Sprawozdanie - Hibernate

Autor: Adrian Żerebiec

Spis treści

1. Zadanie I – zadanie z laboratorium

1.1 Klasa Main

```
package org.azerebiec;

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();
        }
    }
}
```

1.2 Klasa Main

```
package org.azerebiec;

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 UnitsOnStock;

    public Product() {}

    public Product(String productName, int unitsInStock) {
        this.ProductName = productName;
        this.UnitsOnStock = unitsInStock;
    }
}
```

1.3 Plik pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>org.example</groupId>
    <artifactId>AZerebiecJPALAB</artifactId>
    <version>1.0-SNAPSHOT</version>
    <dependencies>
        <dependency>
            <groupId>org.hibernate</groupId>
            <artifactId>hibernate-core</artifactId>
```

```
            <version>5.6.0.Final</version>
        </dependency>

<dependency><groupId>org.apache.derby</groupId><artifactId>der
byclient</artifactId><version>10.4.2.0</version></dependency>
    </dependencies>
    <properties>
        <maven.compiler.source>19</maven.compiler.source>
        <maven.compiler.target>19</maven.compiler.target>
        <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
    </properties>

</project>
```

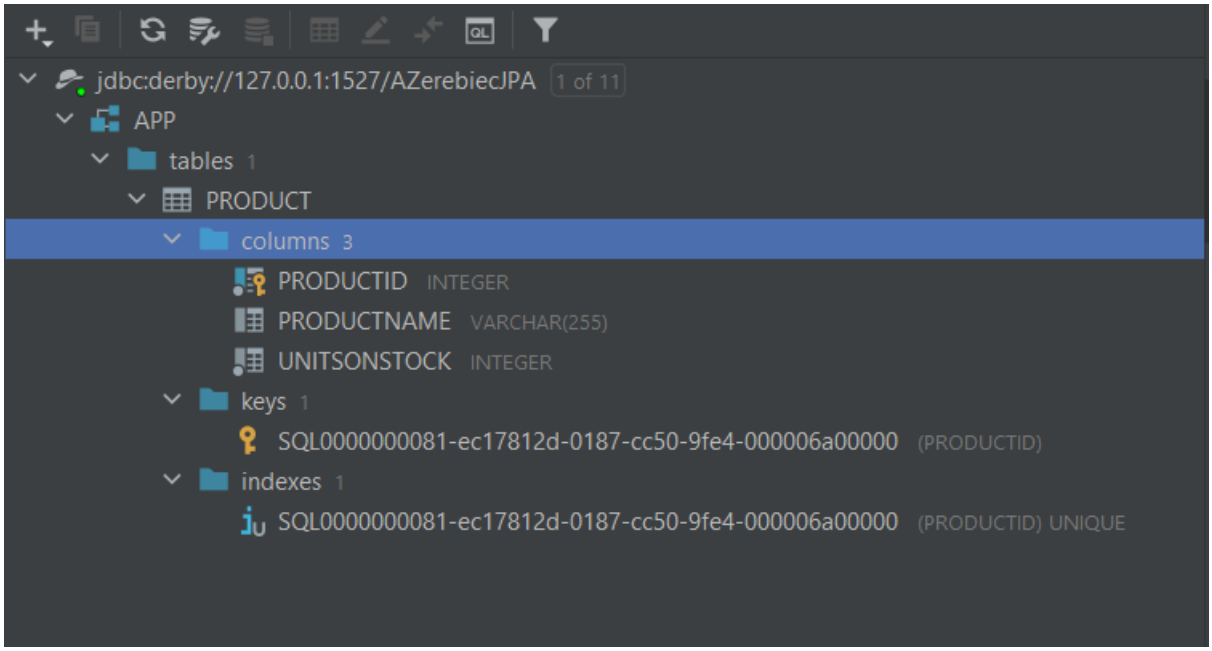
1.4 Plik hibernate.cfg.xml

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
    "-//Hibernate/Hibernate Configuration DTD//EN"
    "http://www.hibernate.org/dtd/hibernate-configuration-
3.0.dtd">
<hibernate-configuration>
  <session-factory>
    <property
name="connection.url">jdbc:derby://127.0.0.1/AZerebiecJPA;crea
te=true</property>
    <property
name="connection.driver_class">org.apache.derby.jdbc.ClientDri
ver</property>
    <!-- DB schema will be updated if needed -->
    <property name="hibernate.hbm2ddl.auto">update</property>
    <property name="show_sql">true</property>
    <property name="format_sql">true</property>
    <mapping class="org.azerebiec.Product"></mapping>
  </session-factory>
</hibernate-configuration>
```

1.5 Tabela Product

	PRODUCTID	PRODUCTNAME	UNITSONSTOCK
1	1	Krzesło	111

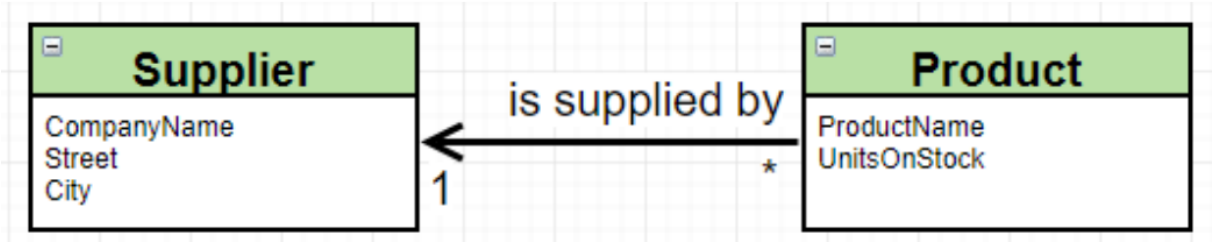
1.6 Podłączenie do bazy



2. Zadanie domowe

2.1 Zadanie II

Zmodyfikuj model wprowadzając pojęcie Dostawcy jak poniżej



2.1.1 Klasa Supplier

```
package org.azerebiec;

import javax.persistence.*;

@Entity
public class Supplier {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int supplierID;
    private String companyName;
    private String street;
    private String city;

    public Supplier(){};
    public Supplier(String companyName, String city,String
```

```
street){
        this.city = city;
        this.street  = street;
        this.companyName = companyName;
    }
    @Override
    public String toString(){
        return "Supplier: " + companyName;
    }
```

2.1.2 Klasa Product

```
package org.azerebiec;

import javax.persistence.*;

@Entity
public class Product {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int productID;
    private String productName;
    private int unitsOnStock;

    @ManyToOne
    @JoinColumn(name="supplierID")
    private Supplier supplier;

    public Product(String productName, int unitsOnStock) {
        this.productName = productName;
        this.unitsOnStock = unitsOnStock;
    }
    public Product() {}


    public void setSupplier(Supplier supplier){
        this.supplier = supplier;
    }

    @Override
    public String toString(){
        return "Product: " + productName;
    }
}
```

2.1.3 Klasa Main

```
package org.azerebiec;

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();

        try {
            Transaction tx = session.beginTransaction();
            Product product = session.find(Product.class,1);
            Supplier supplier = new Supplier("Nowy
dostawca","Lublin","Chłodna");
            session.save(supplier);
            product.setSupplier(supplier);
            session.save(product);
            tx.commit();

        } finally {
            session.close();
        }
    }
}
```

2.1.4 Plik hibernate.cfg.xml

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
    "-//Hibernate/Hibernate Configuration DTD//EN"
    "http://www.hibernate.org/dtd/hibernate-configuration-
3.0.dtd">
<hibernate-configuration>
  <session-factory>
    <property
name="connection.url">jdbc:derby://127.0.0.1/AZerebiecJPA;crea
te=true</property>
    <property
name="connection.driver_class">org.apache.derby.jdbc.ClientDri
ver</property>
    <property
name="dialect">org.hibernate.dialect.DerbyTenSevenDialect</pro
perty>


    <property name="show_sql">true</property>
    <property name="format_sql">true</property>
    <property name="use_sql_comments">true</property>
    <property name="hibernate.hbm2ddl.auto">update</property>

    <mapping class="org.azerebiec.Product"/>
    <mapping class="org.azerebiec.Supplier"/>
  </session-factory>
</hibernate-configuration>
```

2.1.5 Logi SQL

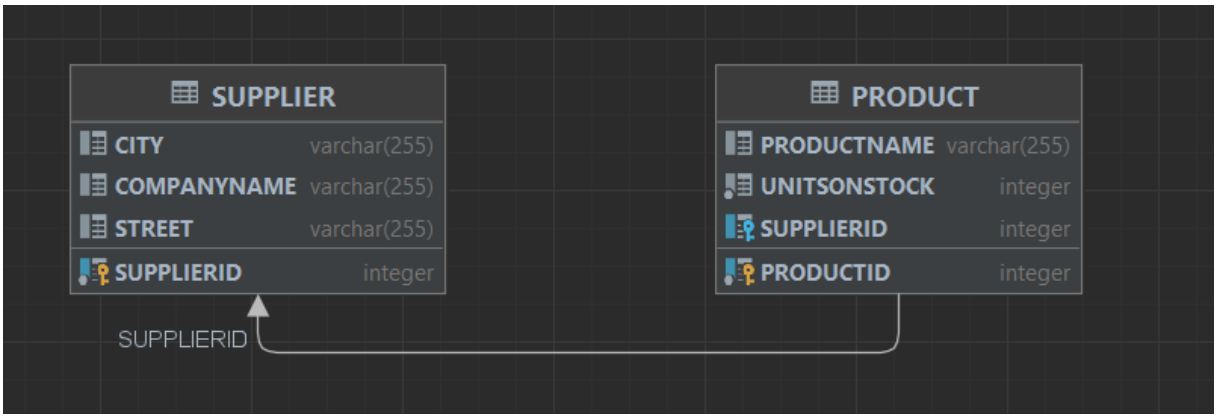
```
Hibernate:

    create table Product (
       productID integer not null,
        productName varchar(255),
        unitsOnStock integer not null,
        supplierID integer,
        primary key (productID)
    )
```

```
Hibernate:

    create table Supplier (
       supplierID integer not null,
        city varchar(255),
        companyName varchar(255),
        street varchar(255),
        primary key (supplierID)
    )
Hibernate:

    alter table Product
       add constraint FKj0x097f8xajoy9j9ryct9pf3o
       foreign key (supplierID)
       references Supplier
```

```
Hibernate:
    select
        product0_.productID as producti1_0_0_,
        product0_.productName as productn2_0_0_,
        product0_.supplierID as supplier4_0_0_,
        product0_.unitsOnStock as unitsons3_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=?
Hibernate:

values
    next value for hibernate_sequence
```

```
Hibernate:
    /* insert org.azerebiec.Supplier
        */ insert
        into
            Supplier
            (city, companyName, street, supplierID)
        values
            (?, ?, ?, ?)
Hibernate:
    /* update
        org.azerebiec.Product */ update
            Product
        set
            productName=?,
            supplierID=?,
            unitsOnStock=?
        where
            productID=?
```

2.1.6 Schemat bazy danych



2.1.7 Tabele

Tabela PRODUCT

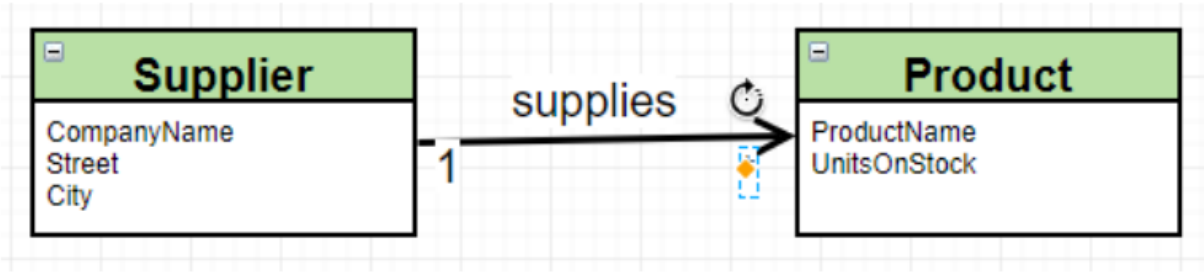
	PRODUCTID	PRODUCTNAME	UNITSONSTOCK	SUPPLIERID
1	1	Krzesło	111	2

Tabela SUPPLIER

	SUPPLIERID	CITY	COMPANYNAME	STREET
1	2	Lublin	Nowy dostawca	Chłodna

2.2 Zadanie III

Odwróć relacje zgodnie z poniższym schematem



2.2.1 Z tabelą łącznikową

2.2.1.1 Klasa Supplier

```
package org.azerebiec;

import javax.persistence.*;
import java.util.LinkedHashSet;
```

```
import java.util.Set;

@Entity
public class Supplier {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int supplierID;
    private String companyName;
    private String street;
    private String city;
    @OneToMany
    private Set<Product> products = new LinkedHashSet<>();

    public Supplier(){};

    public void addProduct(Product product){
        this.products.add(product);
    }
    public Supplier(String companyName, String city,String
street){
        this.city = city;
        this.street  = street;
        this.companyName = companyName;
    }


    @Override
    public String toString(){
        return "Supplier: " + companyName;
    }
}
```

2.2.1.2 Klasa Product

```
package org.azerebiec;

import javax.persistence.*;

@Entity
public class Product {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int productID;
    private String productName;
    private int unitsOnStock;

    public Product(String productName, int unitsOnStock) {
        this.productName = productName;
        this.unitsOnStock = unitsOnStock;
    }
    public Product() {}
```

```
    @Override
    public String toString(){
        return "Product: " + productName;
    }
}
```

2.2.1.3 Klasa Main

```
package org.azerebiec;

import org.hibernate.HibernateException;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.cfg.Configuration;

import javax.persistence.*;


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) {
        try (Session session = getSession()) {
            Transaction tx = session.beginTransaction();

            Product product1 = new Product("Krzesło", 111);
            Product product2 = new Product("Stół", 23);
            Product product3 = new Product("Łóżko", 21);
            Product product4 = new Product("Drzwi", 23);

            Supplier supplier1 = new Supplier("Dostawca 1",
"Chłodna", "Lublin");
```

```
            Supplier supplier2 = new Supplier("Dostawca 2",
"Ciepła", "Kraków");
            supplier1.addProduct(product1);
            supplier1.addProduct(product4);
            supplier1.addProduct(product3);
            supplier2.addProduct(product2);

            session.save(product1);
            session.save(product2);
            session.save(product3);
            session.save(product4);
            session.save(supplier1);
            session.save(supplier2);
            tx.commit();
        }}
}
```

2.2.1.4 Plik hibernate.cfg.xml

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
    "-//Hibernate/Hibernate Configuration DTD//EN"
    "http://www.hibernate.org/dtd/hibernate-configuration-
3.0.dtd">
<hibernate-configuration>
  <session-factory>
    <property
name="connection.url">jdbc:derby://127.0.0.1/AZerebiecJPA;crea
te=true</property>
    <property
name="connection.driver_class">org.apache.derby.jdbc.ClientDri
ver</property>
    <property
name="dialect">org.hibernate.dialect.DerbyTenSevenDialect</pro
perty>


    <property name="show_sql">true</property>
    <property name="format_sql">true</property>
    <property name="use_sql_comments">true</property>
    <property name="hibernate.hbm2ddl.auto">create-
drop</property>

    <mapping class="org.azerebiec.Product"/>
    <mapping class="org.azerebiec.Supplier"/>
  </session-factory>
</hibernate-configuration>
```

2.2.1.5 Logi SQL

```
Hibernate:

    create table Product (
       productID integer not null,
        productName varchar(255),
        unitsOnStock integer not null,
        primary key (productID)
    )
Hibernate:

    create table Supplier (
       supplierID integer not null,
        city varchar(255),
        companyName varchar(255),
        street varchar(255),
        primary key (supplierID)
    )
```

```
Hibernate:

    create table Supplier_Product (
       Supplier_supplierID integer not null,
        products_productID integer not null,
        primary key (Supplier_supplierID, products_productID)
    )
Hibernate:

    alter table Supplier_Product
       add constraint UK_sd4mo32rnl54mui98qw7bn159 unique (products_productID)
Hibernate:

    alter table Supplier_Product
       add constraint FKar5fwoh7a3vqxo0f8fh1ey8ha
       foreign key (products_productID)
       references Product
Hibernate:

    alter table Supplier_Product
       add constraint FKjskj7cplt17tebkn930wt8ke6
       foreign key (Supplier_supplierID)
       references Supplier
```

```
Hibernate:

values
    next value for hibernate_sequence
Hibernate:

values
    next value for hibernate_sequence
Hibernate:

values
    next value for hibernate_sequence
Hibernate:

values
    next value for hibernate_sequence
Hibernate:

values
    next value for hibernate_sequence
Hibernate:

values
    next value for hibernate_sequence
```

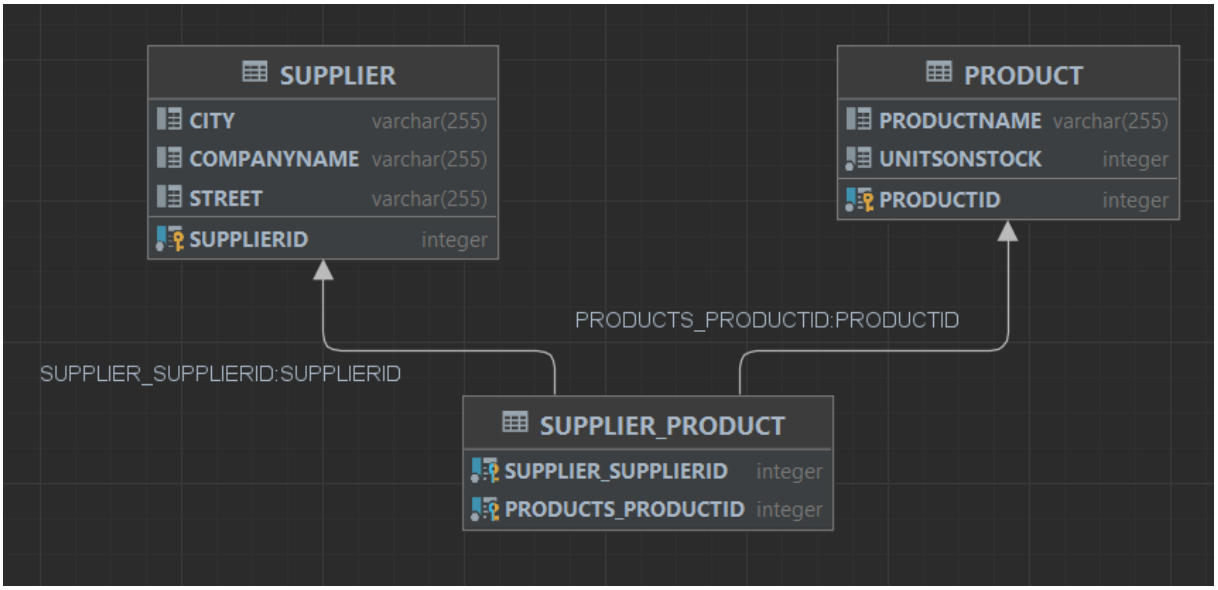
```
Hibernate:
    /* insert org.azerebiec.Product
        */ insert
        into
            Product
            (productName, unitsOnStock, productID)
        values
            (?, ?, ?)
Hibernate:
    /* insert org.azerebiec.Product
        */ insert
        into
            Product
            (productName, unitsOnStock, productID)
        values
            (?, ?, ?)
Hibernate:
    /* insert org.azerebiec.Product
        */ insert
        into
            Product
            (productName, unitsOnStock, productID)
        values
            (?, ?, ?)
```

```
Hibernate:
    /* insert org.azerebiec.Product
        */ insert
        into
            Product
            (productName, unitsOnStock, productID)
        values
            (?, ?, ?)
Hibernate:
    /* insert org.azerebiec.Supplier
        */ insert
        into
            Supplier
            (city, companyName, street, supplierID)
        values
            (?, ?, ?, ?)
Hibernate:
    /* insert org.azerebiec.Supplier
        */ insert
        into
            Supplier
            (city, companyName, street, supplierID)
        values
            (?, ?, ?, ?)
```

```
Hibernate:
    /* insert collection
        row org.azerebiec.Supplier.products */ insert
        into
            Supplier_Product
            (Supplier_supplierID, products_productID)
        values
            (?, ?)
Hibernate:
    /* insert collection
        row org.azerebiec.Supplier.products */ insert
        into
            Supplier_Product
            (Supplier_supplierID, products_productID)
        values
            (?, ?)
Hibernate:
    /* insert collection
        row org.azerebiec.Supplier.products */ insert
        into
            Supplier_Product
            (Supplier_supplierID, products_productID)
        values
            (?, ?)
```

```
Hibernate:
    /* insert collection
        row org.azerebiec.Supplier.products */ insert
        into
            Supplier_Product
            (Supplier_supplierID, products_productID)
        values
            (?, ?)

Process finished with exit code 0
```

2.2.1.6 Schemat bazy danych



2.2.1.7 Tabele

Tabela Product

	PRODUCTID	PRODUCTNAME	UNITSONSTOCK
1	1	Krzesło	111
2	2	Stół	23
3	3	Łóżko	21
4	4	Drzwi	23

Tabele Supplier

	SUPPLIERID	CITY	COMPANYNAME	STREET
1	5	Lublin	Dostawca 1	Chłodna
2	6	Kraków	Dostawca 2	Ciepła

Tabela Supplier_Product

	SUPPLIER_SUPPLIERID	PRODUCTS_PRODUCTID
1	5	1
2	5	3
3	5	4
4	6	2

2.2.2 Bez tabeli łącznikowej

2.2.2.1 Klasa Supplier

```
package org.azerebiec;

import javax.persistence.*;
import java.util.LinkedHashSet;
import java.util.Set;

@Entity
public class Supplier {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int supplierID;
    private String companyName;
    private String street;
    private String city;
    @OneToMany
    @JoinColumn(name="Supplier_FK")
    private Set<Product> products = new LinkedHashSet<>();


    public Supplier(){};

    public void addProduct(Product product){
        this.products.add(product);
    }
    public Supplier(String companyName, String city,String
street){
        this.city = city;
        this.street  = street;
        this.companyName = companyName;
    }


    @Override
    public String toString(){
        return "Supplier: " + companyName;
    }
}
```

2.2.2.2 Klasa Product

```
package org.azerebiec;

import javax.persistence.*;

@Entity
public class Product {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int productID;
```

```
    private String productName;
    private int unitsOnStock;

    public Product(String productName, int unitsOnStock) {
        this.productName = productName;
        this.unitsOnStock = unitsOnStock;
    }
    public Product() {}

    @Override
    public String toString(){
        return "Product: " + productName;
    }
}
```

2.2.2.3 Klasa Main

```
package org.azerebiec;

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) {
        try (Session session = getSession()) {
            Transaction tx = session.beginTransaction();
```

```
            Product product1 = new Product("Krzesło", 111);
            Product product2 = new Product("Stół", 23);
            Product product3 = new Product("Łóżko", 21);
            Product product4 = new Product("Drzwi", 23);

            Supplier supplier1 = new Supplier("Dostawca 1",
"Chłodna", "Lublin");
            Supplier supplier2 = new Supplier("Dostawca 2",
"Ciepła", "Kraków");
            supplier1.addProduct(product1);
            supplier1.addProduct(product4);
            supplier1.addProduct(product3);
            supplier2.addProduct(product2);

            session.save(product1);
            session.save(product2);
            session.save(product3);
            session.save(product4);
            session.save(supplier1);
            session.save(supplier2);
            tx.commit();
        }
    }
}
```

2.2.2.4 Plik hibernate.cfg.xml

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
    "-//Hibernate/Hibernate Configuration DTD//EN"
    "http://www.hibernate.org/dtd/hibernate-configuration-
3.0.dtd">
<hibernate-configuration>
  <session-factory>
    <property
name="connection.url">jdbc:derby://127.0.0.1/AZerebiecJPA;crea
te=true</property>
    <property
name="connection.driver_class">org.apache.derby.jdbc.ClientDri
ver</property>
    <property
name="dialect">org.hibernate.dialect.DerbyTenSevenDialect</pro
perty>


    <property name="show_sql">true</property>
    <property name="format_sql">true</property>
    <property name="use_sql_comments">true</property>
    <property name="hibernate.hbm2ddl.auto">create-
drop</property>
```

```
    <mapping class="org.azerebiec.Product"/>
    <mapping class="org.azerebiec.Supplier"/>
  </session-factory>
</hibernate-configuration>
```

2.2.2.5 Logi SQL

```
Hibernate:

    create table Product (
       productID integer not null,
        productName varchar(255),
        unitsOnStock integer not null,
        Supplier_FK integer,
        primary key (productID)
    )
```

```
Hibernate:

    create table Supplier (
       supplierID integer not null,
        city varchar(255),
        companyName varchar(255),
        street varchar(255),
        primary key (supplierID)
    )
Hibernate:

    alter table Product
       add constraint FKve96qacvsr1a50rgwl94enru
       foreign key (Supplier_FK)
       references Supplier
```

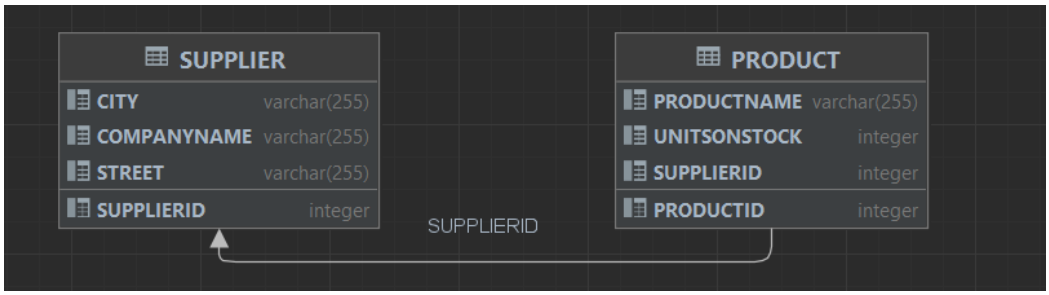
```
Hibernate:

values
    next value for hibernate_sequence
Hibernate:

values
    next value for hibernate_sequence
Hibernate:

values
    next value for hibernate_sequence
Hibernate:

values
    next value for hibernate_sequence
Hibernate:

values
    next value for hibernate_sequence
Hibernate:

values
    next value for hibernate_sequence
```

```
Hibernate:
    /* insert org.azerebiec.Product
        */ insert
        into
            Product
            (productName, unitsOnStock, productID)
        values
            (?, ?, ?)
Hibernate:
    /* insert org.azerebiec.Product
        */ insert
        into
            Product
            (productName, unitsOnStock, productID)
        values
            (?, ?, ?)
Hibernate:
    /* insert org.azerebiec.Product
        */ insert
        into
            Product
            (productName, unitsOnStock, productID)
        values
            (?, ?, ?)
```

```
Hibernate:
    /* insert org.azerebiec.Product
        */ insert
        into
            Product
            (productName, unitsOnStock, productID)
        values
            (?, ?, ?)
Hibernate:
    /* insert org.azerebiec.Supplier
        */ insert
        into
            Supplier
            (city, companyName, street, supplierID)
        values
            (?, ?, ?, ?)
Hibernate:
    /* insert org.azerebiec.Supplier
        */ insert
        into
            Supplier
            (city, companyName, street, supplierID)
        values
```

```
Hibernate:
    /* create one-to-many row org.azerebiec.Supplier.products */ update
        Product
    set
        Supplier_FK=?
    where
        productID=?
Hibernate:
    /* create one-to-many row org.azerebiec.Supplier.products */ update
        Product
    set
        Supplier_FK=?
    where
        productID=?
Hibernate:
    /* create one-to-many row org.azerebiec.Supplier.products */ update
        Product
    set
        Supplier_FK=?
    where
        productID=?
Hibernate:
    /* create one-to-many row org.azerebiec.Supplier.products */ update
        Product
    set
        Supplier_FK=?
    where
        productID=?
```

2.2.2.6 Schemat bazy danych



2.2.2.7 Tabele

Tabela Supplier

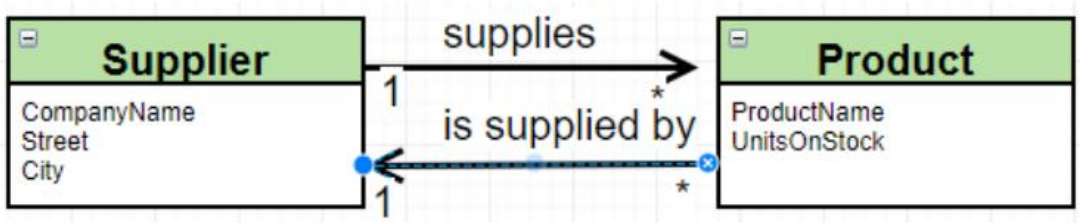
	SUPPLIERID	CITY	COMPANYNAME	STREET
1	5	Lublin	Dostawca 1	Chłodna
2	6	Kraków	Dostawca 2	Ciepła

Tabele Product

	PRODUCTID	PRODUCTNAME	UNITSONSTOCK	SUPPLIER_FK
1	1	Krzesło	111	5
2	2	Stół	23	6
3	3	Łóżko	21	5
4	4	Drzwi	23	5

2.3 Zadanie IV

Zamodeluj relację dwustronną jak poniżej:



2.3.1 Klasa Supplier

```
package org.azerebiec;

import javax.persistence.*;
import java.util.LinkedHashSet;
import java.util.Set;

@Entity
public class Supplier {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int supplierID;
    private String companyName;
    private String street;
    private String city;
    @OneToMany
    @JoinColumn(name="Supplier_FK")
    private Set<Product> products = new LinkedHashSet<>();


    public Supplier(){};

    public void addProduct(Product product){
        this.products.add(product);
    }
    public Supplier(String companyName, String city,String
street){
        this.city = city;
        this.street  = street;
        this.companyName = companyName;
    }


    @Override
    public String toString(){
        return "Supplier: " + companyName;
    }
}
```

2.3.2 Klasa Product

```
package org.azerebiec;

import javax.persistence.*;

@Entity
public class Product {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int productID;
    private String productName;
    private int unitsOnStock;
    @ManyToOne
    @JoinColumn(name="Supplier_FK")
    private Supplier supplier;


    public Product(String productName, int unitsOnStock) {
        this.productName = productName;
        this.unitsOnStock = unitsOnStock;
    }
    public Product() {}

    public void setSupplier(Supplier supplier){
        this.supplier = supplier;
    }


    @Override
    public String toString(){
        return "Product: " + productName;
    }
}
```

2.3.3 Klasa Main

```
package org.azerebiec;

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) {
        try (Session session = getSession()) {
            Transaction tx = session.beginTransaction();

            Product product1 = new Product("Krzesło", 111);
            Product product2 = new Product("Stół", 23);
            Product product3 = new Product("Łóżko", 21);
            Product product4 = new Product("Drzwi", 23);

            Supplier supplier1 = new Supplier("Dostawca 1",
"Chłodna", "Lublin");
            Supplier supplier2 = new Supplier("Dostawca 2",
"Ciepła", "Kraków");

            supplier1.addProduct(product4);
            supplier2.addProduct(product3);

            product1.setSupplier(supplier1);
            product2.setSupplier(supplier2);


            session.save(product1);
            session.save(product2);
            session.save(product3);
            session.save(product4);
            session.save(supplier1);
            session.save(supplier2);
            tx.commit();
        }
    }
}
```

2.3.4 Plik hibernate.cfg.xml

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
    "-//Hibernate/Hibernate Configuration DTD//EN"
    "http://www.hibernate.org/dtd/hibernate-configuration-
3.0.dtd">
<hibernate-configuration>
  <session-factory>
    <property
name="connection.url">jdbc:derby://127.0.0.1/AZerebiecJPA;crea
te=true</property>
    <property
name="connection.driver_class">org.apache.derby.jdbc.ClientDri
ver</property>
    <property
name="dialect">org.hibernate.dialect.DerbyTenSevenDialect</pro
perty>


    <property name="show_sql">true</property>
    <property name="format_sql">true</property>
    <property name="use_sql_comments">true</property>
    <property name="hibernate.hbm2ddl.auto">create-
drop</property>

    <mapping class="org.azerebiec.Product"/>
    <mapping class="org.azerebiec.Supplier"/>
  </session-factory>
</hibernate-configuration>
```

2.3.5 Logi SQL

```
Hibernate:

    create table Product (
       productID integer not null,
        productName varchar(255),
        unitsOnStock integer not null,
        Supplier_FK integer,
        primary key (productID)
    )
Hibernate:

    create table Supplier (
       supplierID integer not null,
        city varchar(255),
        companyName varchar(255),
        street varchar(255),
        primary key (supplierID)
    )
```

```
Hibernate:

    alter table Product
       add constraint FKve96qacvsr1a50rgwl94enru
       foreign key (Supplier_FK)
       references Supplier
```

```
Hibernate:

values
    next value for hibernate_sequence
Hibernate:

values
    next value for hibernate_sequence
Hibernate:

values
    next value for hibernate_sequence
Hibernate:

values
    next value for hibernate_sequence
Hibernate:

values
    next value for hibernate_sequence
Hibernate:

values
    next value for hibernate_sequence
```

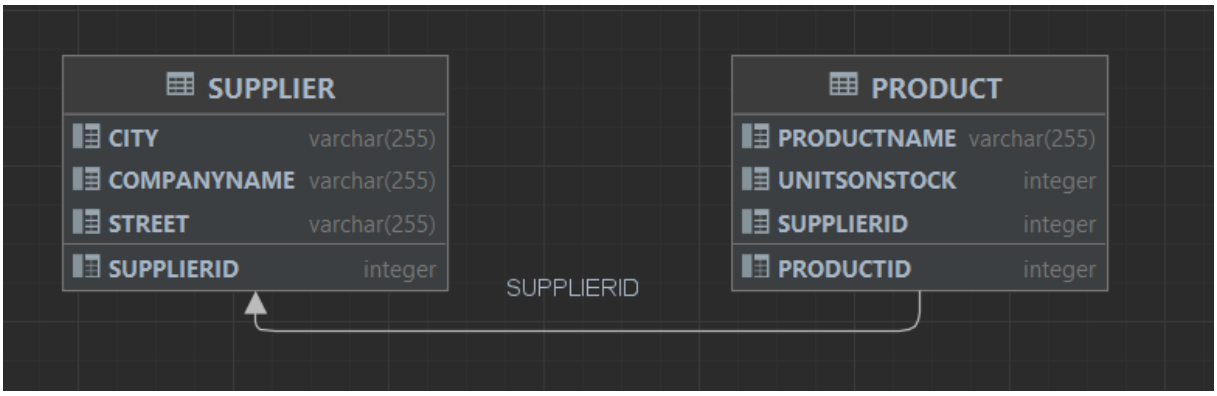
```
Hibernate:
    /* insert org.azerebiec.Product
        */ insert
        into
            Product
            (productName, Supplier_FK, unitsOnStock, productID)
        values
            (?, ?, ?, ?)
Hibernate:
    /* insert org.azerebiec.Product
        */ insert
        into
            Product
            (productName, Supplier_FK, unitsOnStock, productID)
        values
            (?, ?, ?, ?)
Hibernate:
    /* insert org.azerebiec.Product
        */ insert
        into
            Product
            (productName, Supplier_FK, unitsOnStock, productID)
        values
            (?, ?, ?, ?)
```

```
Hibernate:
    /* insert org.azerebiec.Product
        */ insert
        into
            Product
            (productName, Supplier_FK, unitsOnStock, productID)
        values
            (?, ?, ?, ?)
Hibernate:
    /* insert org.azerebiec.Supplier
        */ insert
        into
            Supplier
            (city, companyName, street, supplierID)
        values
            (?, ?, ?, ?)
Hibernate:
    /* insert org.azerebiec.Supplier
        */ insert
        into
            Supplier
            (city, companyName, street, supplierID)
        values
            (?, ?, ?, ?)
```

```
Hibernate:
    /* update
        org.azerebiec.Product */ update
            Product
        set
            productName=?,
            Supplier_FK=?,
            unitsOnStock=?
        where
            productID=?
Hibernate:
    /* update
        org.azerebiec.Product */ update
            Product
        set
            productName=?,
            Supplier_FK=?,
            unitsOnStock=?
        where
            productID=?
```

```
Hibernate:
    /* create one-to-many row org.azerebiec.Supplier.products */ update
        Product
    set
        Supplier_FK=?
    where
        productID=?
Hibernate:
    /* create one-to-many row org.azerebiec.Supplier.products */ update
        Product
    set
        Supplier_FK=?
    where
        productID=?
```

2.3.6 Schemat bazy danych



2.3.7 Tabele

Tabela Product

	PRODUCTID	PRODUCTNAME	UNITSONSTOCK	SUPPLIER_FK
1	1	Krzesło	111	5
2	2	Stół	23	6
3	3	Łóżko	21	6
4	4	Drzwi	23	5

Tabela Supplier

	SUPPLIERID	CITY	COMPANYNAME	STREET
1	5	Lublin	Dostawca 1	Chłodna
2	6	Kraków	Dostawca 2	Ciepła

2.4 Zadanie V

Dodaj klase Category z property int CategoryID, String Name oraz listą produktow List Products

2.4.1 Klasa Category

```
package org.azerebiec;

import javax.persistence.*;

import java.util.ArrayList;
import java.util.List;

@Entity
public class Category {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int categoryID;
    private String name;

    @OneToMany(mappedBy = "category")
    private List<Product> products = new ArrayList<>();

    public Category(){};
    public Category(String name){
        this.name = name;
    }
    public List<Product> getProducts(){
        return products;
    }
    @Override
    public String toString(){
        return name;
    }
}
```

2.4.2 Klasa Product

```
package org.azerebiec;

import javax.persistence.*;


@Entity
public class Product {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int productID;
    private String productName;
    private int unitsOnStock;
    @ManyToOne
    @JoinColumn(name="Supplier_FK")
    private Supplier supplier;

    @ManyToOne
    @JoinColumn(name = "Category_FK")
    private Category category;

    public Product(String productName, int unitsOnStock) {
        this.productName = productName;
        this.unitsOnStock = unitsOnStock;
    }
    public Product() {}

    public void setSupplier(Supplier supplier){
        this.supplier = supplier;
    }
    public void setCategory(Category category){
        this.category = category;
    }

    @Override
    public String toString(){
        return "Nazwa produktu: " + productName+ "(
"+unitsOnStock+ " szt.)"+"\nKategoria: " + category + "
\nDostawca: "+ supplier+"\n";
    }
}
```

2.4.3 Klasa Supplier

```
package org.azerebiec;

import javax.persistence.*;
import java.util.LinkedHashSet;
import java.util.Set;

@Entity
```

```
public class Supplier {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int supplierID;
    private String companyName;
    private String street;
    private String city;
    @OneToMany(mappedBy = "supplier")
    private Set<Product> products = new LinkedHashSet<>();


    public Supplier(){};

    public void addProduct(Product product){
        this.products.add(product);
    }
    public Supplier(String companyName, String city,String
street){
        this.city = city;
        this.street  = street;
        this.companyName = companyName;
    }
    @Override
    public String toString(){
        return companyName;
    }
}
```

2.4.4 Klasa Main

```
package org.azerebiec;

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) {
        try (Session session = getSession()) {
            Transaction tx = session.beginTransaction();

            Product product1 = new Product("Krzesło", 111);
            Product product2 = new Product("Stół", 23);
            Product product3 = new Product("'Nad Niemnem'",
21);
            Product product4 = new Product("'Pan Tadeusz'",
23);
            Category furniture = new Category("Meble");
            Category books = new Category("Książki");
            Supplier supplier1 = new Supplier("Tanie Meble",
"Lublin","Chłodna");
            Supplier supplier2 = new Supplier("Biblioteka
miejska",  "Kraków","Ciepła");

            product1.setSupplier(supplier1);
            product2.setSupplier(supplier1);
            product3.setSupplier(supplier2);
            product4.setSupplier(supplier2);
            product1.setCategory(furniture);
            product2.setCategory(furniture);
            product3.setCategory(books);
            product4.setCategory(books);


            session.save(product1);
            session.save(product2);
            session.save(product3);
            session.save(product4);
            session.save(supplier1);
            session.save(supplier2);
            session.save(furniture);
            session.save(books);
            tx.commit();

            Category category =
session.find(Category.class,7);

            for(Product prod:category.getProducts())
                System.out.println(prod);}}}
```

2.4.5 Plik hibernate.cfg.xml

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
    "-//Hibernate/Hibernate Configuration DTD//EN"
    "http://www.hibernate.org/dtd/hibernate-configuration-
3.0.dtd">
<hibernate-configuration>
  <session-factory>
    <property
name="connection.url">jdbc:derby://127.0.0.1/AZerebiecJPA;crea
te=true</property>
    <property
name="connection.driver_class">org.apache.derby.jdbc.ClientDri
ver</property>
    <property
name="dialect">org.hibernate.dialect.DerbyTenSevenDialect</pro
perty>


    <property name="show_sql">true</property>
    <property name="format_sql">true</property>
    <property name="use_sql_comments">true</property>
    <property name="hibernate.hbm2ddl.auto">update</property>

    <mapping class="org.azerebiec.Product"/>
    <mapping class="org.azerebiec.Supplier"/>
    <mapping class ="org.azerebiec.Category"/>
  </session-factory>
</hibernate-configuration>
```

2.4.6 Logi SQL

```
Hibernate:

    create table Category (
       categoryID integer not null,
        name varchar(255),
        primary key (categoryID)
    )
```

```
Hibernate:

    create table Product (
       productID integer not null,
        productName varchar(255),
        unitsOnStock integer not null,
        Category_FK integer,
        Supplier_FK integer,
        primary key (productID)
    )
Hibernate:

    create table Supplier (
       supplierID integer not null,
        city varchar(255),
        companyName varchar(255),
        street varchar(255),
        primary key (supplierID)
    )
```

```
Hibernate:

    alter table Product
       add constraint FKkrgkxd6gnqyxwwoaogk95pt3d
       foreign key (Category_FK)
       references Category
Hibernate:

    alter table Product
       add constraint FKve96qacvsr1a50rgwl94enru
       foreign key (Supplier_FK)
       references Supplier
```

```
Hibernate:

values
    next value for hibernate_sequence
Hibernate:

values
    next value for hibernate_sequence
Hibernate:

values
    next value for hibernate_sequence
Hibernate:

values
    next value for hibernate_sequence
Hibernate:

values
    next value for hibernate_sequence
Hibernate:

values
    next value for hibernate_sequence
Hibernate:
```

```
Hibernate:
    /* insert org.azerebiec.Product
        */ insert
        into
            Product
            (Category_FK, productName, Supplier_FK, unitsOnStock, productID)
        values
            (?, ?, ?, ?, ?)
Hibernate:
    /* insert org.azerebiec.Product
        */ insert
        into
            Product
            (Category_FK, productName, Supplier_FK, unitsOnStock, productID)
        values
            (?, ?, ?, ?, ?)
Hibernate:
    /* insert org.azerebiec.Product
        */ insert
        into
            Product
            (Category_FK, productName, Supplier_FK, unitsOnStock, productID)
        values
            (?, ?, ?, ?, ?)
```

```
Hibernate:
    /* insert org.azerebiec.Product
        */ insert
        into
            Product
            (Category_FK, productName, Supplier_FK, unitsOnStock, productID)
        values
            (?, ?, ?, ?, ?)
Hibernate:
    /* insert org.azerebiec.Supplier
        */ insert
        into
            Supplier
            (city, companyName, street, supplierID)
        values
            (?, ?, ?, ?)
Hibernate:
    /* insert org.azerebiec.Supplier
        */ insert
        into
            Supplier
            (city, companyName, street, supplierID)
        values
            (?, ?, ?, ?)
```

```
Hibernate:
    /* insert org.azerebiec.Category
        */ insert
        into
            Category
            (name, categoryID)
        values
            (?, ?)
Hibernate:
    /* insert org.azerebiec.Category
        */ insert
        into
            Category
            (name, categoryID)
        values
            (?, ?)
Hibernate:
    /* update
        org.azerebiec.Product */ update
            Product
        set
            Category_FK=?,
            productName=?,
            Supplier_FK=?,
            unitsOnStock=?
        where
            productID=?
```

```
Hibernate:
    /* update
        org.azerebiec.Product */ update
            Product
        set
            Category_FK=?,
            productName=?,
            Supplier_FK=?,
            unitsOnStock=?
        where
            productID=?
Hibernate:
    /* update
        org.azerebiec.Product */ update
            Product
        set
            Category_FK=?,
            productName=?,
            Supplier_FK=?,
            unitsOnStock=?
        where
            productID=?
```

```
Hibernate:
    /* update
        org.azerebiec.Product */ update
            Product
        set
            Category_FK=?,
            productName=?,
            Supplier_FK=?,
            unitsOnStock=?
        where
            productID=?
Hibernate:
    select
        category0_.categoryID as category1_0_0_,
        category0_.name as name2_0_0_
    from
        Category category0_
    where
        category0_.categoryID=?
```

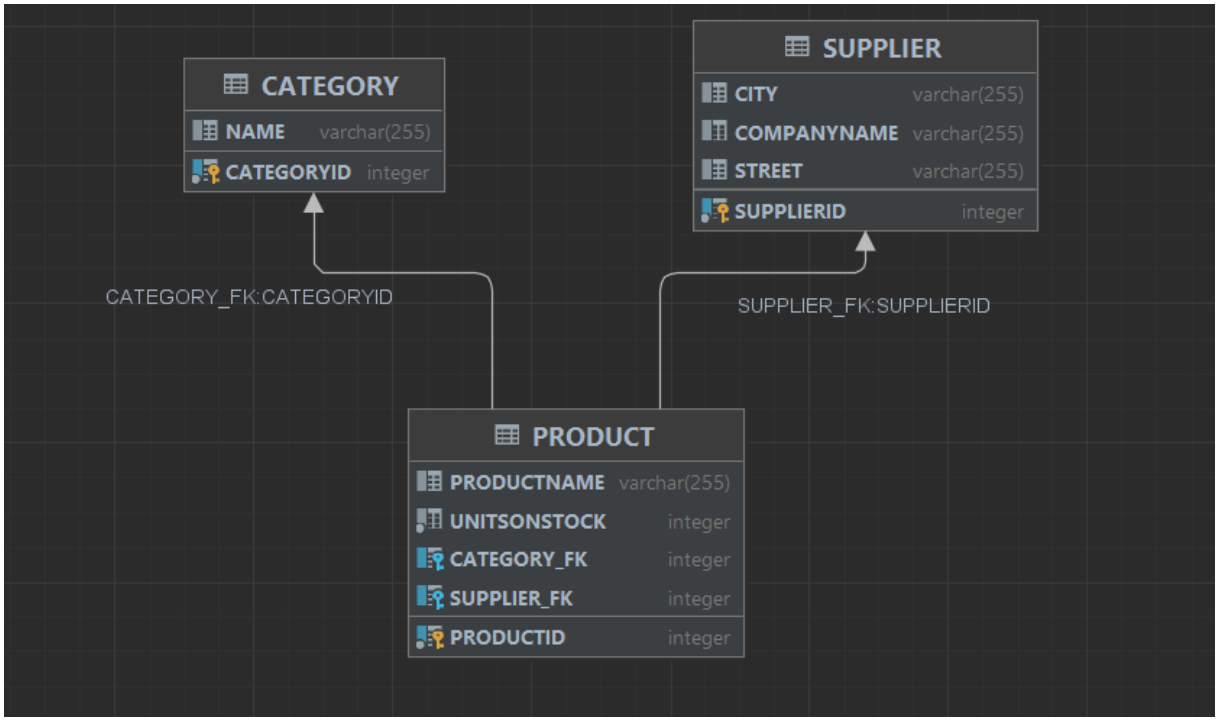
```
Hibernate:
    select
        products0_.Category_FK as category4_1_0_,
        products0_.productID as producti1_1_0_,
        products0_.productID as producti1_1_1_,
        products0_.Category_FK as category4_1_1_,
        products0_.productName as productn2_1_1_,
        products0_.Supplier_FK as supplier5_1_1_,
        products0_.unitsOnStock as unitsons3_1_1_,
        supplier1_.supplierID as supplier1_2_2_,
        supplier1_.city as city2_2_2_,
        supplier1_.companyName as companyn3_2_2_,
        supplier1_.street as street4_2_2_
    from
        Product products0_
    left outer join
        Supplier supplier1_
            on products0_.Supplier_FK=supplier1_.supplierID
    where
        products0_.Category_FK=?
```

2.4.7 Wydobycie produktów z kategorii

```
Category category = session.find(Category.class, o: 7);

for(Product prod:category.getProducts())
    System.out.println(prod);
```

```
Nazwa produktu: Krzesło( 111 szt.)
Kategoria: Meble
Dostawca: Tanie Meble

Nazwa produktu: Stół( 23 szt.)
Kategoria: Meble
Dostawca: Tanie Meble
```

2.4.8 Schemat bazy danych



2.4.9 Tabele

Tabela Products

	PRODUCTID	PRODUCTNAME	UNITSONSTOCK	CATEGORY_FK	SUPPLIER_FK
1	1	Krzesło	111	7	5
2	2	Stół	23	7	5
3	3	'Nad Niemnem'	21	8	6
4	4	'Pan Tadeusz'	23	8	6

Tabela Category

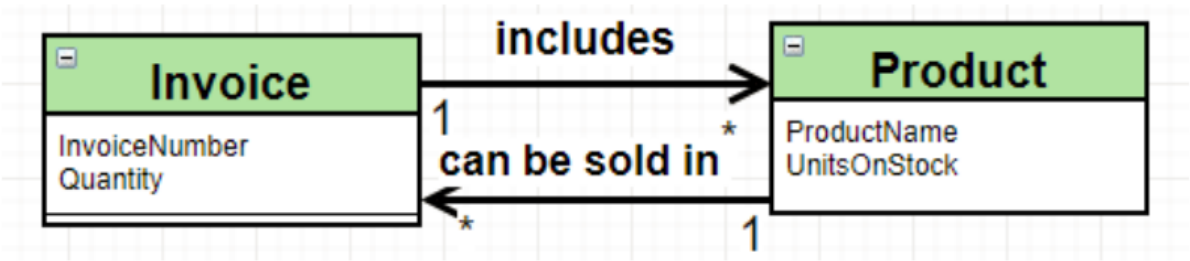
	CATEGORYID	NAME
1	7	Meble
2	8	Książki

Tabela Supplier

	SUPPLIERID	CITY	COMPANYNAME	STREET
1	5	Lublin	Tanie Meble	Chłodna
2	6	Kraków	Biblioteka miejska	Ciepła

2.5 Zadanie VI

Zamodeluj relacje wiele-do-wielu, jak poniżej:



2.5.1 Klasa Invoice

```
package org.azerebiec;

import javax.persistence.*;
import java.util.LinkedHashSet;
import java.util.Set;

@Entity
public class Invoice {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
```

```
    private int invoiceNumber;
    private int quantity;
    @ManyToMany
    private Set<Product> products = new LinkedHashSet<>();

    public Set<Product> getProducts(){
        return products;
    }
    public void addProduct(Product product,int units){
        this.products.add(product);
        this.quantity+=units;
    }

    public Invoice(){};

    @Override
    public String toString(){
        return "Numer faktury: "+ invoiceNumber+ "\nLiczba
produktów: "+quantity;
    }
}
```

2.5.2 Klasa Product

```
package org.azerebiec;

import javax.persistence.*;
import java.util.LinkedHashSet;
import java.util.Set;


@Entity
public class Product {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int productID;
    private String productName;
    private int unitsOnStock;
    @ManyToOne
    @JoinColumn(name="Supplier_FK")
    private Supplier supplier;

    @ManyToOne
    @JoinColumn(name = "Category_FK")
    private Category category;

    @ManyToMany(mappedBy = "products")
    private Set<Invoice> invoices = new LinkedHashSet<>();
```

```
    public Product(String productName, int unitsOnStock) {
        this.productName = productName;
        this.unitsOnStock = unitsOnStock;
    }
    public Product() {}

    public void setSupplier(Supplier supplier){
        this.supplier = supplier;
    }
    public void setCategory(Category category){
        this.category = category;
    }
    public void addInvoice(Invoice invoice, int units){
        this.invoices.add(invoice);
        this.unitsOnStock -= units;
    }
    public Set<Invoice> getInvoices(){
        return invoices;
    }

    @Override
    public String toString(){
        return "Nazwa produktu: " + productName+ "(
"+unitsOnStock+ " szt.)"+"\nKategoria: " + category + "
\nDostawca: "+ supplier+"\n";
    }
}
```

2.5.3 Klasa Main

```
package org.azerebiec;

import org.hibernate.HibernateException;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.cfg.Configuration;

import javax.persistence.criteria.CriteriaBuilder;


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) {
        try (Session session = getSession()) {
            Transaction tx = session.beginTransaction();

            Product product1 = new Product("Krzesło", 111);
            Product product2 = new Product("Stół", 23);
            Product product3 = new Product("'Nad Niemnem'",
21);
            Product product4 = new Product("'Pan Tadeusz'",
23);
            Category furniture = new Category("Meble");
            Category books = new Category("Książki");
            Supplier supplier1 = new Supplier("Tanie Meble",
"Lublin","Chłodna");
            Supplier supplier2 = new Supplier("Biblioteka
miejska",  "Kraków","Ciepła");

            product1.setSupplier(supplier1);
            product2.setSupplier(supplier1);
            product3.setSupplier(supplier2);
            product4.setSupplier(supplier2);
            product1.setCategory(furniture);
            product2.setCategory(furniture);
            product3.setCategory(books);
            product4.setCategory(books);

            Invoice invoice1 = new Invoice();
            Invoice invoice2 = new Invoice();
            Invoice invoice3 = new Invoice();

            invoice1.addProduct(product1,10);
            product1.addInvoice(invoice1, 10);

            invoice2.addProduct(product3,5);
            invoice2.addProduct(product4,2);
            product3.addInvoice(invoice2,5);
            product4.addInvoice(invoice2,2);

            invoice3.addProduct(product2,3);
            invoice3.addProduct(product1,4);
```

```
            product2.addInvoice(invoice3,3);
            product1.addInvoice(invoice3,4);


            session.save(product1);
            session.save(product2);
            session.save(product3);
            session.save(product4);
            session.save(invoice1);
            session.save(invoice2);
            session.save(invoice3);
            session.save(supplier1);
            session.save(supplier2);
            session.save(furniture);
            session.save(books);
            tx.commit();

            System.out.println("Faktura o numerze 6\n");
            Invoice invoice = session.find(Invoice.class,6);

invoice.getProducts().forEach(System.out::println);

            System.out.println("Faktury dla produktu 1");
            Product product = session.find(Product.class,1);

product.getInvoices().forEach(System.out::println);

        }
    }}
```

2.5.4 Pozostałe klasy

Pozostałe klasy nie zostały zmienione i wyglądają jak poprzednio

2.5.5 Plik hibernate.cfg.xml

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
    "-//Hibernate/Hibernate Configuration DTD//EN"
    "http://www.hibernate.org/dtd/hibernate-configuration-
3.0.dtd">
<hibernate-configuration>
  <session-factory>
    <property
name="connection.url">jdbc:derby://127.0.0.1/AZerebiecJPA;crea
te=true</property>
    <property
name="connection.driver_class">org.apache.derby.jdbc.ClientDri
ver</property>
    <property
name="dialect">org.hibernate.dialect.DerbyTenSevenDialect</pro
```

```
perty>


    <property name="show_sql">true</property>
    <property name="format_sql">true</property>
    <property name="use_sql_comments">true</property>
    <property name="hibernate.hbm2ddl.auto">create-
drop</property>

    <mapping class="org.azerebiec.Product"/>
    <mapping class="org.azerebiec.Supplier"/>
    <mapping class ="org.azerebiec.Category"/>
    <mapping class="org.azerebiec.Invoice"/>
  </session-factory>
</hibernate-configuration>
```

2.5.6 Logi SQL

```
Hibernate:

    create table Category (
       categoryID integer not null,
        name varchar(255),
        primary key (categoryID)
    )
```

```
Hibernate:

    create table Invoice (
       invoiceNumber integer not null,
        quantity integer not null,
        primary key (invoiceNumber)
    )
Hibernate:

    create table Invoice_Product (
       invoices_invoiceNumber integer not null,
        products_productID integer not null,
        primary key (invoices_invoiceNumber, products_productID)
    )
Hibernate:

    create table Product (
       productID integer not null,
        productName varchar(255),
        unitsOnStock integer not null,
        Category_FK integer,
        Supplier_FK integer,
        primary key (productID)
    )
```

```
Hibernate:

    create table Supplier (
       supplierID integer not null,
        city varchar(255),
        companyName varchar(255),
        street varchar(255),
        primary key (supplierID)
    )
Hibernate:

    alter table Invoice_Product
       add constraint FK2mn08nt19nrqagr12grh5uho0
       foreign key (products_productID)
       references Product
Hibernate:

    alter table Invoice_Product
       add constraint FKcbqyl9u4eh1tws13u6pk5j2nt
       foreign key (invoices_invoiceNumber)
       references Invoice
Hibernate:

    alter table Product
       add constraint FKkrgkxd6gnqyxwwoaogk95pt3d
       foreign key (Category_FK)
       references Category
```

```
       references Category
Hibernate:

    alter table Product
       add constraint FKve96qacvsr1a50rgwl94enru
       foreign key (Supplier_FK)
       references Supplier
```

```
    next value for hibernate_sequence
Hibernate:

values
    next value for hibernate_sequence
Hibernate:

values
    next value for hibernate_sequence
Hibernate:

values
    next value for hibernate_sequence
Hibernate:

values
    next value for hibernate_sequence
Hibernate:

values
    next value for hibernate_sequence
```

```
Hibernate:
    /* insert org.azerebiec.Product
        */ insert
        into
            Product
            (Category_FK, productName, Supplier_FK, unitsOnStock, productID)
        values
            (?, ?, ?, ?, ?)
Hibernate:
    /* insert org.azerebiec.Product
        */ insert
        into
            Product
            (Category_FK, productName, Supplier_FK, unitsOnStock, productID)
        values
            (?, ?, ?, ?, ?)
Hibernate:
    /* insert org.azerebiec.Product
        */ insert
        into
            Product
            (Category_FK, productName, Supplier_FK, unitsOnStock, productID)
        values
            (?, ?, ?, ?, ?)
```

```
Hibernate:
    /* insert org.azerebiec.Product
        */ insert
        into
            Product
            (Category_FK, productName, Supplier_FK, unitsOnStock, productID)
        values
            (?, ?, ?, ?, ?)
Hibernate:
    /* insert org.azerebiec.Invoice
        */ insert
        into
            Invoice
            (quantity, invoiceNumber)
        values
            (?, ?)
Hibernate:
    /* insert org.azerebiec.Invoice
        */ insert
        into
            Invoice
            (quantity, invoiceNumber)
        values
            (?, ?)
```

```
Hibernate:
    /* insert org.azerebiec.Invoice
        */ insert
        into
            Invoice
            (quantity, invoiceNumber)
        values
            (?, ?)
Hibernate:
    /* insert org.azerebiec.Supplier
        */ insert
        into
            Supplier
            (city, companyName, street, supplierID)
        values
            (?, ?, ?, ?)
Hibernate:
    /* insert org.azerebiec.Supplier
        */ insert
        into
            Supplier
            (city, companyName, street, supplierID)
        values
            (?, ?, ?, ?)
```

```
Hibernate:
    /* update
        org.azerebiec.Product */ update
            Product
        set
            Category_FK=?,
            productName=?,
            Supplier_FK=?,
            unitsOnStock=?
        where
            productID=?
Hibernate:
    /* update
        org.azerebiec.Product */ update
            Product
        set
            Category_FK=?,
            productName=?,
            Supplier_FK=?,
            unitsOnStock=?
        where
            productID=?
```

```
Hibernate:
    /* insert org.azerebiec.Category
        */ insert
        into
            Category
            (name, categoryID)
        values
            (?, ?)
Hibernate:
    /* insert org.azerebiec.Category
        */ insert
        into
            Category
            (name, categoryID)
        values
            (?, ?)
Hibernate:
    /* update
        org.azerebiec.Product */ update
            Product
        set
            Category_FK=?,
            productName=?,
            Supplier_FK=?,
            unitsOnStock=?
        where
            productID=?
```

```
Hibernate:
    /* update
        org.azerebiec.Product */ update
            Product
        set
            Category_FK=?,
            productName=?,
            Supplier_FK=?,
            unitsOnStock=?
        where
            productID=?
Hibernate:
    /* insert collection
        row org.azerebiec.Invoice.products */ insert
        into
            Invoice_Product
            (invoices_invoiceNumber, products_productID)
        values
            (?, ?)
Hibernate:
    /* insert collection
        row org.azerebiec.Invoice.products */ insert
        into
            Invoice_Product
            (invoices_invoiceNumber, products_productID)
        values
            (?, ?)
```

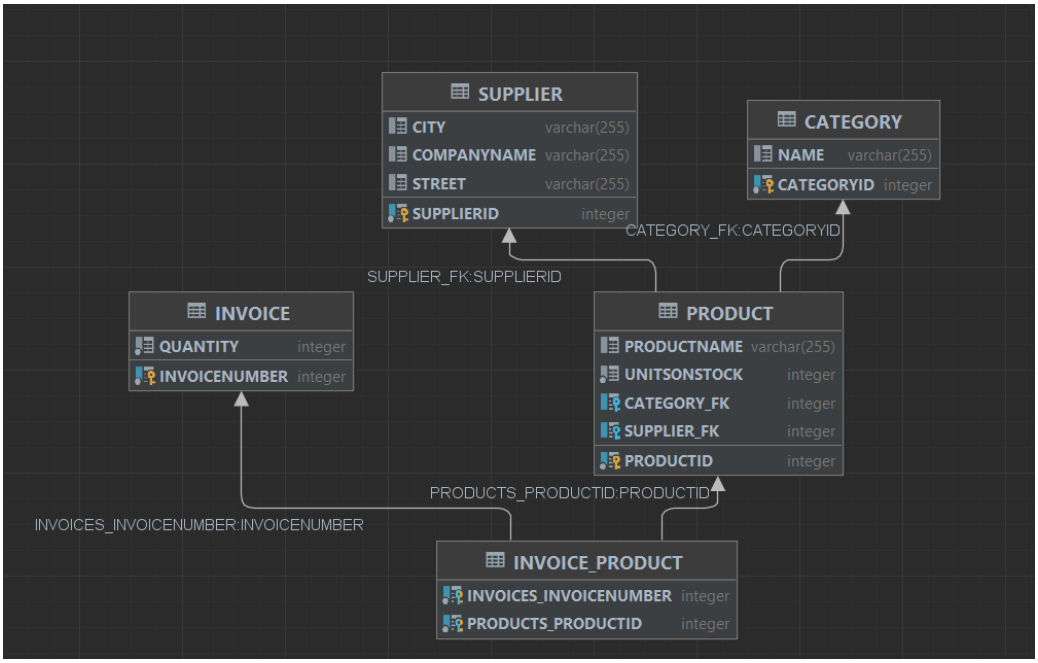
```
Hibernate:
    /* insert collection
        row org.azerebiec.Invoice.products */ insert
        into
            Invoice_Product
            (invoices_invoiceNumber, products_productID)
        values
            (?, ?)
Hibernate:
    /* insert collection
        row org.azerebiec.Invoice.products */ insert
        into
            Invoice_Product
            (invoices_invoiceNumber, products_productID)
        values
            (?, ?)
Hibernate:
    /* insert collection
        row org.azerebiec.Invoice.products */ insert
        into
            Invoice_Product
            (invoices_invoiceNumber, products_productID)
        values
            (?, ?)
```

2.5.7 Wypisanie produktu i faktury

```
System.out.println("Faktura o numerze 6\n");
Invoice invoice = session.find(Invoice.class, o: 6);
invoice.getProducts().forEach(System.out::println);

System.out.println("Faktury dla produktu 1");
Product product = session.find(Product.class, o: 1);
product.getInvoices().forEach(System.out::println);
```

```
Faktura o numerze 6

Nazwa produktu: 'Nad Niemnem'( 16 szt.)
Kategoria: Książki
Dostawca: Biblioteka miejska

Nazwa produktu: 'Pan Tadeusz'( 21 szt.)
Kategoria: Książki
Dostawca: Biblioteka miejska

Faktury dla produktu 1
Numer faktury: 5
Liczba produktów: 10
Numer faktury: 7
Liczba produktów: 7
```

2.5.8 Schemat bazy danych



2.5.9 Tabele

Tabela Invoice

	INVOICENUMBER	QUANTITY
1	5	10
2	6	7
3	7	7

Tabela Product

	PRODUCTID	PRODUCTNAME	UNITSONSTOCK	CATEGORY_FK	SUPPLIER_FK
1	1	Krzesło	97	10	8
2	2	Stół	20	10	8
3	3	'Nad Niemnem'	16	11	9
4	4	'Pan Tadeusz'	21	11	9

Tabela Invoice_Product

	INVOICES_INVOICENUMBER	PRODUCTS_PRODUCTID
1	5	1
2	6	3
3	6	4
4	7	1
5	7	2

2.6 Zadanie VII

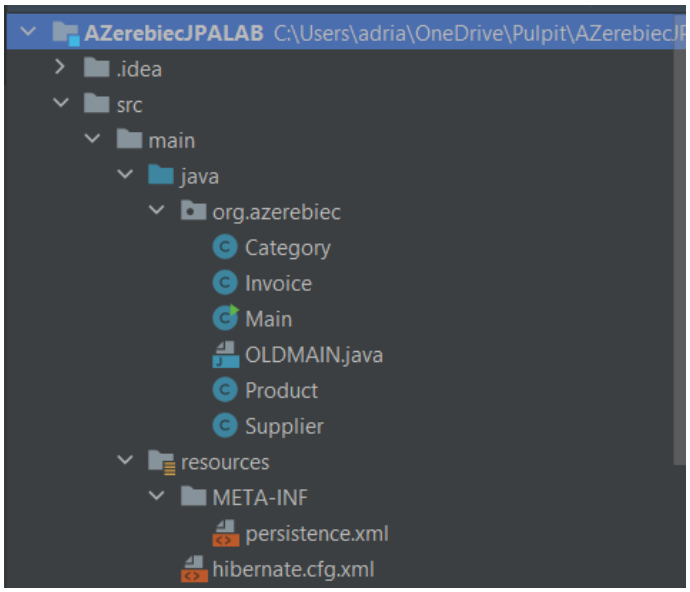
JPA- Stwórz nowego maina w którym zrobisz to samo co w poprzednim ale z wykorzystaniem JPA

2.6.1 Plik 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/AZerebiecJPA"/>
            <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>
```

Znajduję się on w:



2.6.2 Klasa Main

```
package org.azerebiec;

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();

        etx.begin();
        Product product1 = new Product("Krzesło", 111);
        Product product2 = new Product("Stół", 23);
        Product product3 = new Product("'Nad Niemnem'", 21);
        Product product4 = new Product("'Pan Tadeusz'", 23);
        em.persist(product1);
        em.persist(product2);
        em.persist(product3);
        em.persist(product4);
        etx.commit();
        etx.begin();

        Category furniture = new Category("Meble");
        Category books = new Category("Książki");
        em.persist(books);
        em.persist(furniture);
        etx.commit();
        etx.begin();

        Supplier supplier1 = new Supplier("Tanie Meble",
"Lublin","Chłodna");
        Supplier supplier2 = new Supplier("Biblioteka
miejska",  "Kraków","Ciepła");

        em.persist(supplier1);
        em.persist(supplier2);
        etx.commit();
        etx.begin();

        Invoice invoice1 = new Invoice();
        Invoice invoice2 = new Invoice();
        Invoice invoice3 = new Invoice();
        em.persist(invoice1);
        em.persist(invoice2);
        em.persist(invoice3);

        etx.commit();
        etx.begin();

        product1.setSupplier(supplier1);
        product2.setSupplier(supplier1);
        product3.setSupplier(supplier2);
        product4.setSupplier(supplier2);
        product1.setCategory(furniture);
        product2.setCategory(furniture);
        product3.setCategory(books);
        product4.setCategory(books);
```

```
        invoice1.addProduct(product1,10);
        product1.addInvoice(invoice1, 10);

        invoice2.addProduct(product3,5);
        invoice2.addProduct(product4,2);
        product3.addInvoice(invoice2,5);
        product4.addInvoice(invoice2,2);

        invoice3.addProduct(product2,3);
        invoice3.addProduct(product1,4);
        product2.addInvoice(invoice3,3);
        product1.addInvoice(invoice3,4);

        em.persist(invoice1);
        em.persist(invoice2);
        etx.commit();
        System.out.println("Faktura o numerze 6\n");
        Invoice invoice = em.find(Invoice.class,9);
        invoice.getProducts().forEach(System.out::println);

        System.out.println("Faktury dla produktu 1");
        Product product = em.find(Product.class,1);
        product.getInvoices().forEach(System.out::println);
        em.close();

    }
}
```

2.6.3 Pozostałe klasy

Pozostałe klasy pozostały bez zmian i są analogiczne do zadanie 6.

2.6.4 Logi SQL

```
Hibernate:

    create table Category (
       categoryID integer not null,
        name varchar(255),
        primary key (categoryID)
    )
Hibernate:

    create table Invoice (
       invoiceNumber integer not null,
        quantity integer not null,
        primary key (invoiceNumber)
    )
```

```
Hibernate:

    create table Invoice_Product (
       invoices_invoiceNumber integer not null,
        products_productID integer not null,
        primary key (invoices_invoiceNumber, products_productID)
    )
Hibernate:

    create table Product (
       productID integer not null,
        productName varchar(255),
        unitsOnStock integer not null,
        Category_FK integer,
        Supplier_FK integer,
        primary key (productID)
    )
Hibernate:

    create table Supplier (
       supplierID integer not null,
        city varchar(255),
        companyName varchar(255),
        street varchar(255),
        primary key (supplierID)
    )
```

```
Hibernate:

    alter table Invoice_Product
       add constraint FK2mn08nt19nrqagr12grh5uho0
       foreign key (products_productID)
       references Product
Hibernate:

    alter table Invoice_Product
       add constraint FKcbqyl9u4eh1tws13u6pk5j2nt
       foreign key (invoices_invoiceNumber)
       references Invoice
Hibernate:

    alter table Product
       add constraint FKkrgkxd6gnqyxwwoaogk95pt3d
       foreign key (Category_FK)
       references Category
Hibernate:

    alter table Product
       add constraint FKve96qacvsr1a50rgwl94enru
       foreign key (Supplier_FK)
       references Supplier
```

```
Hibernate:

values
    next value for hibernate_sequence
Hibernate:

values
    next value for hibernate_sequence
Hibernate:

values
    next value for hibernate_sequence
Hibernate:

values
    next value for hibernate_sequence
Hibernate:
    insert
    into
        Product
        (Category_FK, productName, Supplier_FK, unitsOnStock, productID)
    values
        (?, ?, ?, ?, ?)
```

```
Hibernate:
    insert
    into
        Product
        (Category_FK, productName, Supplier_FK, unitsOnStock, productID)
    values
        (?, ?, ?, ?, ?)
Hibernate:
    insert
    into
        Product
        (Category_FK, productName, Supplier_FK, unitsOnStock, productID)
    values
        (?, ?, ?, ?, ?)
Hibernate:
    insert
    into
        Product
        (Category_FK, productName, Supplier_FK, unitsOnStock, productID)
    values
        (?, ?, ?, ?, ?)
Hibernate:
```

```
values
    next value for hibernate_sequence
Hibernate:
    insert
    into
        Category
        (name, categoryID)
    values
        (?, ?)
Hibernate:
    insert
    into
        Category
        (name, categoryID)
    values
        (?, ?)
Hibernate:

values
    next value for hibernate_sequence
Hibernate:

values
    next value for hibernate_sequence
```

```
Hibernate:
    insert
    into
        Supplier
        (city, companyName, street, supplierID)
    values
        (?, ?, ?, ?)
Hibernate:
    insert
    into
        Supplier
        (city, companyName, street, supplierID)
    values
        (?, ?, ?, ?)
Hibernate:

values
    next value for hibernate_sequence
Hibernate:

values
    next value for hibernate_sequence
Hibernate:

values
    next value for hibernate_sequence
```

```
Hibernate:
    insert
    into
        Invoice
        (quantity, invoiceNumber)
    values
        (?, ?)
Hibernate:
    insert
    into
        Invoice
        (quantity, invoiceNumber)
    values
        (?, ?)
Hibernate:
    insert
    into
        Invoice
        (quantity, invoiceNumber)
    values
        (?, ?)
```

```
Hibernate:
    update
        Product
    set
        Category_FK=?,
        productName=?,
        Supplier_FK=?,
        unitsOnStock=?
    where
        productID=?
Hibernate:
    update
        Product
    set
        Category_FK=?,
        productName=?,
        Supplier_FK=?,
        unitsOnStock=?
    where
        productID=?
```

```
Hibernate:
    update
        Product
    set
        Category_FK=?,
        productName=?,
        Supplier_FK=?,
        unitsOnStock=?
    where
        productID=?
Hibernate:
    update
        Product
    set
        Category_FK=?,
        productName=?,
        Supplier_FK=?,
        unitsOnStock=?
    where
        productID=?
Hibernate:
    update
        Invoice
    set
        quantity=?
    where
        invoiceNumber=?
```

```
Hibernate:
    update
        Invoice
    set
        quantity=?
    where
        invoiceNumber=?
Hibernate:
    update
        Invoice
    set
        quantity=?
    where
        invoiceNumber=?
Hibernate:
    insert
    into
        Invoice_Product
        (invoices_invoiceNumber, products_productID)
    values
        (?, ?)
Hibernate:
    insert
    into
        Invoice_Product
        (invoices_invoiceNumber, products_productID)
    values
        (?, ?)
```

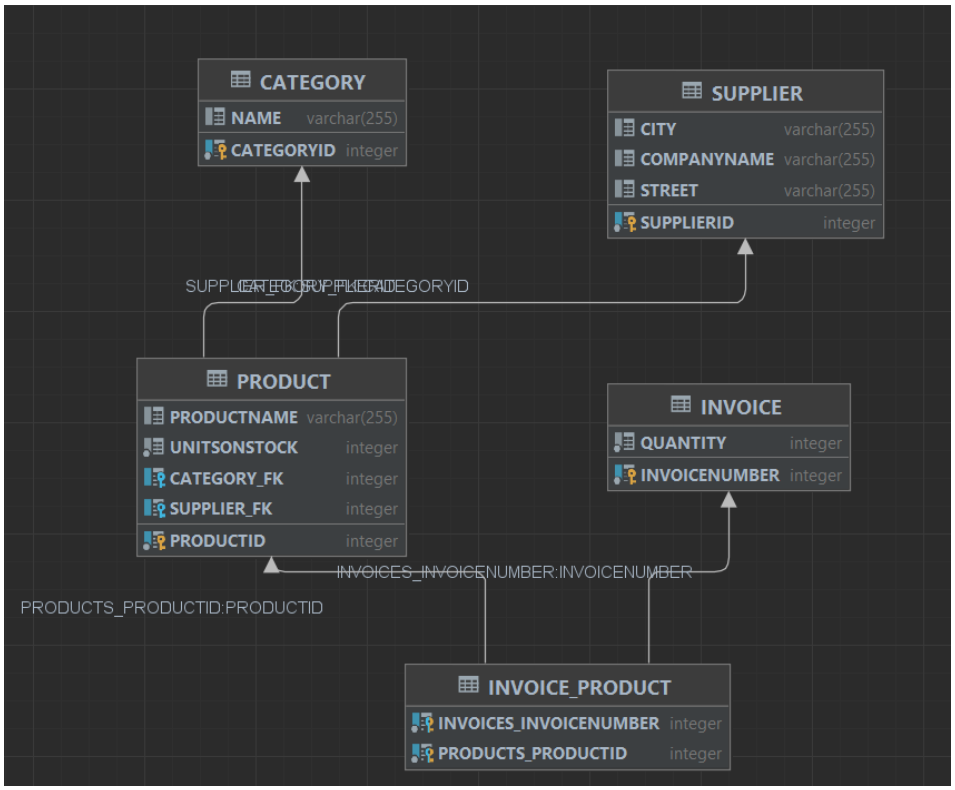
```
Hibernate:
    insert
    into
        Invoice_Product
        (invoices_invoiceNumber, products_productID)
    values
        (?, ?)
Hibernate:
    insert
    into
        Invoice_Product
        (invoices_invoiceNumber, products_productID)
    values
        (?, ?)
Hibernate:
    insert
    into
        Invoice_Product
        (invoices_invoiceNumber, products_productID)
    values
        (?, ?)
```

2.6.5 Wypisanie produktu i faktury

```
System.out.println("Faktura o numerze 6\n");
Invoice invoice = em.find(Invoice.class, o: 9);
invoice.getProducts().forEach(System.out::println);

System.out.println("Faktury dla produktu 1");
Product product = em.find(Product.class, o: 1);
product.getInvoices().forEach(System.out::println);
em.close();
```

```
Faktura o numerze 6

Nazwa produktu: Krzesło( 97 szt.)
Kategoria: Meble
Dostawca: Tanie Meble

Faktury dla produktu 1
Numer faktury: 9
Liczba produktów: 10
Numer faktury: 11
Liczba produktów: 7
```

2.6.6 Schemat bazy danych



2.6.7 Tabele

Tabela Invoice

	INVOICENUMBER	QUANTITY
1	9	10
2	10	7
3	11	7

Tabela Product

	PRODUCTID	PRODUCTNAME	UNITSONSTOCK	CATEGORY_FK	SUPPLIER_FK
1	1	Krzesło	97	6	7
2	2	Stół	20	6	7
3	3	'Nad Niemnem'	16	5	8
4	4	'Pan Tadeusz'	21	5	8

Tabela Invoice_Product

	INVOICES_INVOICENUMBER	PRODUCTS_PRODUCTID
1	9	1
2	10	3
3	10	4
4	11	1
5	11	2

2.7 Zadanie VIII

Kaskady - Zmodyfikuj model w taki sposób aby było możliwe kaskadowe tworzenie faktur wraz z nowymi produktami, oraz produktów wraz z nową fakturą

2.7.1 Klasa Product

```
package org.azerebiec;

import javax.persistence.*;
import java.io.Serializable;
import java.util.LinkedHashSet;
import java.util.Set;


@Entity
public class Product implements Serializable {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int productID;
    private String productName;
    private int unitsOnStock;
    @ManyToOne(cascade = CascadeType.PERSIST)
    @JoinColumn(name="Supplier_FK")
    private Supplier supplier;

    @ManyToOne(cascade = CascadeType.PERSIST)
    @JoinColumn(name = "Category_FK")
    private Category category;

    @ManyToMany(mappedBy = "products",cascade =
CascadeType.PERSIST)
    private Set<Invoice> invoices = new LinkedHashSet<>();


    public Product(String productName, int unitsOnStock) {
        this.productName = productName;
        this.unitsOnStock = unitsOnStock;
    }
    public Product() {}

    public void setSupplier(Supplier supplier){
        this.supplier = supplier;
    }
    public void setCategory(Category category){
        this.category = category;
    }
    public void addInvoice(Invoice invoice, int units){
        this.invoices.add(invoice);
        this.unitsOnStock -= units;
    }
```

```
    public Set<Invoice> getInvoices(){
        return invoices;
    }

    @Override
    public String toString(){
        return "Nazwa produktu: " + productName+ "(
"+unitsOnStock+ " szt.)"+"\nKategoria: " + category + "
\nDostawca: "+ supplier+"\n";
    }
}
```

2.7.2 Klasa Supplier

```
package org.azerebiec;

import javax.persistence.*;
import java.io.Serializable;
import java.util.LinkedHashSet;
import java.util.Set;

@Entity
public class Supplier implements Serializable {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int supplierID;
    private String companyName;
    private String street;
    private String city;
    @OneToMany(mappedBy = "supplier",cascade =
CascadeType.PERSIST)
    private Set<Product> products = new LinkedHashSet<>();


    public Supplier(){};

    public void addProduct(Product product){
        this.products.add(product);
    }
    public Supplier(String companyName, String city,String
street){
        this.city = city;
        this.street  = street;
        this.companyName = companyName;
    }
    @Override
    public String toString(){
        return companyName;
    }
}
```

2.7.3 Klasa Invoice

```
package org.azerebiec;

import javax.persistence.*;
import java.io.Serializable;
import java.util.LinkedHashSet;
import java.util.Set;

@Entity
public class Invoice implements Serializable {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int invoiceNumber;
    private int quantity;
    @ManyToMany(cascade = CascadeType.PERSIST)
    private Set<Product> products = new LinkedHashSet<>();

    public Set<Product> getProducts(){
        return products;
    }
    public void addProduct(Product product,int units){
        this.products.add(product);
        this.quantity+=units;
    }

    public Invoice(){};

    @Override
    public String toString(){
        return "Numer faktury: "+ invoiceNumber+ "\nLiczba
produktów: "+quantity;
    }
}
```

2.7.4 Klasa Category

```
package org.azerebiec;

import javax.persistence.*;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.List;

@Entity
public class Category implements Serializable {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int categoryID;
    private String name;
```

```
    @OneToMany(mappedBy = "category", cascade =
CascadeType.PERSIST)
    private List<Product> products = new ArrayList<>();

    public Category(){};
    public Category(String name){
        this.name = name;
    }
    public List<Product> getProducts(){
        return products;
    }
    @Override
    public String toString(){
        return name;
    }

}
```

2.7.5 Klasa Main

```
package org.azerebiec;

import javax.persistence.*;


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();

        etx.begin();
        Product product1 = new Product("Krzesło", 111);
        Product product2 = new Product("Stół", 23);
        Product product3 = new Product("'Nad Niemnem'", 21);
        Product product4 = new Product("'Pan Tadeusz'", 23);

        Category furniture = new Category("Meble");
        Category books = new Category("Książki");
```

```
        Supplier supplier1 = new Supplier("Tanie Meble",
"Lublin","Chłodna");
        Supplier supplier2 = new Supplier("Biblioteka
miejska",  "Kraków","Ciepła");


        Invoice invoice1 = new Invoice();
        Invoice invoice2 = new Invoice();
        Invoice invoice3 = new Invoice();

        product1.setSupplier(supplier1);
        product2.setSupplier(supplier1);
        product3.setSupplier(supplier2);
        product4.setSupplier(supplier2);
        product1.setCategory(furniture);
        product2.setCategory(furniture);
        product3.setCategory(books);
        product4.setCategory(books);



        invoice1.addProduct(product1,10);
        product1.addInvoice(invoice1, 10);

        invoice2.addProduct(product3,5);
        invoice2.addProduct(product4,2);
        product3.addInvoice(invoice2,5);
        product4.addInvoice(invoice2,2);

        invoice3.addProduct(product2,3);
        invoice3.addProduct(product1,4);
        product2.addInvoice(invoice3,3);
        product1.addInvoice(invoice3,4);

        em.persist(invoice1);
        em.persist(invoice2);
        etx.commit();
        System.out.println("Faktura o numerze 5\n");
        Invoice invoice = em.find(Invoice.class,5);
        invoice.getProducts().forEach(System.out::println);

        System.out.println("Faktury dla produktu 1");
        Product product = em.find(Product.class,2);
        product.getInvoices().forEach(System.out::println);
        em.close();

    }
}
```

2.7.6 Plik 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/AZerebiecJPA"/>
            <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>
```

2.7.7 Logi SQL

```
Hibernate:

    create table Category (
       categoryID integer not null,
        name varchar(255),
        primary key (categoryID)
    )
Hibernate:

    create table Invoice (
       invoiceNumber integer not null,
        quantity integer not null,
        primary key (invoiceNumber)
    )
```

```
Hibernate:

    create table Invoice_Product (
       invoices_invoiceNumber integer not null,
        products_productID integer not null,
        primary key (invoices_invoiceNumber, products_productID)
    )
Hibernate:

    create table Product (
       productID integer not null,
        productName varchar(255),
        unitsOnStock integer not null,
        Category_FK integer,
        Supplier_FK integer,
        primary key (productID)
    )
Hibernate:

    create table Supplier (
       supplierID integer not null,
        city varchar(255),
        companyName varchar(255),
        street varchar(255),
        primary key (supplierID)
    )
```

```
Hibernate:

    alter table Invoice_Product
       add constraint FK2mn08nt19nrqagr12grh5uho0
       foreign key (products_productID)
       references Product
Hibernate:

    alter table Invoice_Product
       add constraint FKcbqyl9u4eh1tws13u6pk5j2nt
       foreign key (invoices_invoiceNumber)
       references Invoice
Hibernate:

    alter table Product
       add constraint FKkrgkxd6gnqyxwwoaogk95pt3d
       foreign key (Category_FK)
       references Category
Hibernate:

    alter table Product
       add constraint FKve96qacvsr1a50rgwl94enru
       foreign key (Supplier_FK)
       references Supplier
```

```
Hibernate:

values
    next value for hibernate_sequence
Hibernate:

values
    next value for hibernate_sequence
Hibernate:

values
    next value for hibernate_sequence
Hibernate:

values
    next value for hibernate_sequence
Hibernate:

values
    next value for hibernate_sequence
Hibernate:

values
    next value for hibernate_sequence
Hibernate:

values
    next value for hibernate_sequence
```

```
Hibernate:

values
    next value for hibernate_sequence
Hibernate:

values
    next value for hibernate_sequence
Hibernate:

values
    next value for hibernate_sequence
Hibernate:
    insert
    into
        Invoice
        (quantity, invoiceNumber)
    values
        (?, ?)
Hibernate:
    insert
    into
        Category
        (name, categoryID)
    values
        (?, ?)
```

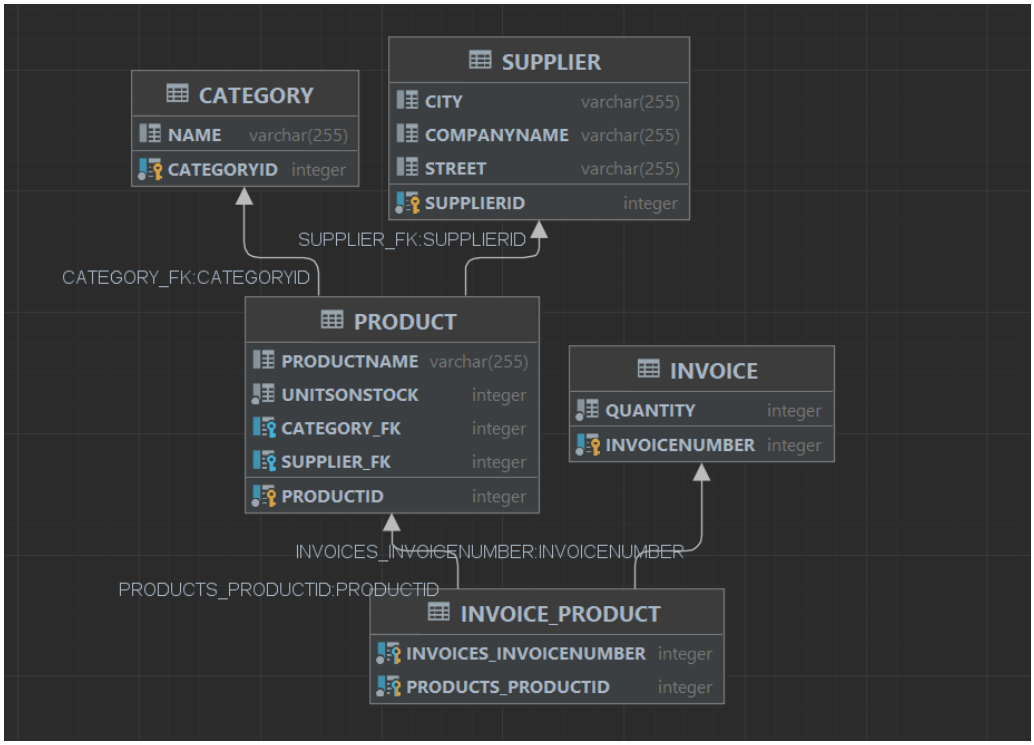
```
Hibernate:
    insert
    into
        Category
        (name, categoryID)
    values
        (?, ?)
Hibernate:
    insert
    into
        Supplier
        (city, companyName, street, supplierID)
    values
        (?, ?, ?, ?)
Hibernate:
    insert
    into
        Product
        (Category_FK, productName, Supplier_FK, unitsOnStock, productID)
    values
        (?, ?, ?, ?, ?)
Hibernate:
    insert
    into
        Invoice
        (quantity, invoiceNumber)
    values
        (?, ?)
```

```
Hibernate:
    insert
    into
        Product
        (Category_FK, productName, Supplier_FK, unitsOnStock, productID)
    values
        (?, ?, ?, ?, ?)
Hibernate:
    insert
    into
        Invoice
        (quantity, invoiceNumber)
    values
        (?, ?)
Hibernate:
    insert
    into
        Category
        (name, categoryID)
    values
        (?, ?)
Hibernate:
    insert
    into
        Supplier
        (city, companyName, street, supplierID)
    values
        (?, ?, ?, ?)
```

```
Hibernate:
    insert
    into
        Product
        (Category_FK, productName, Supplier_FK, unitsOnStock, productID)
    values
        (?, ?, ?, ?, ?)
Hibernate:
    insert
    into
        Product
        (Category_FK, productName, Supplier_FK, unitsOnStock, productID)
    values
        (?, ?, ?, ?, ?)
Hibernate:
    insert
    into
        Invoice_Product
        (invoices_invoiceNumber, products_productID)
    values
        (?, ?)
Hibernate:
    insert
    into
        Invoice_Product
        (invoices_invoiceNumber, products_productID)
    values
        (?, ?)
```

```
Hibernate:
    insert
    into
        Invoice_Product
        (invoices_invoiceNumber, products_productID)
    values
        (?, ?)
Hibernate:
    insert
    into
        Invoice_Product
        (invoices_invoiceNumber, products_productID)
    values
        (?, ?)
Hibernate:
    insert
    into
        Invoice_Product
        (invoices_invoiceNumber, products_productID)
    values
        (?, ?)
```

2.7.8 Schemat bazy danych



2.7.9 Tabele i wynik działania

Tabela Product

	PRODUCTID	PRODUCTNAME	UNITSONSTOCK	CATEGORY_FK	SUPPLIER_FK
1	2	Krzesło	97	3	4
2	6	Stół	20	3	4
3	8	'Nad Niemnem'	16	9	10
4	11	'Pan Tadeusz'	21	9	10

Tabela Invoice

	INVOICENUMBER	QUANTITY
1	1	10
2	5	7
3	7	7

Tabela Invoice_Product

	INVOICES_INVOICENUMBER	PRODUCTS_PRODUCTID
1	1	2
2	5	2
3	5	6
4	7	8
5	7	11

Wynik działania i kod

```
Nazwa produktu: Stół( 20 szt.)
Kategoria: Meble
Dostawca: Tanie Meble

Nazwa produktu: Krzesło( 97 szt.)
Kategoria: Meble
Dostawca: Tanie Meble

Faktury dla produktu 1
Numer faktury: 1
Liczba produktów: 10
Numer faktury: 5
Liczba produktów: 7
```

```
System.out.println("Faktura o numerze 5\n");
Invoice invoice = em.find(Invoice.class, o: 5);
invoice.getProducts().forEach(System.out::println);

System.out.println("Faktury dla produktu 1");
Product product = em.find(Product.class, o: 2);
product.getInvoices().forEach(System.out::println);
em.close();
```

2.8 Zadanie IX

2.8.1 Wbudowana w tabelę dostawców

2.8.1.1 Klasa Address

```
package org.azerebiec;

import javax.persistence.Embeddable;

@Embeddable
public class Address {
    private String street;
    private  String city;

    public Address(String street, String city){
        this.city = city;
        this.street = street;
    }
    public Address(){}

    @Override
    public String toString(){
        return "Adres: "+ street + " " + city;
    }


}
```

2.8.1.2 Klasa Supplier

```
package org.azerebiec;

import javax.persistence.*;
import java.io.Serializable;
import java.util.LinkedHashSet;
import java.util.Set;

@Entity
public class Supplier implements Serializable {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int supplierID;
    private String companyName;
    @Embedded
    private Address address;
    @OneToMany(mappedBy = "supplier",cascade =
CascadeType.PERSIST)
    private Set<Product> products = new LinkedHashSet<>();


    public Supplier(){};

    public void addProduct(Product product){
        this.products.add(product);
    }
    public Supplier(String companyName, Address address){
        this.address = address;
        this.companyName = companyName;
    }
    @Override
    public String toString(){
        return companyName;
    }
}
```

2.8.1.3 Klasa Main

```
package org.azerebiec;

import javax.persistence.*;


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();

        etx.begin();

        Address address1 = new Address("Chłodna","Lublin");
        Supplier supplier1 = new Supplier("Tanie Meble",
address1);
        Address address2 = new Address("Ciepła", "Kraków");
        Supplier supplier2 = new Supplier("Biblioteka
miejska",  address2);



        em.persist(supplier1);
        em.persist(supplier2);
        etx.commit();
        em.close();

    }
}
```

2.8.1.4 Logi SQL

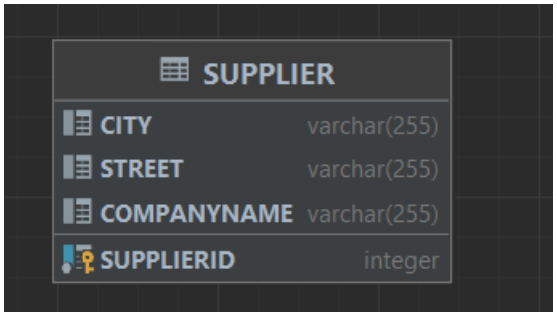
Zamieszczam tylko te dotyczące zadania

```
create table Supplier (
   supplierID integer not null,
    city varchar(255),
    street varchar(255),
    companyName varchar(255),
    primary key (supplierID)
)
```

```
Hibernate:

values
    next value for hibernate_sequence
Hibernate:

values
    next value for hibernate_sequence
Hibernate:
    insert
    into
        Supplier
        (city, street, companyName, supplierID)
    values
        (?, ?, ?, ?)
Hibernate:
    insert
    into
        Supplier
        (city, street, companyName, supplierID)
    values
        (?, ?, ?, ?)
```

2.8.1.5 Schemat bazy

Oczywiście powstają także tabele z poprzednich zadań takie jak Product ale pomijam je, gdyż nie są one istotą zadania.



2.8.1.6 Tabele

Tabela Supplier

	SUPPLIERID	CITY	STREET	COMPANYNAME
1	1	Lublin	Chłodna	Tanie Meble
2	2	Kraków	Ciepła	Biblioteka miejska

2.8.2 W klasie dostawców

Zmodyfikuj model w taki sposób, że dane adresowe znajdują się w klasie dostawców. Zmapuj to do dwóch osobnych tabel.

2.8.2.1 Klasa Supplier

```
package org.azerebiec;

import javax.persistence.*;
import java.io.Serializable;
import java.util.LinkedHashSet;
import java.util.Set;

@Entity
@SecondaryTable(name="ADDRESS")
public class Supplier implements Serializable {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int supplierID;
    private String companyName;

    @Column(table="ADDRESS")
    private String city;

    @Column(table = "ADDRESS")
    private String street;

    @OneToMany(mappedBy = "supplier",cascade =
CascadeType.PERSIST)
    private Set<Product> products = new LinkedHashSet<>();


    public Supplier(){};

    public void addProduct(Product product){
        this.products.add(product);
    }
    public Supplier(String companyName, String street, String
city){
        this.street = street;
        this.city = city;
        this.companyName = companyName;
    }
    @Override
    public String toString(){
        return companyName;
    }
}
```

2.8.2.2 Klasa Main

```
package org.azerebiec;

import javax.persistence.*;


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();

        etx.begin();

        Supplier supplier1 = new Supplier("Tanie
Meble","Chłodna","Lublin");
        Supplier supplier2 = new Supplier("Biblioteka
miejska",  "Ciepła", "Kraków");



        em.persist(supplier1);
        em.persist(supplier2);
        etx.commit();
        em.close();

    }
}
```

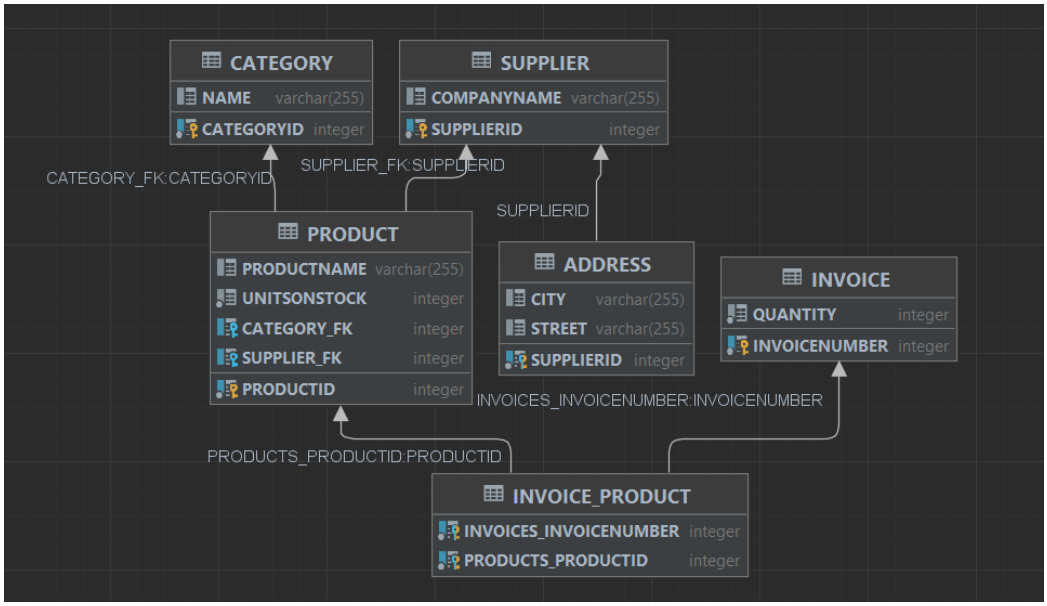
2.8.2.3 Logi SQL

```
Hibernate:

    create table Supplier (
       supplierID integer not null,
        companyName varchar(255),
        primary key (supplierID)
    )
Hibernate:

    alter table ADDRESS
       add constraint FKrnhq3kd77uysu3ckudhfpcxlx
       foreign key (supplierID)
       references Supplier
```

```
Hibernate:

values
    next value for hibernate_sequence
Hibernate:

values
    next value for hibernate_sequence
Hibernate:
    insert
    into
        Supplier
        (companyName, supplierID)
    values
        (?, ?)
Hibernate:
    insert
    into
        ADDRESS
        (city, street, supplierID)
    values
        (?, ?, ?)
```

```
Hibernate:
    insert
    into
        Supplier
        (companyName, supplierID)
    values
        (?, ?)
Hibernate:
    insert
    into
        ADDRESS
        (city, street, supplierID)
    values
        (?, ?, ?)
```

2.9.2.4 Schemat bazy danych



2.8.2.5 Tabele

Tabela Supplier

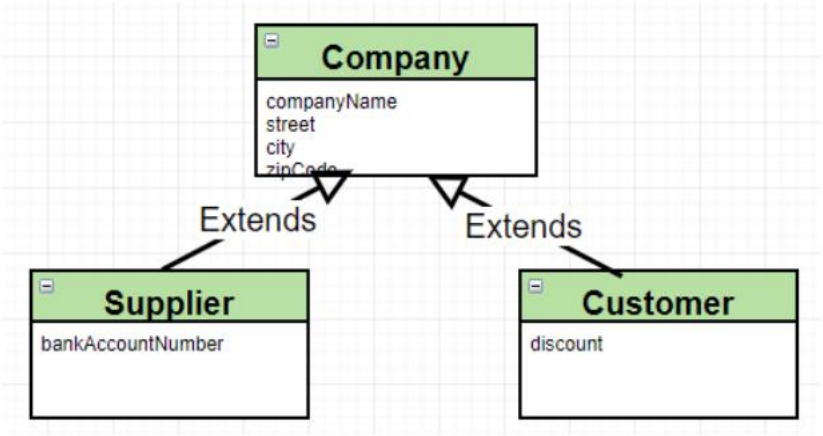
	SUPPLIERID	COMPANYNAME
1	1	Tanie Meble
2	2	Biblioteka miejska

Tabela Address

	CITY	STREET	SUPPLIERID
1	Lublin	Chłodna	1
2	Kraków	Ciepła	2

2.9 Zadanie X

Wprowadź do modelu następującą hierarchie:



Z racji, iż w zadaniu nie są potrzebne inne tabele nie będą one tworzone.

2.9.1 SINGLE TABLE

2.9.1.1 Klasa Company

```
package org.azerebiec;

import javax.persistence.*;

@Entity
@Inheritance(strategy = InheritanceType.SINGLE_TABLE)
public abstract class Company {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    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.street = street;
        this.city = city;
        this.zipCode = zipCode;
    }
}
```

2.9.1.2 Klasa Customer

```
package org.azerebiec;


import javax.persistence.Entity;
import java.io.Serializable;

@Entity
public class Customer extends Company implements Serializable
{
    private int discount;

    public Customer(String companyName,String street, String
city, String zipCode, int discount){
        super(companyName,street,city,zipCode);
        this.discount = discount;
    }
    public Customer(){
        super();
    }
}
```

2.9.1.3 Klasa Supplier

```
package org.azerebiec;


import javax.persistence.Entity;
import java.io.Serializable;

@Entity
public class Supplier extends Company implements Serializable
{
    private String bankAccountNumber;

    public Supplier(){
        super();
    }

    public Supplier(String companyName,String street, String
city, String zipCode,String bankAccountNumber){
        super(companyName,street,city,zipCode);
        this.bankAccountNumber = bankAccountNumber;
    }

}
```

2.9.1.4 Klasa Main

```
package org.azerebiec;

import javax.persistence.*;


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();

        etx.begin();
        Supplier supplier1 = new Supplier("Nowy
dostawca","Ogrodnicza","Kraków","20-
```

```
221","8923167498741298134 2");
        Customer customer1 = new Customer("Fabryka
szczęścia","Lubelska","Rzeszów","25-231",10);
        Customer customer2 = new Customer("Tanie
Meble","Poniatowskiego","Białystok","13-291",5);
        Supplier supplier2 = new Supplier("Twoja
paczka","Śląska","Wrocław","60-911","4617298641239");


        em.persist(supplier1);
        em.persist(supplier2);
        em.persist(customer1);
        em.persist(customer2);
        etx.commit();
        em.close();

    }
}
```

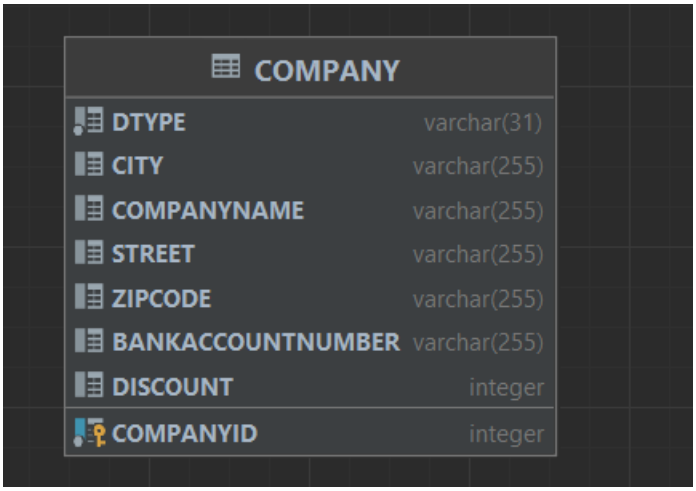
2.9.1.5 Logi SQL

```
Hibernate:

    create table Company (
       DTYPE varchar(31) not null,
        companyID integer not null,
        city varchar(255),
        companyName varchar(255),
        street varchar(255),
        zipCode varchar(255),
        bankAccountNumber varchar(255),
        discount integer,
        primary key (companyID)
    )
```

```
Hibernate:

values
    next value for hibernate_sequence
Hibernate:

values
    next value for hibernate_sequence
Hibernate:

values
    next value for hibernate_sequence
Hibernate:

values
    next value for hibernate_sequence
Hibernate:
    insert
    into
        Company
        (city, companyName, street, zipCode, bankAccountNumber, DTYPE, companyID)
    values
        (?, ?, ?, ?, ?, 'Supplier', ?)
```

```
Hibernate:
    insert
    into
        Company
        (city, companyName, street, zipCode, bankAccountNumber, DTYPE, companyID)
    values
        (?, ?, ?, ?, ?, 'Supplier', ?)
Hibernate:
    insert
    into
        Company
        (city, companyName, street, zipCode, discount, DTYPE, companyID)
    values
        (?, ?, ?, ?, ?, 'Customer', ?)
Hibernate:
    insert
    into
        Company
        (city, companyName, street, zipCode, discount, DTYPE, companyID)
    values
        (?, ?, ?, ?, ?, 'Customer', ?)
```

2.9.1.6 Schemat bazy danych



2.9.1.7 Tabele

Tabela Company

	DTYPE	COMPANYID	CITY	COMPANYNAME	STREET	ZIPCODE	BANKACCOUNTNUMBER	DISCOUNT
1	Supplier	1	Kraków	Nowy dostawca	Ogrodnicza	20-221	89231674987412981342	<null>
2	Supplier	2	Wrocław	Twoja paczka	Śląska	60-911	4617298641239	<null>
3	Customer	3	Rzeszów	Fabryka szczęścia	Lubelska	25-231	<null>	10
4	Customer	4	Białystok	Tanie Meble	Poniatowskiego	13-291	<null>	5

2.9.2 TYPE PER CLASS

2.9.2.1 Klasa Company

```
package org.azerebiec;

import javax.persistence.*;

@Entity
@Inheritance(strategy = InheritanceType.TABLE_PER_CLASS)
public abstract class Company {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    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.street = street;
        this.city = city;
        this.zipCode = zipCode;
    }}
```

2.9.2.2 Pozostałe klasy

Pozostałe klasy nie zostały zmienione

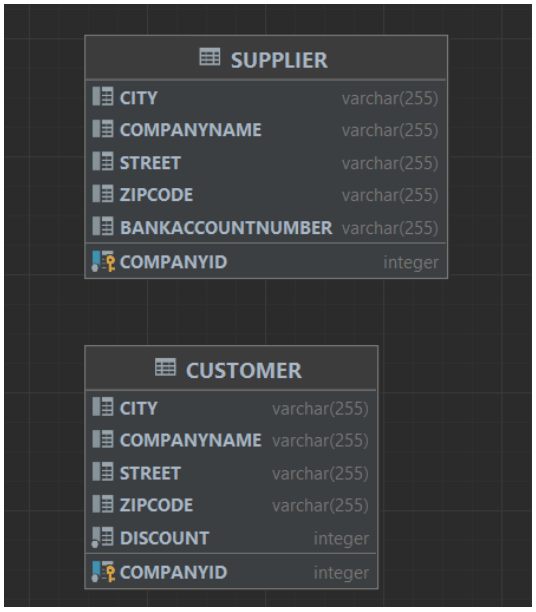
2.9.2.3 Logi SQL

```
Hibernate:

    create table Customer (
       companyID integer not null,
        city varchar(255),
        companyName varchar(255),
        street varchar(255),
        zipCode varchar(255),
        discount integer not null,
        primary key (companyID)
    )
Hibernate:

    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)
    )
```

```
Hibernate:

values
    next value for hibernate_sequence
Hibernate:

values
    next value for hibernate_sequence
Hibernate:

values
    next value for hibernate_sequence
Hibernate:

values
    next value for hibernate_sequence
Hibernate:
    insert
    into
        Supplier
        (city, companyName, street, zipCode, bankAccountNumber, companyID)
    values
        (?, ?, ?, ?, ?, ?)
```

```
Hibernate:
    insert
    into
        Supplier
        (city, companyName, street, zipCode, bankAccountNumber, companyID)
    values
        (?, ?, ?, ?, ?, ?)
Hibernate:
    insert
    into
        Customer
        (city, companyName, street, zipCode, discount, companyID)
    values
        (?, ?, ?, ?, ?, ?)
Hibernate:
    insert
    into
        Customer
        (city, companyName, street, zipCode, discount, companyID)
    values
        (?, ?, ?, ?, ?, ?)
```

2.9.2.4 Schemat bazy danych



2.9.2.5 Tabele

Tabela Customer

	COMPANYID	CITY	COMPANYNAME	STREET	ZIPCODE	DISCOUNT
1	3	Rzeszów	Fabryka szczęścia	Lubelska	25-231	10
2	4	Białystok	Tanie Meble	Poniatowskiego	13-291	5

Tabela Supplier

	COMPANYID	CITY	COMPANYNAME	STREET	ZIPCODE	BANKACCOUNTNUMBER
1	1	Kraków	Nowy dostawca	Ogrodnicza	20-221	89231674987412981342
2	2	Wrocław	Twoja paczka	Śląska	60-911	4617298641239

2.9.3 JOINED

2.9.3.1 Klasa Company

```
package org.azerebiec;

import javax.persistence.*;

@Entity
@Inheritance(strategy = InheritanceType.JOINED)
public abstract class Company {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    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.street = street;
        this.city = city;
        this.zipCode = zipCode;
    }

}
```

2.9.3.2 Pozostałe klasy

Pozostałe klasy zostały niezmienione

2.9.3.3 Logi SQL

```
Hibernate:

    create table Company (
       companyID integer not null,
        city varchar(255),
        companyName varchar(255),
        street varchar(255),
        zipCode varchar(255),
        primary key (companyID)
    )
Hibernate:

    create table Customer (
       discount integer not null,
        companyID integer not null,
        primary key (companyID)
    )
```

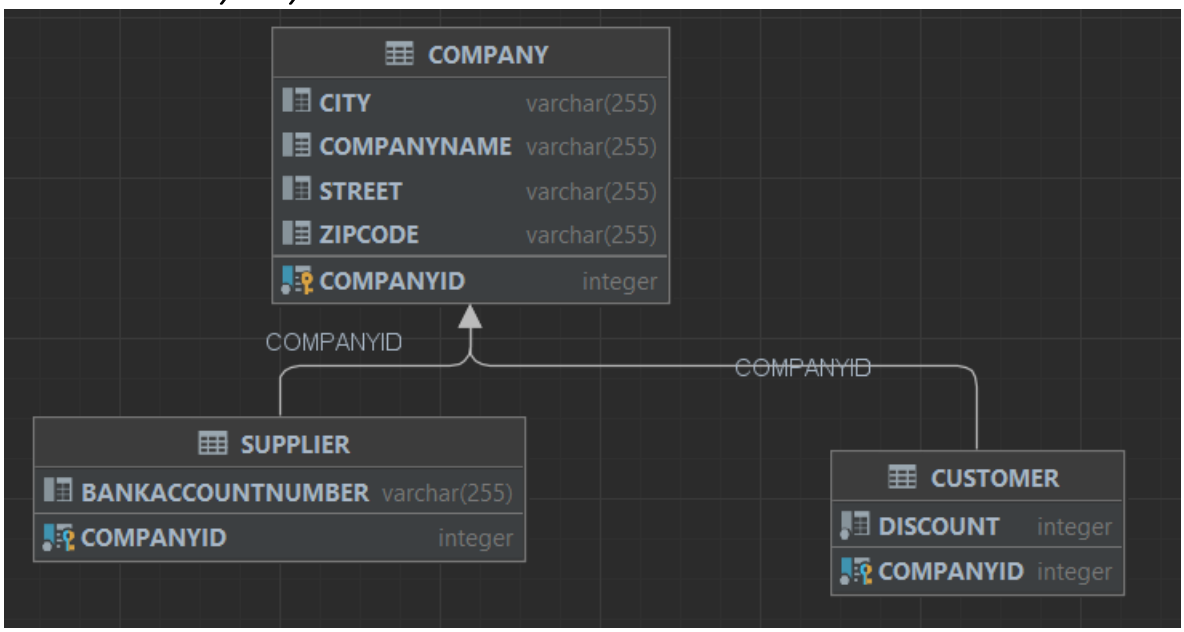
```
Hibernate:

    create table Supplier (
       bankAccountNumber varchar(255),
        companyID integer not null,
        primary key (companyID)
    )
Hibernate:

    alter table Customer
       add constraint FKn7fvr687iixps0s6i5casr6f3
       foreign key (companyID)
       references Company
Hibernate:

    alter table Supplier
       add constraint FKpinunrb4v5p4aemt2k4fnkjp8
       foreign key (companyID)
       references Company
```

```
Hibernate:

values
    next value for hibernate_sequence
Hibernate:

values
    next value for hibernate_sequence
Hibernate:

values
    next value for hibernate_sequence
Hibernate:

values
    next value for hibernate_sequence
Hibernate:
    insert
    into
        Company
        (city, companyName, street, zipCode, companyID)
    values
        (?, ?, ?, ?, ?)
```

```
Hibernate:
    insert
    into
        Supplier
        (bankAccountNumber, companyID)
    values
        (?, ?)
Hibernate:
    insert
    into
        Company
        (city, companyName, street, zipCode, companyID)
    values
        (?, ?, ?, ?, ?)
Hibernate:
    insert
    into
        Supplier
        (bankAccountNumber, companyID)
    values
        (?, ?)
```

```
Hibernate:
    insert
    into
        Company
        (city, companyName, street, zipCode, companyID)
    values
        (?, ?, ?, ?, ?)
Hibernate:
    insert
    into
        Customer
        (discount, companyID)
    values
        (?, ?)
Hibernate:
    insert
    into
        Company
        (city, companyName, street, zipCode, companyID)
    values
        (?, ?, ?, ?, ?)
Hibernate:
    insert
    into
        Customer
        (discount, companyID)
    values
        (?, ?)
```

2.9.3.4 Schemat bazy danych



2.9.3.5 Tabele

Tabela Company

	COMPANYID	CITY	COMPANYNAME	STREET	ZIPCODE
1	1	Kraków	Nowy dostawca	Ogrodnicza	20-221
2	2	Wrocław	Twoja paczka	Śląska	60-911
3	3	Rzeszów	Fabryka szczęścia	Lubelska	25-231
4	4	Białystok	Tanie Meble	Poniatowskiego	13-291

Tabela Supplier

	BANKACCOUNTNUMBER	COMPANYID
1	89231674987412981342	1
2	4617298641239	2

Tabela Customer

	DISCOUNT	COMPANYID
1	10	3
2	5	4

2.10 Końcowe pliki w projekcie