### 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

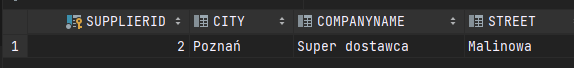
Opis wygenerowany automatycznie

#### **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

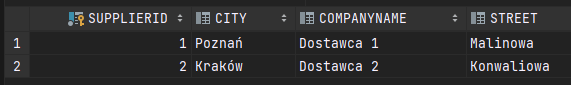
#### **Utworzone tabele**

#### **Tabela Products**

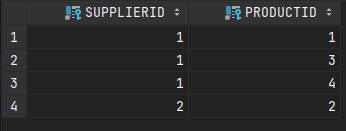
Obraz zawierający tekst, sprzęt elektroniczny, tablica wyników, wyświetlanie

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\_

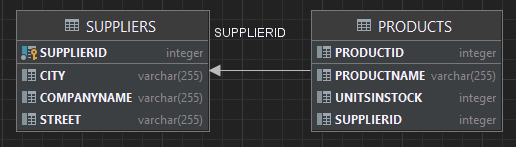
#### **Rezultat wykonania kodu**

Taki sam jak poprzednio.

Obraz zawierający tekst

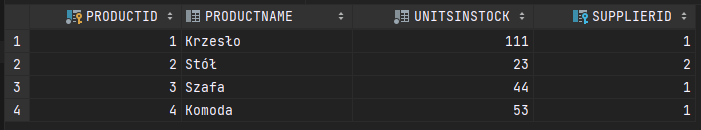
Opis wygenerowany automatycznie

#### **Diagram bazy danych**

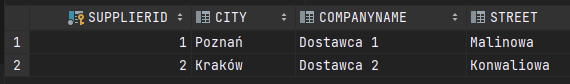


#### **Utworzone tabele**

#### **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**;**

@Entity

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

public class Supplier **{**

@Id

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

@SequenceGenerator**(**name **=** "Supplier\_GEN"**,** sequenceName **=** "Supplier\_SEQ"**)**

public int supplierID**;**

private String companyName**;**

private String street**;**

private String city**;**

@OneToMany**(**mappedBy **=** "supplier"**)**

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**.\*;**

@Entity

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

public class Product **{**

@Id

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

@SequenceGenerator**(**name **=** "Product\_GEN"**,** sequenceName **=** "Product\_SEQ"**)**

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**

Pomijam **DROP TABLE** oraz kod odpowiedzialny za tworzenie i korzystanie z sekwencji.

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**

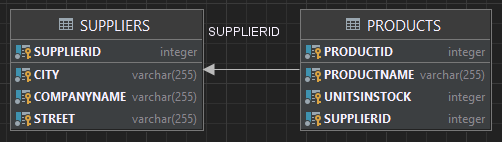
Obraz zawierający tekst

Opis wygenerowany automatycznie

Obraz zawierający tekst

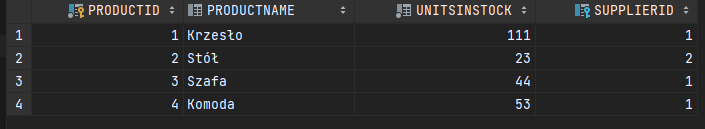
Opis wygenerowany automatycznie

#### **Diagram bazy danych**

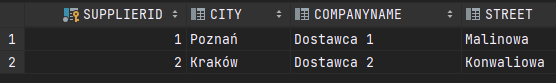


#### **Utworzone tabele**

#### **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"**); 🡨 wiem, że powinno być po polsku**

// 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**.**SEQUENCE**,** generator **=** "Product\_GEN"**)**

@SequenceGenerator**(**name **=** "Product\_GEN"**,** sequenceName **=** "Product\_SEQ"**)**

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

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

**}**

@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**;**

@Entity

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

public class Category **{**

@Id

@GeneratedValue**(**strategy **=** GenerationType**.**SEQUENCE**,** generator **=** "Category\_GEN"**)**

@SequenceGenerator**(**name **=** "Category\_GEN"**,** sequenceName **=** "Category\_SEQ"**)**

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.

create table Categories (

categoryID integer not null,

name varchar(255),

primary key (categoryID)

)

alter table Products

add column categoryID integer

alter table Products

add constraint FKn4dvny5ajgqgw20l5nb7imd5t

foreign key (categoryID)

references Categories

/\*

from

Product \*/ select

product0\_.productID as producti1\_1\_,

product0\_.categoryID as category4\_1\_,

product0\_.productName as productn2\_1\_,

product0\_.supplierID as supplier5\_1\_,

product0\_.unitsInStock as unitsins3\_1\_

from

Products product0\_

select

supplier0\_.supplierID as supplier1\_2\_0\_,

supplier0\_.city as city2\_2\_0\_,

supplier0\_.companyName as companyn3\_2\_0\_,

supplier0\_.street as street4\_2\_0\_

from

Suppliers supplier0\_

where

supplier0\_.supplierID=?

select

supplier0\_.supplierID as supplier1\_2\_0\_,

supplier0\_.city as city2\_2\_0\_,

supplier0\_.companyName as companyn3\_2\_0\_,

supplier0\_.street as street4\_2\_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=?

/\*

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)**

Obraz zawierający tekst, ściana

Opis wygenerowany automatycznie

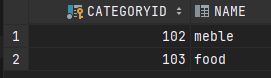
#### **Utworzone/zmodyfikowane tabele**

#### **Tabela Products**

Obraz zawierający tekst, monitor, zrzut ekranu, ekran

Opis wygenerowany automatycznie

#### **Tabela Categories**



### Relacja wiele do wielu (dodanie faktury)

#### **Zaimplementowane klasy**

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

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

Query query**;**

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

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

// Create new products and assign a category to them

Category electronics **=** **new** Category**(**"elektronika"**);**

Product smartphone **=** **new** Product**(**"Smartfon"**,** electronics**,** 321**);**

Product tablet **=** **new** Product**(**"Tablet"**,** electronics**,** 123**);**

electronics**.**addProduct**(**smartphone**);**

electronics**.**addProduct**(**tablet**);**

// Sell existing products

Invoice invoice1 **=** **new** Invoice**();**

int soldCount **=** 55**;**

query **=** session**.**createQuery**(**"from Product where unitsInStock >" **+** soldCount**);**

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

products**.**forEach**(**p **->** **{**

**try** **{**

p**.**sell**(**invoice1**,** soldCount**);**

**}** **catch** **(**InvalidAttributeValueException e**)** **{**

e**.**printStackTrace**();**

**}**

**});**

// Sell new products

smartphone**.**sell**(**invoice1**,** 3**);**

Invoice invoice2 **=** **new** Invoice**();**

smartphone**.**sell**(**invoice2**,** 2**);**

smartphone**.**sell**(**invoice2**,** 1**);**

session**.**save**(**invoice1**);**

session**.**save**(**invoice2**);**

session**.**save**(**electronics**);**

session**.**save**(**smartphone**);**

session**.**save**(**tablet**);**

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

// Tests

// Get the list of products sold in the specific invoice (I'm checking all invoices)

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

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

Invoice invoice **=** **((**Invoice**)** i**);**

List**<**Product**>** products\_ **=** **(**List**<**Product**>)** invoice**.**getProducts**();**

System**.**out**.**println**(**"\nFaktura numer: " **+** i **+** ": "**);**

System**.**out**.**println**(**"\tŁączna liczba produktów: " **+** invoice**.**getQuantity**());**

products\_**.**forEach**(**p **->** System**.**out**.**println**(**"\t- " **+** p**.**getName**()** **+** ","**));**

**if** **(**products\_**.**size**()** **==** 0**)** System**.**out**.**println**(**"\tBrak"**);**

**});**

// Get all invoices for a product

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

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

Product product **=** **((**Product**)** p**);**

List**<**Invoice**>** invoices **=** **(**List**<**Invoice**>)** product**.**getInvoices**();**

System**.**out**.**println**(**"\nFaktury, na których występuje produkt: " **+** product**.**getName**()** **+** ": "**);**

invoices**.**forEach**(**i **->** System**.**out**.**println**(**"\t- " **+** i **+** ","**));**

**if** **(**invoices**.**size**()** **==** 0**)** System**.**out**.**println**(**"\tBrak"**);**

**});**

**}**

**}**

#### **Klasa Product**

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

**import** javax**.**management**.**InvalidAttributeValueException**;**

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

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

**import** java**.**util**.**HashSet**;**

@Entity

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

public class Product **{**

@Id

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

@SequenceGenerator**(**name **=** "Product\_GEN"**,** sequenceName **=** "Product\_SEQ"**)**

private int productID**;**

private String productName**;**

private int unitsInStock**;**

@ManyToOne

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

private Supplier supplier**;**

@ManyToOne

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

private Category category**;**

@ManyToMany**(**mappedBy **=** "products"**)**

private Collection**<**Invoice**>** invoices **=** **new** HashSet**<>();**

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

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

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

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

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

**}**

@Override

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

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

**}**

public String getName**()** **{**

**return** productName**;**

**}**

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**;**

**}**

public Collection**<**Invoice**>** getInvoices**()** **{**

**return** invoices**;**

**}**

public void sell**(**Invoice invoice**,** int quantity**)** **throws** InvalidAttributeValueException **{**

**if** **(**unitsInStock **<** quantity**)** **{**

**throw** **new** InvalidAttributeValueException**(**"Unable to sell " **+** quantity **+** " products"**);**

**}**

unitsInStock **-=** quantity**;**

invoice**.**addProduct**(this,** quantity**);**

invoices**.**add**(**invoice**);**

**}**

**}**

#### **Klasa Invoice**

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

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

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

**import** java**.**util**.**HashSet**;**

@Entity

public class Invoice **{**

@Id

@GeneratedValue**(**strategy **=** GenerationType**.**SEQUENCE**,** generator **=** "Invoice\_GEN"**)**

@SequenceGenerator**(**name **=** "Invoice\_GEN"**,** sequenceName **=** "Invoice\_SEQ"**)**

private int invoiceNumber**;**

private int quantity **=** 0**;**

@ManyToMany

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

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

public Invoice**()** **{}**

@Override

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

**return** String**.**valueOf**(**invoiceNumber**);**

**}**

public void addProduct**(**Product product**,** int quantity**)** **{**

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

**this.**quantity **+=** quantity**;**

**}**

public int getInvoiceNumber**()** **{**

**return** invoiceNumber**;**

**}**

public int getQuantity**()** **{**

**return** quantity**;**

**}**

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

**return** products**;**

**}**

**}**

#### **Logi SQL**

Pomijam **DROP TABLE** oraz kod odpowiedzialny za tworzenie i korzystanie z sekwencji.

create table Invoice (

invoiceNumber integer not null,

quantity integer not null,

primary key (invoiceNumber)

)

create table Invoice\_Products (

invoices\_invoiceNumber integer not null,

products\_productID integer not null

)

alter table Invoice\_Products

add constraint FKs9kojcr3iu3dm7fww8n0v442n

foreign key (products\_productID)

references Products

alter table Invoice\_Products

add constraint FKlgae5neonlp88wdmxtb6qdppw

foreign key (invoices\_invoiceNumber)

references Invoice

/\*

from

Product

where

unitsInStock >55 \*/ select

product0\_.productID as producti1\_3\_,

product0\_.categoryID as category4\_3\_,

product0\_.productName as productn2\_3\_,

product0\_.supplierID as supplier5\_3\_,

product0\_.unitsInStock as unitsins3\_3\_

from

Products product0\_

where

product0\_.unitsInStock>55

select

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

category0\_.name as name2\_0\_0\_

from

Categories category0\_

where

category0\_.categoryID=?

select

supplier0\_.supplierID as supplier1\_4\_0\_,

supplier0\_.city as city2\_4\_0\_,

supplier0\_.companyName as companyn3\_4\_0\_,

supplier0\_.street as street4\_4\_0\_

from

Suppliers supplier0\_

where

supplier0\_.supplierID=?

select

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

category0\_.name as name2\_0\_0\_

from

Categories category0\_

where

category0\_.categoryID=?

/\* insert com.matipl01.Invoice

\*/ insert

into

Invoice

(quantity, invoiceNumber)

values

(?, ?)

/\* insert com.matipl01.Invoice

\*/ insert

into

Invoice

(quantity, invoiceNumber)

values

(?, ?)

/\* insert com.matipl01.Invoice

\*/ insert

into

Invoice

(quantity, invoiceNumber)

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=?

/\* insert collection

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

into

Invoice\_Products

(invoices\_invoiceNumber, products\_productID)

values

(?, ?)

/\* insert collection

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

into

Invoice\_Products

(invoices\_invoiceNumber, products\_productID)

values

(?, ?)

/\* insert collection

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

into

Invoice\_Products

(invoices\_invoiceNumber, products\_productID)

values

(?, ?)

/\* insert collection

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

into

Invoice\_Products

(invoices\_invoiceNumber, products\_productID)

values

(?, ?)

/\* insert collection

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

into

Invoice\_Products

(invoices\_invoiceNumber, products\_productID)

values

(?, ?)

/\*

from

Invoice \*/ select

invoice0\_.invoiceNumber as invoicen1\_1\_,

invoice0\_.quantity as quantity2\_1\_

from

Invoice invoice0\_

/\*

from

Product \*/ select

product0\_.productID as producti1\_3\_,

product0\_.categoryID as category4\_3\_,

product0\_.productName as productn2\_3\_,

product0\_.supplierID as supplier5\_3\_,

product0\_.unitsInStock as unitsins3\_3\_

from

Products product0\_

select

supplier0\_.supplierID as supplier1\_4\_0\_,

supplier0\_.city as city2\_4\_0\_,

supplier0\_.companyName as companyn3\_4\_0\_,

supplier0\_.street as street4\_4\_0\_

from

Suppliers supplier0\_

where

supplier0\_.supplierID=?

select

invoices0\_.products\_productID as products2\_2\_0\_,

invoices0\_.invoices\_invoiceNumber as invoices1\_2\_0\_,

invoice1\_.invoiceNumber as invoicen1\_1\_1\_,

invoice1\_.quantity as quantity2\_1\_1\_

from

Invoice\_Products invoices0\_

inner join

Invoice invoice1\_

on invoices0\_.invoices\_invoiceNumber=invoice1\_.invoiceNumber

where

invoices0\_.products\_productID=?

select

invoices0\_.products\_productID as products2\_2\_0\_,

invoices0\_.invoices\_invoiceNumber as invoices1\_2\_0\_,

invoice1\_.invoiceNumber as invoicen1\_1\_1\_,

invoice1\_.quantity as quantity2\_1\_1\_

from

Invoice\_Products invoices0\_

inner join

Invoice invoice1\_

on invoices0\_.invoices\_invoiceNumber=invoice1\_.invoiceNumber

where

invoices0\_.products\_productID=?

select

invoices0\_.products\_productID as products2\_2\_0\_,

invoices0\_.invoices\_invoiceNumber as invoices1\_2\_0\_,

invoice1\_.invoiceNumber as invoicen1\_1\_1\_,

invoice1\_.quantity as quantity2\_1\_1\_

from

Invoice\_Products invoices0\_

inner join

Invoice invoice1\_

on invoices0\_.invoices\_invoiceNumber=invoice1\_.invoiceNumber

where

invoices0\_.products\_productID=?

select

invoices0\_.products\_productID as products2\_2\_0\_,

invoices0\_.invoices\_invoiceNumber as invoices1\_2\_0\_,

invoice1\_.invoiceNumber as invoicen1\_1\_1\_,

invoice1\_.quantity as quantity2\_1\_1\_

from

Invoice\_Products invoices0\_

inner join

Invoice invoice1\_

on invoices0\_.invoices\_invoiceNumber=invoice1\_.invoiceNumber

where

invoices0\_.products\_productID=?

select

invoices0\_.products\_productID as products2\_2\_0\_,

invoices0\_.invoices\_invoiceNumber as invoices1\_2\_0\_,

invoice1\_.invoiceNumber as invoicen1\_1\_1\_,

invoice1\_.quantity as quantity2\_1\_1\_

from

Invoice\_Products invoices0\_

inner join

Invoice invoice1\_

on invoices0\_.invoices\_invoiceNumber=invoice1\_.invoiceNumber

where

invoices0\_.products\_productID=?

select

invoices0\_.products\_productID as products2\_2\_0\_,

invoices0\_.invoices\_invoiceNumber as invoices1\_2\_0\_,

invoice1\_.invoiceNumber as invoicen1\_1\_1\_,

invoice1\_.quantity as quantity2\_1\_1\_

from

Invoice\_Products invoices0\_

inner join

Invoice invoice1\_

on invoices0\_.invoices\_invoiceNumber=invoice1\_.invoiceNumber

where

invoices0\_.products\_productID=?

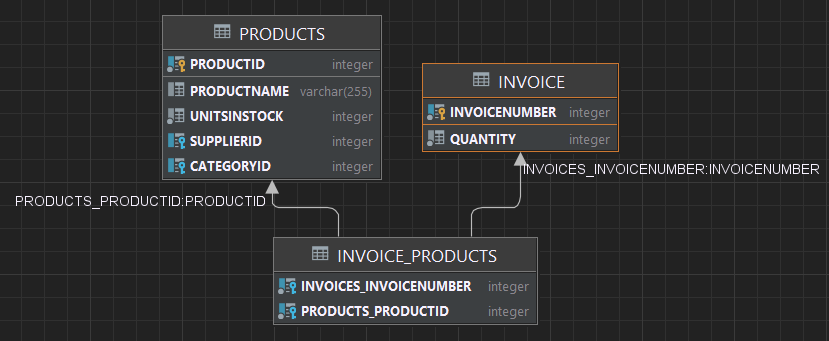
#### **Rezultat wykonania kodu**

Aby rezultat był lepiej widoczny, zmieniłem w pliku konfiguracyjnym wartość **show\_sql** z na **false**. Wówczas, otrzymujemy na konsoli poniższy rezultat:

Obraz zawierający tekst

Opis wygenerowany automatycznie

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



#### **Utworzone/zmodyfikowane tabele**

#### **Tabela Products**

Obraz zawierający tekst, monitor, zrzut ekranu, sprzęt elektroniczny

Opis wygenerowany automatycznie

#### **Tabela Categories**

Obraz zawierający stół

Opis wygenerowany automatycznie

### JPA

### Relacja wiele do wielu (faktura)

#### **Zaimplementowane klasy**

#### **Klasa Main**

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

**import** javax**.**management**.**InvalidAttributeValueException**;**

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

**import** java**.**util**.**List**;**

class Main **{**

private static final EntityManagerFactory emf**;**

static **{**

**try** **{**

emf **=** Persistence**.**createEntityManagerFactory**(**"derby"**);**

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

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

**}**

**}**

public static EntityManager getEntityManager**()** **{**

**return** emf**.**createEntityManager**();**

**}**

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

final EntityManager em **=** getEntityManager**();**

EntityTransaction etx **=** em**.**getTransaction**();**

Query query**;**

// Add products only

etx**.**begin**();**

Product product1 **=** **new** Product**(**"Smartfon"**,** 25**);**

Product product2 **=** **new** Product**(**"Tablet"**,** 45**);**

Product product3 **=** **new** Product**(**"Konsola"**,** 11**);**

em**.**persist**(**product1**);**

em**.**persist**(**product2**);**

em**.**persist**(**product3**);**

etx**.**commit**();**

etx**.**begin**();**

// Create invoices

Invoice invoice1 **=** **new** Invoice**();**

Invoice invoice2 **=** **new** Invoice**();**

// Sell existing products

int soldCount **=** 20**;**

query **=** em**.**createQuery**(**"from Product where unitsInStock >" **+** soldCount**);**

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

products**.**forEach**(**p **->** **{**

**if** **(((**Product**)** p**).**getUnitsInStock**()** **>=** soldCount**)** **{**

**try** **{**

**((**Product**)** p**).**sell**(**invoice1**,** soldCount**);**

**}** **catch** **(**InvalidAttributeValueException e**)** **{**

e**.**printStackTrace**();**

**}**

**}**

**});**

// Create the new products and sell them

product1 **=** **new** Product**(**"Mikrofalówka"**,** 4**);**

product2 **=** **new** Product**(**"Lodówka"**,** 14**);**

product3 **=** **new** Product**(**"Wirówka"**,** 17**);**

**try** **{**

product1**.**sell**(**invoice2**,** 3**);**

product2**.**sell**(**invoice1**,** 11**);**

product2**.**sell**(**invoice2**,** 2**);**

product3**.**sell**(**invoice2**,** 17**);**

**}** **catch** **(**InvalidAttributeValueException e**)** **{**

e**.**printStackTrace**();**

**}**

em**.**persist**(**invoice1**);**

em**.**persist**(**invoice2**);**

etx**.**commit**();**

// Tests

// Get the list of products sold in the specific invoice (I'm checking all invoices)

query **=** em**.**createQuery**(**"from Invoice"**);**

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

Invoice invoice **=** **((**Invoice**)** i**);**

List**<**Product**>** products\_ **=** **(**List**<**Product**>)** invoice**.**getProducts**();**

System**.**out**.**println**(**"\nFaktura numer: " **+** i **+** ": "**);**

System**.**out**.**println**(**"\tŁączna liczba produktów: " **+** invoice**.**getQuantity**());**

products\_**.**forEach**(**p **->** System**.**out**.**println**(**"\t- " **+** p**.**getName**()** **+** ","**));**

**if** **(**products\_**.**size**()** **==** 0**)** System**.**out**.**println**(**"\tBrak"**);**

**});**

// Get all invoices for a product

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

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

Product product **=** **((**Product**)** p**);**

List**<**Invoice**>** invoices **=** **(**List**<**Invoice**>)** product**.**getInvoices**();**

System**.**out**.**println**(**"\nFaktury, na których występuje produkt: " **+** product**.**getName**()** **+** ": "**);**

invoices**.**forEach**(**i **->** System**.**out**.**println**(**"\t- " **+** i **+** ","**));**

**if** **(**invoices**.**size**()** **==** 0**)** System**.**out**.**println**(**"\tBrak"**);**

**});**

em**.**close**();**

**}**

**}**

#### **Klasa Product**

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

**import** javax**.**management**.**InvalidAttributeValueException**;**

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

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

**import** java**.**util**.**HashSet**;**

@Entity

public class Product **{**

@Id

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

private int productID**;**

private String productName**;**

private int unitsInStock**;**

@ManyToMany**(**mappedBy **=** "products"**)**

private final Collection**<**Invoice**>** invoices **=** **new** HashSet**<>();**

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

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

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

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

**}**

@Override

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

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

**}**

public String getName**()** **{**

**return** productName**;**

**}**

public int getUnitsInStock**()** **{**

**return** unitsInStock**;**

**}**

public Collection**<**Invoice**>** getInvoices**()** **{**

**return** invoices**;**

**}**

public void sell**(**Invoice invoice**,** int quantity**)** **throws** InvalidAttributeValueException **{**

**if** **(**unitsInStock **<** quantity**)** **{**

**throw** **new** InvalidAttributeValueException**(**"Unable to sell " **+** quantity **+** " products"**);**

**}**

unitsInStock **-=** quantity**;**

invoice**.**addProduct**(this,** quantity**);**

invoices**.**add**(**invoice**);**

**}**

**}**

#### **Klasa Invoice**

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

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

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

**import** java**.**util**.**HashSet**;**

@Entity

public class Invoice **{**

@Id

@GeneratedValue**(**strategy **=** GenerationType**.**SEQUENCE**,** generator **=** "Invoice\_GEN"**)**

@SequenceGenerator**(**name **=** "Invoice\_GEN"**,** sequenceName **=** "Invoice\_SEQ"**)**

private int invoiceNumber**;**

private int quantity **=** 0**;**

@ManyToMany**(**cascade **=** CascadeType**.**PERSIST**)**

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

public Invoice**()** **{}**

@Override

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

**return** String**.**valueOf**(**invoiceNumber**);**

**}**

public void addProduct**(**Product product**,** int quantity**)** **{**

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

**this.**quantity **+=** quantity**;**

**}**

public int getInvoiceNumber**()** **{**

**return** invoiceNumber**;**

**}**

public int getQuantity**()** **{**

**return** quantity**;**

**}**

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

**return** products**;**

**}**

**}**

* 1. **Plik konfiguracyjny persistence.xml**

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

<persistence xmlns=**"http://java.sun.com/xml/ns/persistence"**

xmlns:xsi=**"http://www.w3.org/2001/XMLSchema-instance"**

xsi:schemaLocation=**"http://java.sun.com/xml/ns/persistence**

**http://java.sun.com/xml/ns/persistence/persistence\_2)0.xsd"** version=**"2.0"**>

<persistence-unit name=**"derby"** transaction-type=**"RESOURCE\_LOCAL"**>

<properties>

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

<property name=**"hibernate.connection.driver\_class"**

value=**"org.apache.derby.jdbc.ClientDriver"**/>

<property name=**"hibernate.connection.url"**

value=**"jdbc:derby://127.0.0.1/LopacinskiMateuszJPA"**/>

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

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

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

</properties>

</persistence-unit>

</persistence>

#### **Logi SQL**

Pomijam **DROP TABLE** oraz kod odpowiedzialny za tworzenie i korzystanie z sekwencji.

create table Invoice (

invoiceNumber integer not null,

quantity integer not null,

primary key (invoiceNumber)

)

create table Invoice\_Product (

invoices\_invoiceNumber integer not null,

products\_productID integer not null

)

create table Product (

productID integer not null,

productName varchar(255),

unitsInStock integer not null,

primary key (productID)

)

alter table Invoice\_Product

add constraint FK2mn08nt19nrqagr12grh5uho0

foreign key (products\_productID)

references Product

alter table Invoice\_Product

add constraint FKcbqyl9u4eh1tws13u6pk5j2nt

foreign key (invoices\_invoiceNumber)

references Invoice

insert

into

Product

(productName, unitsInStock, productID)

values

(?, ?, ?)

insert

into

Product

(productName, unitsInStock, productID)

values

(?, ?, ?)

insert

into

Product

(productName, unitsInStock, productID)

values

(?, ?, ?)

select

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

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

product0\_.unitsInStock as unitsins3\_2\_

from

Product product0\_

where

product0\_.unitsInStock>20

insert

into

Invoice

(quantity, invoiceNumber)

values

(?, ?)

insert

into

Product

(productName, unitsInStock, productID)

values

(?, ?, ?)

insert

into

Invoice

(quantity, invoiceNumber)

values

(?, ?)

insert

into

Product

(productName, unitsInStock, productID)

values

(?, ?, ?)

insert

into

Product

(productName, unitsInStock, productID)

values

(?, ?, ?)

update

Product

set

productName=?,

unitsInStock=?

where

productID=?

update

Product

set

productName=?,

unitsInStock=?

where

productID=?

insert

into

Invoice\_Product

(invoices\_invoiceNumber, products\_productID)

values

(?, ?)

insert

into

Invoice\_Product

(invoices\_invoiceNumber, products\_productID)

values

(?, ?)

insert

into

Invoice\_Product

(invoices\_invoiceNumber, products\_productID)

values

(?, ?)

insert

into

Invoice\_Product

(invoices\_invoiceNumber, products\_productID)

values

(?, ?)

insert

into

Invoice\_Product

(invoices\_invoiceNumber, products\_productID)

values

(?, ?)

insert

into

Invoice\_Product

(invoices\_invoiceNumber, products\_productID)

values

(?, ?)

select

invoice0\_.invoiceNumber as invoicen1\_0\_,

invoice0\_.quantity as quantity2\_0\_

from

Invoice invoice0\_

#### **Rezultat wykonania kodu**

Obraz zawierający tekst

Opis wygenerowany automatycznie

Obraz zawierający tekst

Opis wygenerowany automatycznie

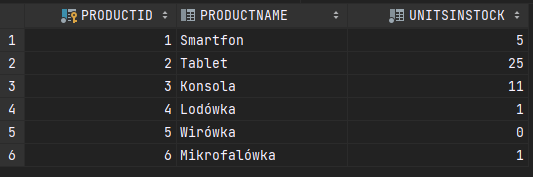
#### **Diagram bazy danych**

Obraz zawierający tekst, tablica wyników, zrzut ekranu

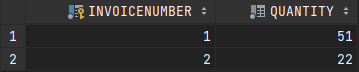
Opis wygenerowany automatycznie

#### **Utworzone/zmodyfikowane tabele**

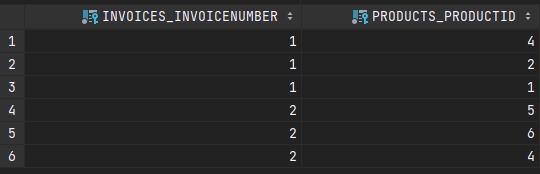
#### **Tabela Product**



#### **Tabela Invoice**

****

#### **Tabela Invoice\_Product**



### Kaskady

#### **Zaimplementowane klasy**

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

Ponownie zmieniona została tylko metoda public static void main**(**String**[]** args**)**, dlatego poniżej umieszczam tylko kod tej metody.

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

EntityManager em **=** getEntityManager**();**

EntityTransaction etx **=** em**.**getTransaction**();**

// Create products

Product product1 **=** **new** Product**(**"Smartfon"**,** 25**);**

Product product2 **=** **new** Product**(**"Tablet"**,** 45**);**

Product product3 **=** **new** Product**(**"Konsola"**,** 11**);**

Product product4 **=** **new** Product**(**"Smartwatch"**,** 32**);**

// Create invoices

Invoice invoice1 **=** **new** Invoice**();**

Invoice invoice2 **=** **new** Invoice**();**

// Add products to invoices

etx**.**begin**();**

invoice1**.**addProduct**(**product1**,** 5**);**

invoice2**.**addProduct**(**product3**,** 7**);**

invoice1**.**addProduct**(**product2**,** 8**);**

invoice1**.**addProduct**(**product3**,** 7**);**

em**.**persist**(**invoice1**);**

em**.**persist**(**invoice2**);**

etx**.**commit**();**

// Add invoices to products

etx**.**begin**();**

product1**.**sell**(**invoice1**,** 11**);**

product2**.**sell**(**invoice2**,** 12**);**

product3**.**sell**(**invoice2**,** 3**);**

product4**.**sell**(**invoice1**,** 7**);**

em**.**persist**(**product1**);**

em**.**persist**(**product2**);**

em**.**persist**(**product3**);**

em**.**persist**(**product4**);**

etx**.**commit**();**

em**.**close**();**

**}**

#### **Klasa Product**

Zmiana dotyczyła jedynie dekoratora @ManyToMany oraz metody public void sell**(**Invoice invoice**,** int quantity**)**.

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

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

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

**import** java**.**util**.**HashSet**;**

@Entity

public class Product **{**

@Id

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

@SequenceGenerator**(**name **=** "Product\_GEN"**,** sequenceName **=** "Product\_SEQ"**)**

private int productID**;**

private String productName**;**

private int unitsInStock**;**

@ManyToMany**(**

cascade **=** CascadeType**.**PERSIST**,**

fetch **=** FetchType**.**EAGER**,**

mappedBy **=** "products"

**)**

private final Collection**<**Invoice**>** invoices **=** **new** HashSet**<>();**

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

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

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

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

**}**

@Override

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

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

**}**

public String getName**()** **{**

**return** productName**;**

**}**

public Collection**<**Invoice**>** getInvoices**()** **{**

**return** invoices**;**

**}**

public int getUnitsInStock**()** **{**

**return** unitsInStock**;**

**}**

public void sell**(**Invoice invoice**,** int quantity**)** **throws** IllegalArgumentException **{**

**if** **(**unitsInStock **<** quantity**)** **{**

**throw** **new** IllegalArgumentException**(**"Unable to sell " **+** quantity **+** " products"**);**

**}**

invoices**.**add**(**invoice**);**

invoice**.**updateQuantity**(**quantity**);**

**}**

**}**

#### **Klasa Invoice**

Zmiana dotyczyła jedynie dekoratora @ManyToMany.

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

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

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

**import** java**.**util**.**HashSet**;**

@Entity

public class Invoice **{**

@Id

@GeneratedValue**(**strategy **=** GenerationType**.**SEQUENCE**,** generator **=** "Invoice\_GEN"**)**

@SequenceGenerator**(**name **=** "Invoice\_GEN"**,** sequenceName **=** "Invoice\_SEQ"**)**

private int invoiceNumber**;**

private int quantity **=** 0**;**

@ManyToMany**(**cascade **=** CascadeType**.**PERSIST**)**

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

public Invoice**()** **{}**

@Override

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

**return** String**.**valueOf**(**invoiceNumber**);**

**}**

public void addProduct**(**Product product**,** int quantity**)** **throws** IllegalArgumentException **{**

**if** **(**product**.**getUnitsInStock**()** **<** quantity**)** **{**

**throw** **new** IllegalArgumentException**(**"Unable to sell " **+** quantity **+** " products"**);**

**}**

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

**this.**quantity **+=** quantity**;**

**}**

public int getQuantity**()** **{**

**return** quantity**;**

**}**

public void updateQuantity**(**int quantity**)** **{**

**this.**quantity **+=** quantity**;**

**}**

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

**return** products**;**

**}**

**}**

#### **Logi SQL**

Pomijam **DROP TABLE** oraz kod odpowiedzialny za tworzenie i korzystanie z sekwencji.

create table Invoice (

invoiceNumber integer not null,

quantity integer not null,

primary key (invoiceNumber)

)

create table Invoice\_Product (

invoices\_invoiceNumber integer not null,

products\_productID integer not null

)

create table Product (

productID integer not null,

productName varchar(255),

unitsInStock integer not null,

primary key (productID)

)

alter table Invoice\_Product

add constraint FK2mn08nt19nrqagr12grh5uho0

foreign key (products\_productID)

references Product

alter table Invoice\_Product

add constraint FKcbqyl9u4eh1tws13u6pk5j2nt

foreign key (invoices\_invoiceNumber)

references Invoice

insert

into

Invoice

(quantity, invoiceNumber)

values

(?, ?)

insert

into

Product

(productName, unitsInStock, productID)

values

(?, ?, ?)

insert

into

Product

(productName, unitsInStock, productID)

values

(?, ?, ?)

insert

into

Product

(productName, unitsInStock, productID)

values

(?, ?, ?)

insert

into

Invoice

(quantity, invoiceNumber)

values

(?, ?)

insert

into

Invoice\_Product

(invoices\_invoiceNumber, products\_productID)

values

(?, ?)

insert

into

Invoice\_Product

(invoices\_invoiceNumber, products\_productID)

values

(?, ?)

insert

into

Invoice\_Product

(invoices\_invoiceNumber, products\_productID)

values

(?, ?)

insert

into

Invoice\_Product

(invoices\_invoiceNumber, products\_productID)

values

(?, ?)

insert

into

Product

(productName, unitsInStock, productID)

values

(?, ?, ?)

update

Invoice

set

quantity=?

where

invoiceNumber=?

update

Invoice

set

quantity=?

where

invoiceNumber=?

#### **Diagram bazy danych**

Obraz zawierający tekst, tablica wyników

Opis wygenerowany automatycznie

#### **Utworzone/zmodyfikowane tabele**

#### **Tabela Product**

Obraz zawierający tekst, sprzęt elektroniczny, tablica wyników

Opis wygenerowany automatycznie

#### **Tabela Invoice**

****

#### **Tabela Invoice\_Product**



### Embedded class

#### **9a. Adresy wbudowane w tabelę dostawców**

#### **Zaimplementowane klasy**

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

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

EntityManager em **=** getEntityManager**();**

EntityTransaction etx **=** em**.**getTransaction**();**

// Create Suppliers

Supplier supplier1 **=** **new** Supplier**(**"Owocowy Raj"**,** **new** Address**(**"Kraków"**,** "Miodowa"**));**

Supplier supplier2 **=** **new** Supplier**(**"Elektro"**,** **new** Address**(**"Warszawa"**,** "Długa"**));**

// Save suppliers

etx**.**begin**();**

em**.**persist**(**supplier1**);**

em**.**persist**(**supplier2**);**

etx**.**commit**();**

em**.**close**();**

**}**

#### **Klasa Supplier**

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

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

@Entity

public class Supplier **{**

@Id

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

@SequenceGenerator**(**name **=** "Supplier\_GEN"**,** sequenceName **=** "Supplier\_SEQ"**)**

public int supplierID**;**

private String companyName**;**

@Embedded

private Address address**;**

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

public Supplier**(**String companyName**,** Address address**)** **{**

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

**this.**address **=** address**;**

**}**

@Override

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

**return** companyName**;**

**}**

public void setAddress**(**Address address**)** **{**

**this.**address **=** address**;**

**}**

public Address getAddress**()** **{**

**return** address**;**

**}**

**}**

#### **Klasa Address**

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

**import** javax**.**persistence**.**Embeddable**;**

@Embeddable

public class Address **{**

private String street**;**

private String city**;**

public Address**()** **{}**

public Address**(**String street**,** String city**)** **{**

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

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

**}**

@Override

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

**return** city **+** ", ul. " **+** street**;**

**}**

public String getStreet**()** **{**

**return** street**;**

**}**

public String getCity**()** **{**

**return** city**;**

**}**

public void setStreet**(**String street**)** **{**

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

**}**

public void setCity**(**String city**)** **{**

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

**}**

**}**

#### **Logi SQL**

Pomijam **DROP TABLE** oraz kod odpowiedzialny za tworzenie i korzystanie z sekwencji.

create table Supplier (

supplierID integer not null,

city varchar(255),

street varchar(255),

companyName varchar(255),

primary key (supplierID)

)

insert

into

Supplier

(city, street, companyName, supplierID)

values

(?, ?, ?, ?)

insert

into

Supplier

(city, street, companyName, supplierID)

values

(?, ?, ?, ?)

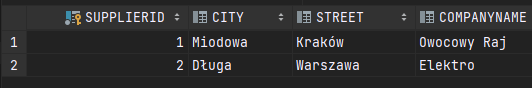
#### **Diagram bazy danych**

Obraz zawierający tekst, tablica wyników, zrzut ekranu

Opis wygenerowany automatycznie

#### **Utworzone/zmodyfikowane tabele**

#### **Tabela Supplier**



#### **9b. Adresy w osobnej tabeli**

#### **Zaimplementowane klasy**

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

Identyczna jak poprzednio.

#### **Klasa Supplier**

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

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

@Entity

@SecondaryTable**(**name **=** "Address"**)**

public class Supplier **{**

@Id

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

@SequenceGenerator**(**name **=** "Supplier\_GEN"**,** sequenceName **=** "Supplier\_SEQ"**)**

public int supplierID**;**

private String companyName**;**

@Column**(**table **=** "Address"**)**

private String city**;**

@Column**(**table **=** "Address"**)**

private String street**;**

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

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

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

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

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

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

**}**

@Override

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

**return** companyName**;**

**}**

public String getCity**()** **{**

**return** city**;**

**}**

public String getStreet**()** **{**

**return** street**;**

**}**

**}**

#### **Logi SQL**

Pomijam **DROP TABLE** oraz kod odpowiedzialny za tworzenie i korzystanie z sekwencji.

create table Address (

id integer not null,

city varchar(255),

street varchar(255),

primary key (id)

);

create table Supplier (

supplierID integer not null,

companyName varchar(255),

address\_id integer,

primary key (supplierID)

);

alter table Supplier

add constraint FKcbqyl9u4eh1tws13u6pk5j2nt

foreign key (address\_id)

references Address

insert

into

Address

(city, street, id)

values

(?, ?, ?)

insert

into

Supplier

(companyName, address\_id, supplierID)

values

(?, ?, ?)

insert

into

Address

(city, street, id)

values

(?, ?, ?)

insert

into

Supplier

(companyName, address\_id, supplierID)

values

(?, ?, ?)

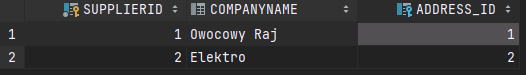
#### **Diagram bazy danych**

Obraz zawierający tekst, tablica wyników

Opis wygenerowany automatycznie

#### **Utworzone/zmodyfikowane tabele**

#### **Tabela Supplier**



#### **Tabela Address**



### Dziedziczenie

#### **10a. Type Per Class**

#### **Zaimplementowane klasy**

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

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

EntityManager em **=** getEntityManager**();**

EntityTransaction etx **=** em**.**getTransaction**();**

// Save suppliers

etx**.**begin**();**

Customer customer1 **=** **new** Customer**(**"Klient 1"**,** "Hofmana Vlastimila"**,** "Kraków"**,** "30‑210"**,** 5.5**);**

Customer customer2 **=** **new** Customer**(**"Klient 2"**,** "3 Maja Al."**,** "Kraków"**,** "30‑063"**,** 9.75**);**

Supplier supplier1 **=** **new** Supplier**(**"Dostawca 1"**,** "Mikołaja Kopernika 3"**,** "Warszawa"**,** "00-367"**,** "123123123123123123123123"**);**

Supplier supplier2 **=** **new** Supplier**(**"Dostawca 2"**,** "Oboźna"**,** "Kraków"**,** "30-011"**,** "999888777666555444333222"**);**

em**.**persist**(**customer1**);**

em**.**persist**(**customer2**);**

em**.**persist**(**supplier1**);**

em**.**persist**(**supplier2**);**

etx**.**commit**();**

em**.**close**();**

**}**

#### **Klasa Company**

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

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

@Entity

@Inheritance**(**strategy **=** InheritanceType**.**TABLE\_PER\_CLASS**)**

public abstract class Company **{**

@Id

@GeneratedValue**(**strategy **=** GenerationType**.**SEQUENCE**,** generator **=** "Company\_GEN"**)**

@SequenceGenerator**(**name **=** "Company\_GEN"**,** sequenceName **=** "Company\_SEQ"**)**

private int companyID**;**

private String companyName**;**

private String street**;**

private String city**;**

private String zipCode**;**

public Company**()** **{}**

public Company**(**String companyName**,** String street**,** String city**,** String zipCode**)** **{**

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

**this.**zipCode **=** zipCode**;**

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

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

**}**

@Override

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

**return** companyName**;**

**}**

public String getCompanyName**()** **{**

**return** companyName**;**

**}**

public String getStreet**()** **{**

**return** street**;**

**}**

public String getCity**()** **{**

**return** city**;**

**}**

public String getZipCode**()** **{**

**return** zipCode**;**

**}**

**}**

#### **Klasa Supplier**

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

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

@Entity

public class Supplier **extends** Company **{**

private String bankAccountNumber**;**

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

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

**super(**companyName**,** street**,** city**,** zipCode**);**

**this.**bankAccountNumber **=** bankAccountNumber**;**

**}**

**}**

#### **Klasa Customer**

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

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

@Entity

public class Customer **extends** Company **{**

private double discount**;** // %

public Customer**()** **{}**

public Customer**(**String companyName**,** String street**,** String city**,** String zipCode**,** double discount**)** **{**

**super(**companyName**,** street**,** city**,** zipCode**);**

**this.**discount **=** discount**;**

**}**

**}**

#### **Logi SQL**

Pomijam **DROP TABLE** oraz kod odpowiedzialny za tworzenie i korzystanie z sekwencji.

create table Customer (

companyID integer not null,

city varchar(255),

companyName varchar(255),

street varchar(255),

zipCode varchar(255),

discount double not null,

primary key (companyID)

)

create table Supplier (

companyID integer not null,

city varchar(255),

companyName varchar(255),

street varchar(255),

zipCode varchar(255),

bankAccountNumber varchar(255),

primary key (companyID)

)

insert

into

Customer

(city, companyName, street, zipCode, discount, companyID)

values

(?, ?, ?, ?, ?, ?)

insert

into

Customer

(city, companyName, street, zipCode, discount, companyID)

values

(?, ?, ?, ?, ?, ?)

insert

into

Supplier

(city, companyName, street, zipCode, bankAccountNumber, companyID)

values

(?, ?, ?, ?, ?, ?)

insert

into

Supplier

(city, companyName, street, zipCode, bankAccountNumber, companyID)

values

(?, ?, ?, ?, ?, ?)

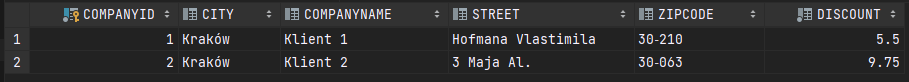
#### **Diagram bazy danych**

Obraz zawierający tekst, tablica wyników, zrzut ekranu

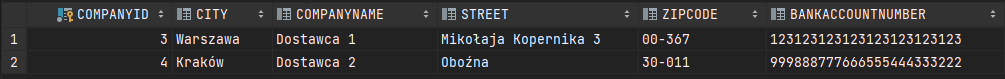
Opis wygenerowany automatycznie

#### **Utworzone/zmodyfikowane tabele**

#### **Tabela Customer**



#### **Tabela Supplier**



#### **10b. Single Table**

#### **Zaimplementowane klasy**

Takie same, jak poprzednio. Zmieniony został jedynie dekorator klasy public abstract class Company na @Inheritance**(**strategy **=** InheritanceType**.**SINGLE\_TABLE**)**.

#### **Logi SQL**

Pomijam **DROP TABLE** oraz kod odpowiedzialny za tworzenie i korzystanie z sekwencji.

create table Company (

DTYPE varchar(31) not null,

companyID integer not null,

city varchar(255),

companyName varchar(255),

street varchar(255),

zipCode varchar(255),

discount double,

bankAccountNumber varchar(255),

primary key (companyID)

)

insert

into

Company

(city, companyName, street, zipCode, discount, DTYPE, companyID)

values

(?, ?, ?, ?, ?, 'Customer', ?)

insert

into

Company

(city, companyName, street, zipCode, discount, DTYPE, companyID)

values

(?, ?, ?, ?, ?, 'Customer', ?)

insert

into

Company

(city, companyName, street, zipCode, bankAccountNumber, DTYPE, companyID)

values

(?, ?, ?, ?, ?, 'Supplier', ?)

insert

into

Company

(city, companyName, street, zipCode, bankAccountNumber, DTYPE, companyID)

values

(?, ?, ?, ?, ?, 'Supplier', ?)

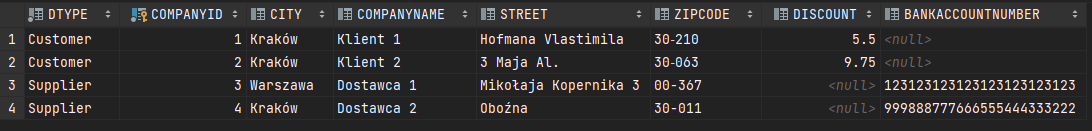
#### **Diagram bazy danych**

Obraz zawierający tekst

Opis wygenerowany automatycznie

#### **Utworzone/zmodyfikowane tabele**

#### **Tabela Company**



#### **10c. Joined**

#### **Zaimplementowane klasy**

Takie same, jak poprzednio. Zmieniony został jedynie dekorator klasy public abstract class Company na @Inheritance**(**strategy **=** InheritanceType**.**JOINED**)**.

#### **Logi SQL**

Pomijam **DROP TABLE** oraz kod odpowiedzialny za tworzenie i korzystanie z sekwencji.

create table Company (

companyID integer not null,

city varchar(255),

companyName varchar(255),

street varchar(255),

zipCode varchar(255),

primary key (companyID)

)

create table Customer (

discount double not null,

companyID integer not null,

primary key (companyID)

)

create table Supplier (

bankAccountNumber varchar(255),

companyID integer not null,

primary key (companyID)

)

alter table Customer

add constraint FKn7fvr687iixps0s6i5casr6f3

foreign key (companyID)

references Company

alter table Supplier

add constraint FKpinunrb4v5p4aemt2k4fnkjp8

foreign key (companyID)

references Company

insert

into

Company

(city, companyName, street, zipCode, companyID)

values

(?, ?, ?, ?, ?)

insert

into

Customer

(discount, companyID)

values

(?, ?)

insert

into

Company

(city, companyName, street, zipCode, companyID)

values

(?, ?, ?, ?, ?)

insert

into

Customer

(discount, companyID)

values

(?, ?)

insert

into

Company

(city, companyName, street, zipCode, companyID)

values

(?, ?, ?, ?, ?)

insert

into

Supplier

(bankAccountNumber, companyID)

values

(?, ?)

insert

into

Company

(city, companyName, street, zipCode, companyID)

values

(?, ?, ?, ?, ?)

insert

into

Supplier

(bankAccountNumber, companyID)

values

(?, ?)

#### **Diagram bazy danych**

Obraz zawierający tekst, tablica wyników, zrzut ekranu

Opis wygenerowany automatycznie

#### **Utworzone/zmodyfikowane tabele**

#### **Tabela Company**

Obraz zawierający tekst

Opis wygenerowany automatycznie

#### **Tabela Customer**



#### **Tabela Supplier**

