xbw89h0araa5nub4llrij88

foreign key (productID)

references Suppliers

/\*

from

Product \*/ select

product0\_.productID as producti1\_2\_,

product0\_.categoryID as category4\_2\_,

product0\_.productName as productn2\_2\_,

product0\_.supplierID as supplier5\_2\_,

product0\_.unitsInStock as unitsins3\_2\_

from

Products product0\_

select

supplier0\_.supplierID as supplier1\_3\_0\_,

supplier0\_.city as city2\_3\_0\_,

supplier0\_.companyName as companyn3\_3\_0\_,

supplier0\_.street as street4\_3\_0\_

from

Suppliers supplier0\_

where

supplier0\_.supplierID=?

select

supplier0\_.supplierID as supplier1\_3\_0\_,

supplier0\_.city as city2\_3\_0\_,

supplier0\_.companyName as companyn3\_3\_0\_,

supplier0\_.street as street4\_3\_0\_

from

Suppliers supplier0\_

where

supplier0\_.supplierID=?

/\* insert com.matipl01.Category

\*/ insert

into

Categories

(name, categoryID)

values

(?, ?)

/\* insert com.matipl01.Category

\*/ insert

into

Categories

(name, categoryID)

values

(?, ?)

/\* insert com.matipl01.Product

\*/ insert

into

Products

(categoryID, productName, supplierID, unitsInStock, productID)

values

(?, ?, ?, ?, ?)

/\* insert com.matipl01.Product

\*/ insert

into

Products

(categoryID, productName, supplierID, unitsInStock, productID)

values

(?, ?, ?, ?, ?)

/\* update

com.matipl01.Product \*/ update

Products

set

categoryID=?,

productName=?,

supplierID=?,

unitsInStock=?

where

productID=?

/\* update

com.matipl01.Product \*/ update

Products

set

categoryID=?,

productName=?,

supplierID=?,

unitsInStock=?

where

productID=?

/\* update

com.matipl01.Product \*/ update

Products

set

categoryID=?,

productName=?,

supplierID=?,

unitsInStock=?

where

productID=?

/\* update

com.matipl01.Product \*/ update

Products

set

categoryID=?,

productName=?,

supplierID=?,

unitsInStock=?

where

productID=?

/\* insert collection

row com.matipl01.Category.products \*/ insert

into

CategoryProducts

(categoryID, productID)

values

(?, ?)

/\* insert collection

row com.matipl01.Category.products \*/ insert

into

CategoryProducts

(categoryID, productID)

values

(?, ?)

/\* insert collection

row com.matipl01.Category.products \*/ insert

into

CategoryProducts

(categoryID, productID)

values

(?, ?)

/\* insert collection

row com.matipl01.Category.products \*/ insert

into

CategoryProducts

(categoryID, productID)

values

(?, ?)

/\* insert collection

row com.matipl01.Category.products \*/ insert

into

CategoryProducts

(categoryID, productID)

values

(?, ?)

/\* insert collection

row com.matipl01.Category.products \*/ insert

into

CategoryProducts

(categoryID, productID)

values

(?, ?)

/\*

from

Category \*/ select

category0\_.categoryID as category1\_0\_,

category0\_.name as name2\_0\_

from

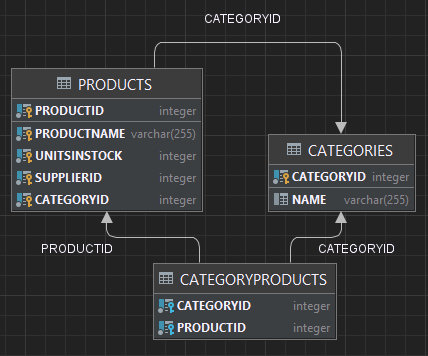
Categories category0\_

#### **Rezultat wykonania kodu**

Obraz zawierający tekst

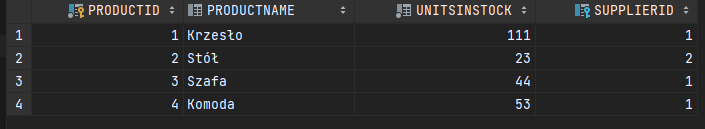
Opis wygenerowany automatycznie

#### **Diagram bazy danych (tylko tabele z tego podpunktu)**



#### **Utworzone tabele**

#### **Tabela Products**



#### **Tabela Suppliers**