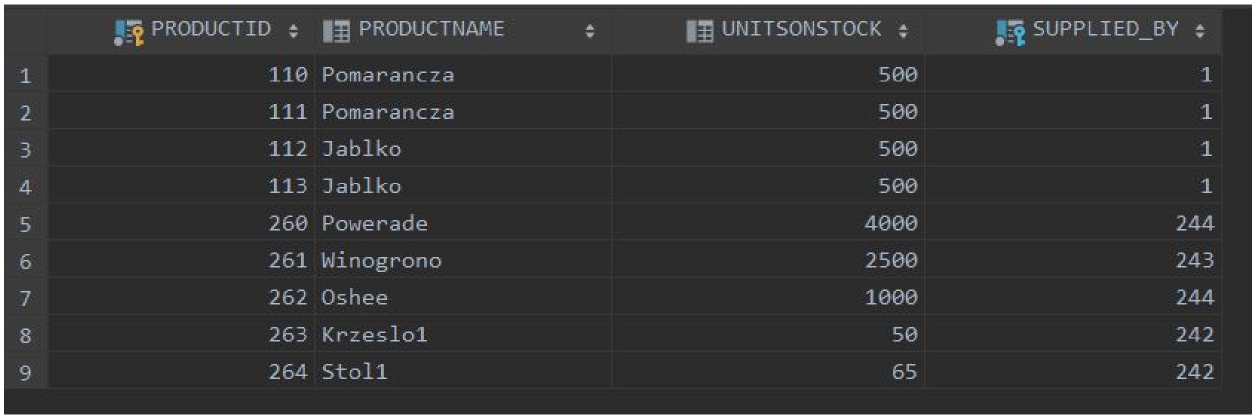
Mateusz Nabywaniec data laboratorium 13.11.19 r.
Zadanie Hibernate

III Dodanie relacji Supplier-Product (product is supplied by supplier)

```java
import javax.persistence.*;

@Entity
@Table(name = "PRODUCTS" )
public class Product {

   @Id
   @GeneratedValue(strategy = GenerationType.AUTO)
   private int productID;

   private String ProductName;
   private Integer UnitsOnStock;

   @ManyToOne
   @JoinColumn(name = "SUPPLIED_BY")
   private Supplier suppliedBy;

   public Product(String productName) {
       ProductName = productName;
   }

   public Product(String productName, Integer unitsOnStock, Supplier supplier) {
       ProductName = productName;
       UnitsOnStock = unitsOnStock;
       this.suppliedBy = supplier;
   }

   public Product(String productName, Integer unitsOnStock) {
       ProductName = productName;
       UnitsOnStock = unitsOnStock;
   }

   public Product() {
   }

   public void setSuppliedBy(Supplier supplier) {
       this.suppliedBy = supplier;
   }
}
```

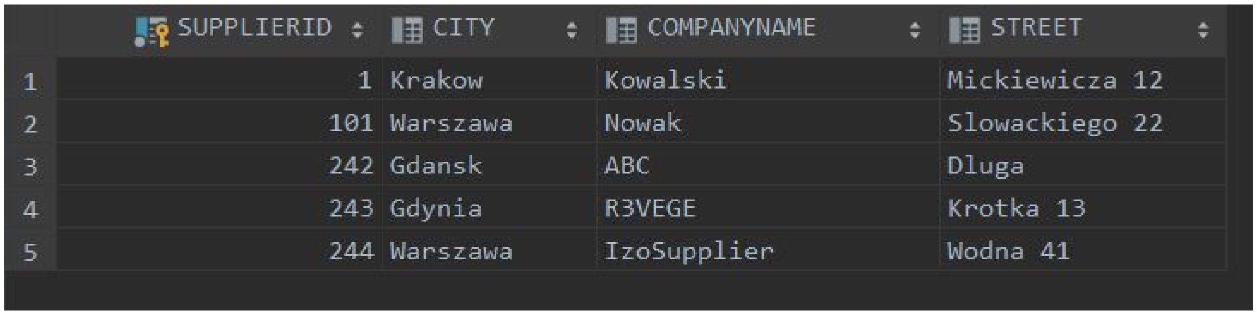


	PRODUCTID	PRODUCTNAME	UNITSONSTOCK	SUPPLIED_BY
1	110	Pomarancza	500	1
2	111	Pomarancza	500	1
3	112	Jablko	500	1
4	113	Jablko	500	1
5	260	Powerade	4000	244
6	261	Winogrono	2500	243
7	262	Oshee	1000	244
8	263	Krzeslo1	50	242
9	264	Stol1	65	242

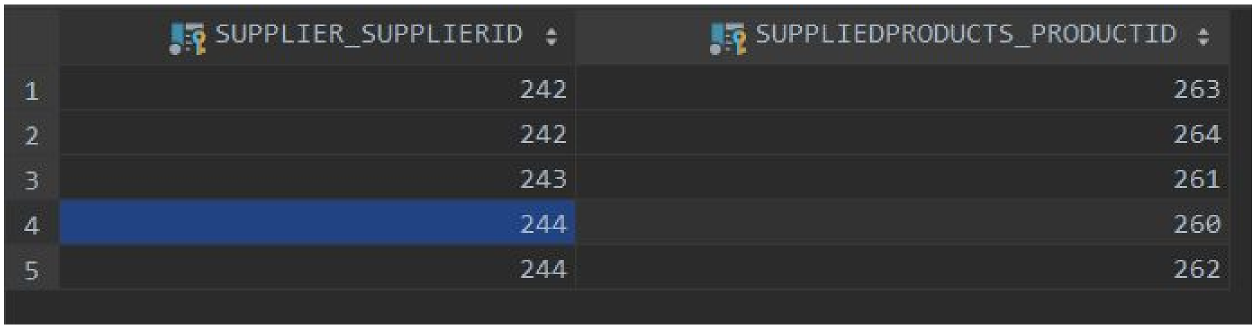
IV Dodanie relacji supplier - products (supplier supplies products).

```
import org.hibernate.dialect.ProgressDialect;

import javax.persistence.*;
import java.util.HashSet;
import java.util.List;
import java.util.Set;

@Entity
@Table(name = "SUPPLIERS" )
public class Supplier {
   @Id
   @GeneratedValue(strategy = GenerationType.AUTO)
   private int supplierID;

   private String CompanyName;
   private String Street;
   private String City;

   @OneToMany
   private Set<Product> suppliedProducts;

   public Supplier(String companyName, String street, String city) {
       CompanyName = companyName;
       Street = street;
       City = city;
   }

   public Supplier(){};

   public String getStreet() {
       return Street;
   }
```

```
    public void setStreet(String street) {
        Street = street;
    }

    public String getCity() {
        return City;
    }

    public void setCity(String city) {
        City = city;
    }

    public void addProduct(Product product){
        this.suppliedProducts.add(product);
        product.setSuppliedBy(this);
    }


}
```



	SUPPLIERID	CITY	COMPANYNAME	STREET
1	1	Krakow	Kowalski	Mickiewicza 12
2	101	Warszawa	Nowak	Slowackiego 22
3	242	Gdansk	ABC	Dluga
4	243	Gdynia	R3VEGE	Krotka 13
5	244	Warszawa	IzoSupplier	Wodna 41



	SUPPLIER_SUPPLIERID	SUPPLIEDPRODUCTS_PRODUCTID
1	242	263
2	242	264
3	243	261
4	244	260
5	244	262

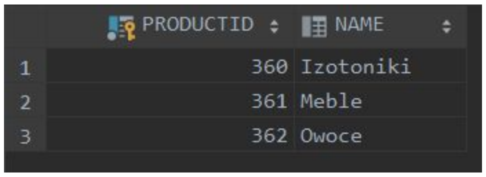
V Relacja obustronna suppliers - products - połączenie obu metod

VI Dodanie klasy Category

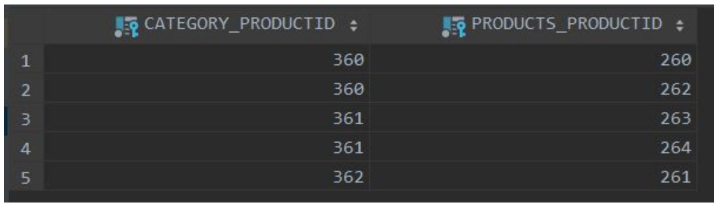
```
import javax.persistence.*;

@Entity
@Table(name = "PRODUCTS" )
public class Product {

   @Id
   @GeneratedValue(strategy = GenerationType.AUTO)
   private int productID;

   private String ProductName;
   private Integer UnitsOnStock;

   @ManyToOne
   @JoinColumn(name = "SUPPLIED_BY")
   private Supplier suppliedBy;

   @ManyToOne
   private Category category;

   public Product(String productName) {
       ProductName = productName;
   }

   public Product(String productName, Integer unitsOnStock, Supplier supplier) {
       ProductName = productName;
       UnitsOnStock = unitsOnStock;
       this.suppliedBy = supplier;
   }

   public Product(String productName, Integer unitsOnStock) {
       ProductName = productName;
       UnitsOnStock = unitsOnStock;
   }

   public Product() {
   }


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


   public void setCategory(Category category){
       this.category = category;
       if(!category.getProducts().contains(this)){
           category.addProduct(this);
       }
   }
}
```

```
import org.hibernate.mapping.Collection;

import javax.persistence.*;
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

@Entity
@Table(name = "CATEGORIES" )
public class Category {

   @Id
   @GeneratedValue(strategy = GenerationType.AUTO)
   private int productID;
   private String Name;

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

   public Category(){};

   public Category(String name){
       this.Name = name;
   }

   public void addProduct(Product product){
       this.products.add(product);
       product.setCategory(this);
   }

   public List<Product> getProducts(){
       return this.products; //
   }


}
//nt CategoryID, String Name oraz listą produktow
//List<Product> Products
```

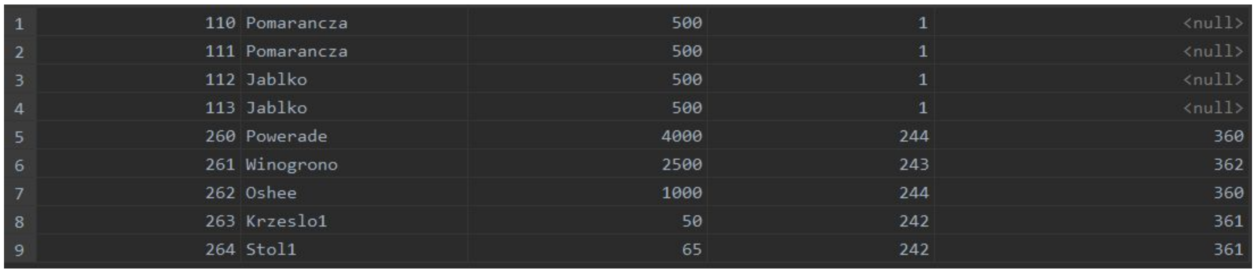
Stan bazy po dodaniu paru kategorii.



	PRODUCTID	NAME
1	360	Izotoniki
2	361	Meble
3	362	Owoce

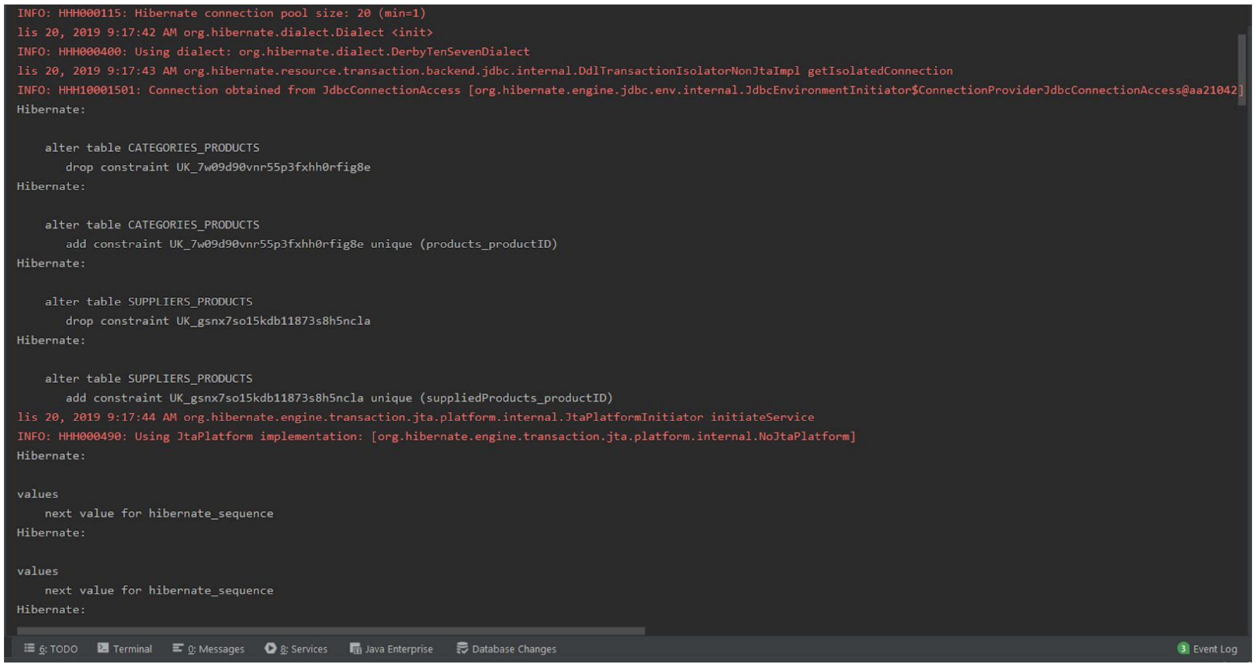


	CATEGORY_PRODUCTID	PRODUCTS_PRODUCTID
1	360	260
2	360	262
3	361	263
4	361	264
5	362	261



1	110	Pomarancza	500	1	<null>
2	111	Pomarancza	500	1	<null>
3	112	Jablko	500	1	<null>
4	113	Jablko	500	1	<null>
5	260	Powerade	4000	244	360
6	261	Winogrono	2500	243	362
7	262	Oshee	1000	244	360
8	263	Krzeslo1	50	242	361
9	264	Stol1	65	242	361

Logi wywołań SQLowych.



```
INFO: HHH000115: Hibernate connection pool size: 20 (min=1)
lis 20, 2019 9:17:42 AM org.hibernate.dialect.Dialect <init>
INFO: HHH000400: Using dialect: org.hibernate.dialect.DerbyTenSevenDialect
lis 20, 2019 9:17:43 AM org.hibernate.resource.transaction.backend.jdbc.internal.DdlTransactionIsolatorNonJtaImpl getIsolatedConnection
INFO: HHH10001501: Connection obtained from JdbcConnectionAccess [org.hibernate.engine.jdbc.env.internal.JdbcEnvironmentInitiator$ConnectionProviderJdbcConnectionAccess@aa21042]
Hibernate: 

    alter table CATEGORIES_PRODUCTS 
       drop constraint UK_7w09d90vnr55p3fxhh0rfig8e
Hibernate: 

    alter table CATEGORIES_PRODUCTS 
       add constraint UK_7w09d90vnr55p3fxhh0rfig8e unique (products_productID)
Hibernate: 

    alter table SUPPLIERS_PRODUCTS 
       drop constraint UK_gsnx7so15kdb11873s8h5ncla
Hibernate: 

    alter table SUPPLIERS_PRODUCTS 
       add constraint UK_gsnx7so15kdb11873s8h5ncla unique (suppliedProducts_productID)
lis 20, 2019 9:17:44 AM org.hibernate.engine.transaction.jta.platform.internal.JtaPlatformInitiator initiateService
INFO: HHH000490: Using JtaPlatform implementation: [org.hibernate.engine.transaction.jta.platform.internal.NoJtaPlatform]
Hibernate: 

values
    next value for hibernate_sequence
Hibernate: 

values
    next value for hibernate_sequence
Hibernate: 
```

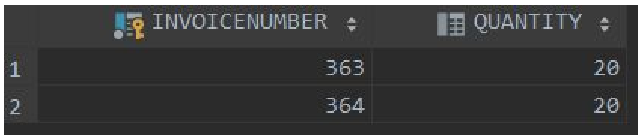
VII Dodanie klasy Invoices (relacja M-N z Products)

```
import javax.persistence.*;
import java.util.ArrayList;
import java.util.HashSet;
import java.util.List;
import java.util.Set;

@Entity
@Table(name = "INVOICES" )
public class Invoice {

   @Id
   @GeneratedValue(strategy = GenerationType.AUTO)
   private int InvoiceNumber;
   private Integer quantity;

   @ManyToMany
   private Set<Product> includesProducts = new HashSet<>();

   public Invoice(){}

   public Invoice(Product product, int quantity){
       this.includesProducts.add(product);
       this.quantity = quantity;
   }

   public void addProduct(Product product, int quantity){
       this.includesProducts.add(product);
       this.quantity += quantity;
   }


}
```

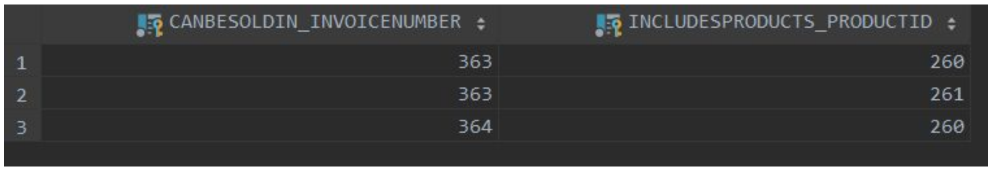
```
import javax.persistence.*;
import java.util.Set;

@Entity
@Table(name = "PRODUCTS" )
public class Product {

   @Id
   @GeneratedValue(strategy = GenerationType.AUTO)
   private int productID;

   private String ProductName;
   private Integer UnitsOnStock;
```

```
    @ManyToOne
    @JoinColumn(name = "SUPPLIED_BY")
    private Supplier suppliedBy;

    @ManyToOne
    private Category category;

    @ManyToMany(mappedBy = "includesProducts")
    private Set<Invoice> canBeSoldIn;

    public Product(String productName) {
        ProductName = productName;
    }

    public Product(String productName, Integer unitsOnStock, Supplier supplier) {
        ProductName = productName;
        UnitsOnStock = unitsOnStock;
        this.suppliedBy = supplier;
    }

    public Product(String productName, Integer unitsOnStock) {
        ProductName = productName;
        UnitsOnStock = unitsOnStock;
    }

    public Product() {
    }


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


    public void setCategory(Category category){
        this.category = category;
        if(!category.getProducts().contains(this)){
            category.addProduct(this);
        }
    }
}
```

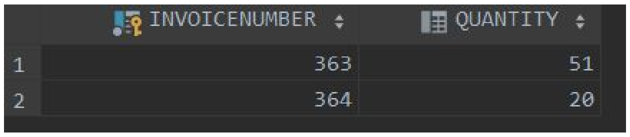
Stan bazy po dodaniu zamówień



	INVOICENUMBER	QUANTITY
1	363	20
2	364	20

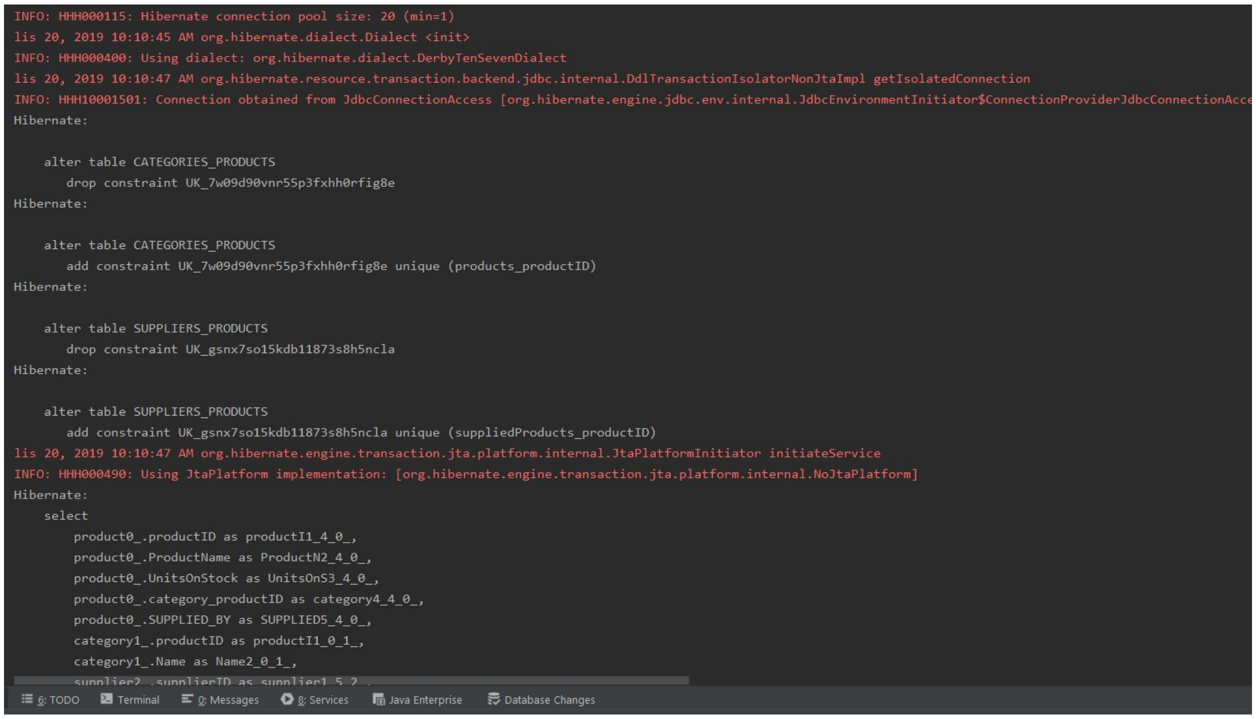


	CANBESOLDIN_INVOICENUMBER	INCLUDESPRODUCTS_PRODUCTID
1	363	260
2	363	261
3	364	260



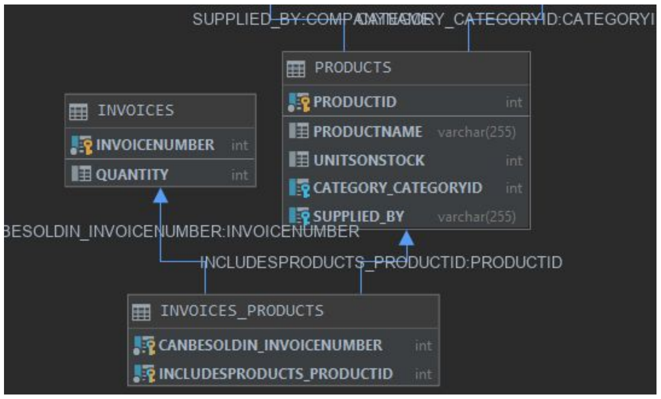
	INVOICENUMBER	QUANTITY
1	363	51
2	364	20

Logi wywołań SQLowych



```
INFO: HHH000115: Hibernate connection pool size: 20 (min=1)
lis 20, 2019 10:10:45 AM org.hibernate.dialect.Dialect <init>
INFO: HHH000400: Using dialect: org.hibernate.dialect.DerbyTenSevenDialect
lis 20, 2019 10:10:47 AM org.hibernate.resource.transaction.backend.jdbc.internal.DdlTransactionIsolatorNonJtaImpl getIsolatedConnection
INFO: HHH10001501: Connection obtained from JdbcConnectionAccess [org.hibernate.engine.jdbc.env.internal.JdbcEnvironmentInitiator$ConnectionProviderJdbcConnectionAcce
Hibernate: 

    alter table CATEGORIES_PRODUCTS 
       drop constraint UK_7w09d90vnr55p3fxhh0rfig8e
Hibernate: 

    alter table CATEGORIES_PRODUCTS 
       add constraint UK_7w09d90vnr55p3fxhh0rfig8e unique (products_productID)
Hibernate: 

    alter table SUPPLIERS_PRODUCTS 
       drop constraint UK_gsnx7so15kdb11873s8h5ncla
Hibernate: 

    alter table SUPPLIERS_PRODUCTS 
       add constraint UK_gsnx7so15kdb11873s8h5ncla unique (suppliedProducts_productID)
lis 20, 2019 10:10:47 AM org.hibernate.engine.transaction.jta.platform.internal.JtaPlatformInitiator initiateService
INFO: HHH000490: Using JtaPlatform implementation: [org.hibernate.engine.transaction.jta.platform.internal.NoJtaPlatform]
Hibernate: 
    select
        product0_.productID as productI1_4_0_,
        product0_.ProductName as ProductN2_4_0_,
        product0_.UnitsOnStock as UnitsOnS3_4_0_,
        product0_.category_productID as category4_4_0_,
        product0_.SUPPLIED_BY as SUPPLIED5_4_0_,
        category1_.productID as productI1_0_1_,
        category1_.Name as Name2_0_1_,
        supplier2_.supplierID as supplier1_5_2_,
```

Schemat:

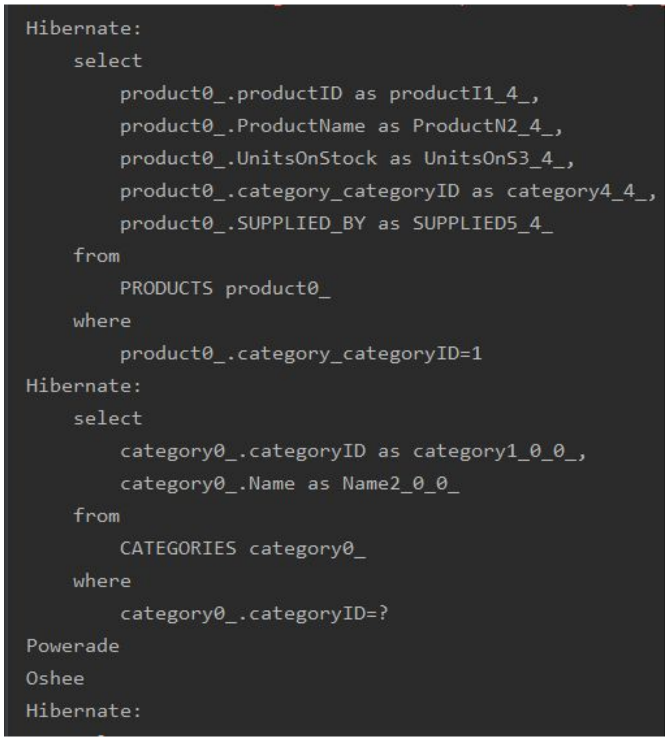


VIII Kontynuacja
XI JPI - relacja Products- Supplier

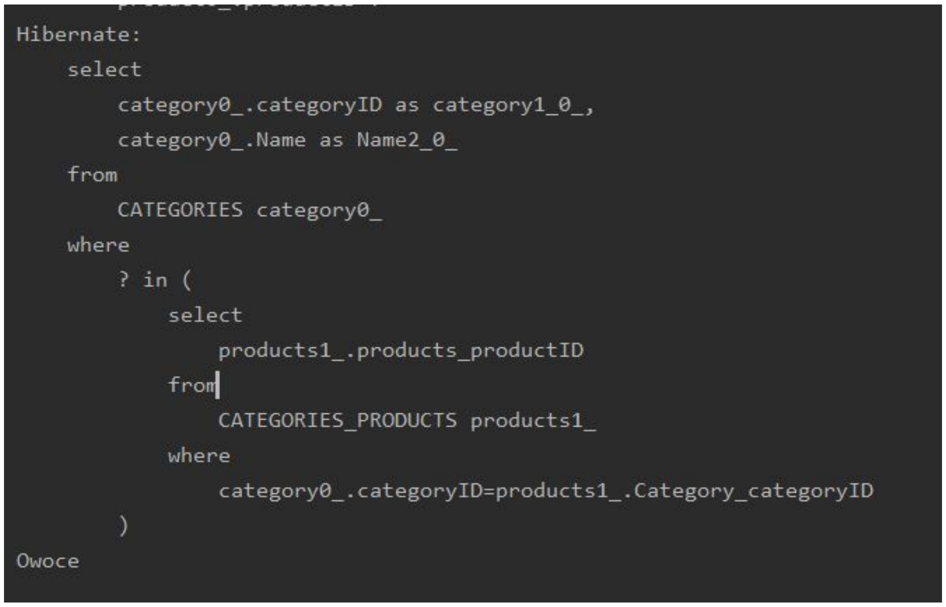
```
public class MainJPA {
   public static EntityManagerFactory entityManagerFactory = null;

   public static void main(String argv[]) {

       EntityManager em = getEntityManager();

       EntityTransaction transaction = em.getTransaction();
       transaction.begin();

       Category c1 = new Category("Izotoniki");
       Category c2 = new Category("Meble");
       Category c3 = new Category("Owoce");
       Category c4 = new Category("Nabial");

       em.persist(c1);
       em.persist(c2);
       em.persist(c3);
       em.persist(c4);

/*
       Category c1 = em.find(Category.class, 1);
       Category c2 = em.find(Category.class, 2);
       Category c3 = em.find(Category.class, 3);
       Category c4 = em.find(Category.class, 4);

       Product p1 = new Product("Powerade", 1000);
       Product p2 = new Product("Oshee", 1240);
       Product p3 = new Product("Krzeslo1", 232);
       Product p4 = new Product("Lozko1", 12);
```

```
        Product p5 = new Product("Jablko", 1000);
        Product p6 = new Product("Banan", 600);
        Product p7 = new Product("Mleko1", 100);
        Product p8 = new Product("Ser zolty1", 250);

        em.persist(p1);
        em.persist(p2);
        em.persist(p3);
        em.persist(p4);
        em.persist(p5);
        em.persist(p6);
        em.persist(p7);
        em.persist(p8);


        c1.addProduct(p1);
        c1.addProduct(p2);
        c2.addProduct(p3);
        c2.addProduct(p4);
        c3.addProduct(p5);
        c3.addProduct(p6);
        c4.addProduct(p7);
        c4.addProduct(p8);
*/
        TypedQuery<Product> query = em.createQuery("from Product as product" +
                " where product.category.categoryID= 1", Product.class);
        List<Product> allProducts = query.getResultList();
        for(Product prod: allProducts){
            System.out.println(prod.getProductName());
        }

        Product p1 = em.find(Product.class, 17);

        TypedQuery<Category> query2 = em.createQuery("from Category as category" +
               " where :p member category.products", Category.class);
        query2.setParameter("p", p1);
        Category cat = query2.getSingleResult();
        System.out.println(cat.getName());

        transaction.commit();
        em.close();

    }
    private static EntityManager getEntityManager() {
        if (entityManagerFactory == null) {
            entityManagerFactory = Persistence.createEntityManagerFactory("derby");
        }
        return entityManagerFactory.createEntityManager();
    }


}
```

Efekt wywołania:

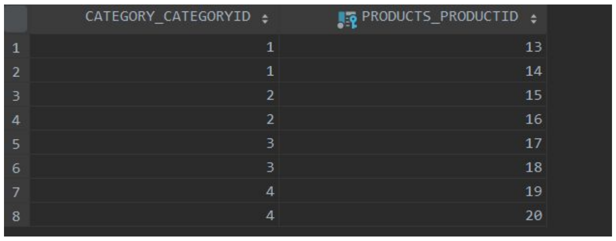


```
Hibernate: 
    select
        product0_.productID as productI1_4_,
        product0_.ProductName as ProductN2_4_,
        product0_.UnitsOnStock as UnitsOnS3_4_,
        product0_.category_categoryID as category4_4_,
        product0_.SUPPLIED_BY as SUPPLIED5_4_ 
    from
        PRODUCTS product0_ 
    where
        product0_.category_categoryID=1
Hibernate: 
    select
        category0_.categoryID as category1_0_0_,
        category0_.Name as Name2_0_0_ 
    from
        CATEGORIES category0_ 
    where
        category0_.categoryID=?
Powerade
Oshee
Hibernate: 
```

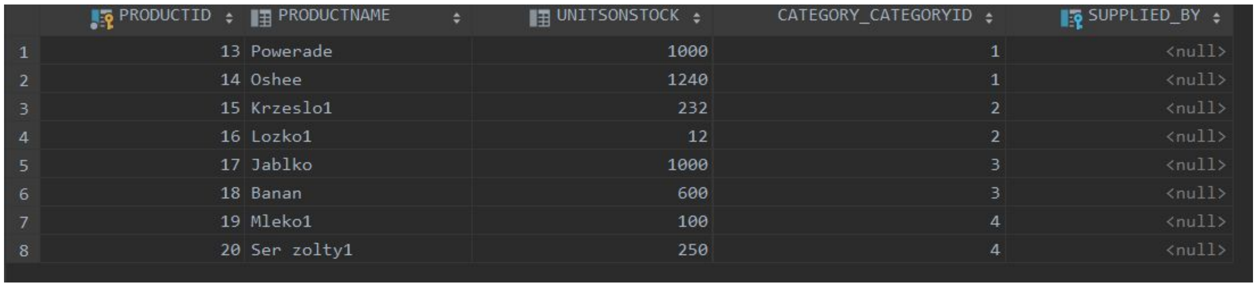


```
Hibernate: 
    select
        category0_.categoryID as category1_0_,
        category0_.Name as Name2_0_ 
    from
        CATEGORIES category0_ 
    where
        ? in (
            select
                products1_.products_productID 
            from
                CATEGORIES_PRODUCTS products1_ 
            where
                category0_.categoryID=products1_.Category_categoryID
        )
Owoce
```

Stan bazy:

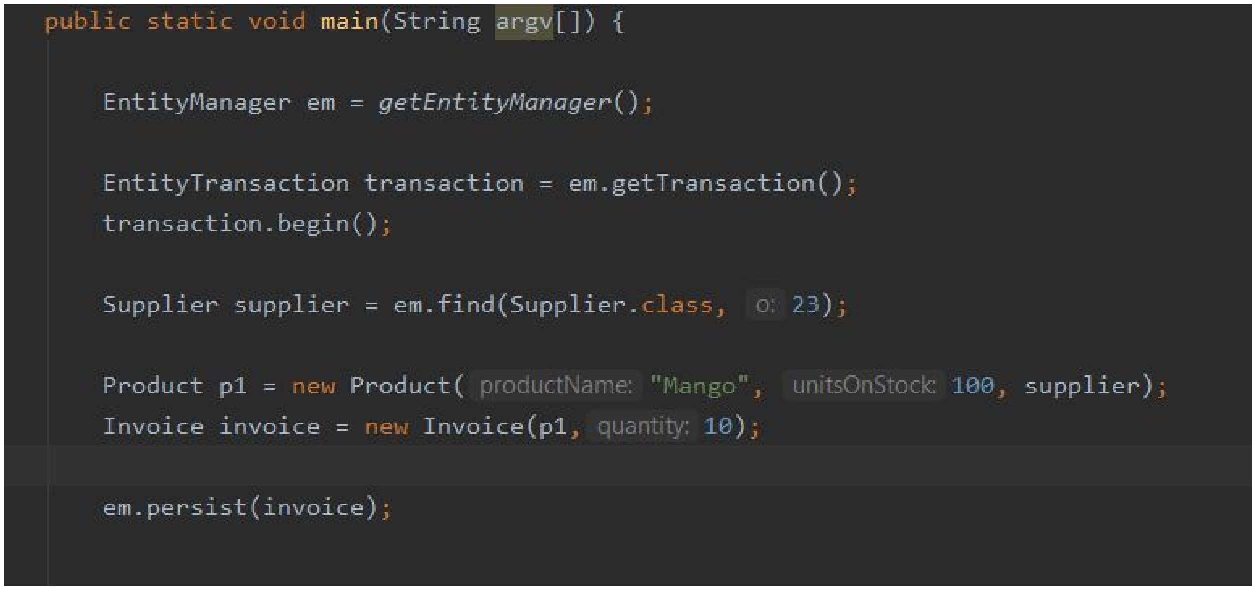


	CATEGORY_CATEGORYID	PRODUCTS_PRODUCTID
1	1	13
2	1	14
3	2	15
4	2	16
5	3	17
6	3	18
7	4	19
8	4	20

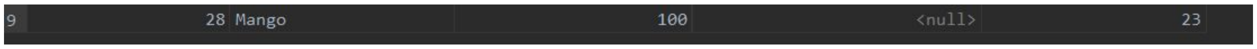


	PRODUCTID	PRODUCTNAME	UNITSONSTOCK	CATEGORY_CATEGORYID	SUPPLIED_BY
1	13	Powerade	1000	1	<null>
2	14	Oshee	1240	1	<null>
3	15	Krzeslo1	232	2	<null>
4	16	Lozko1	12	2	<null>
5	17	Jablko	1000	3	<null>
6	18	Banan	600	3	<null>
7	19	Mleko1	100	4	<null>
8	20	Ser zolty1	250	4	<null>

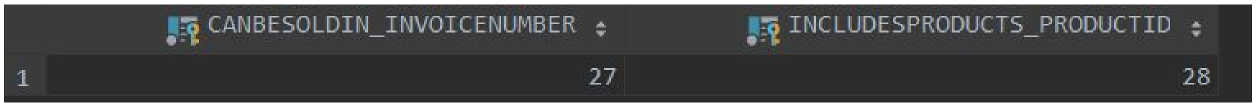
X Kaskady - tworzenie faktur z produktami



```java
public static void main(String argv[]) {

    EntityManager em = getEntityManager();

    EntityTransaction transaction = em.getTransaction();
    transaction.begin();

    Supplier supplier = em.find(Supplier.class,  o: 23);

    Product p1 = new Product( productName: "Mango",  unitsOnStock: 100, supplier);
    Invoice invoice = new Invoice(p1,  quantity: 10);

    em.persist(invoice);

```

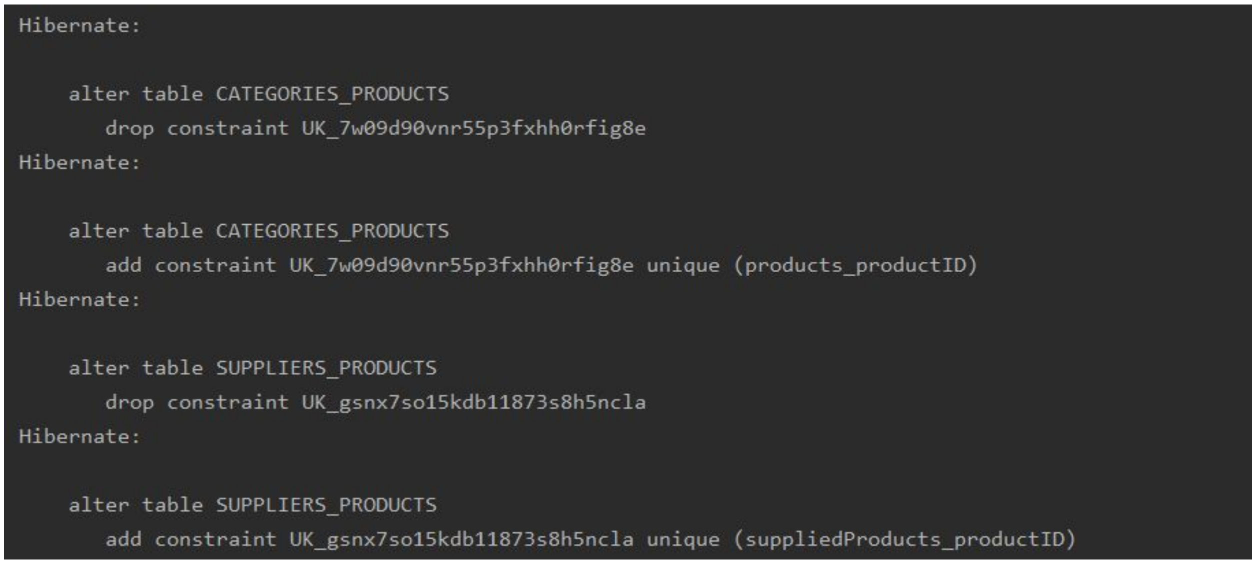


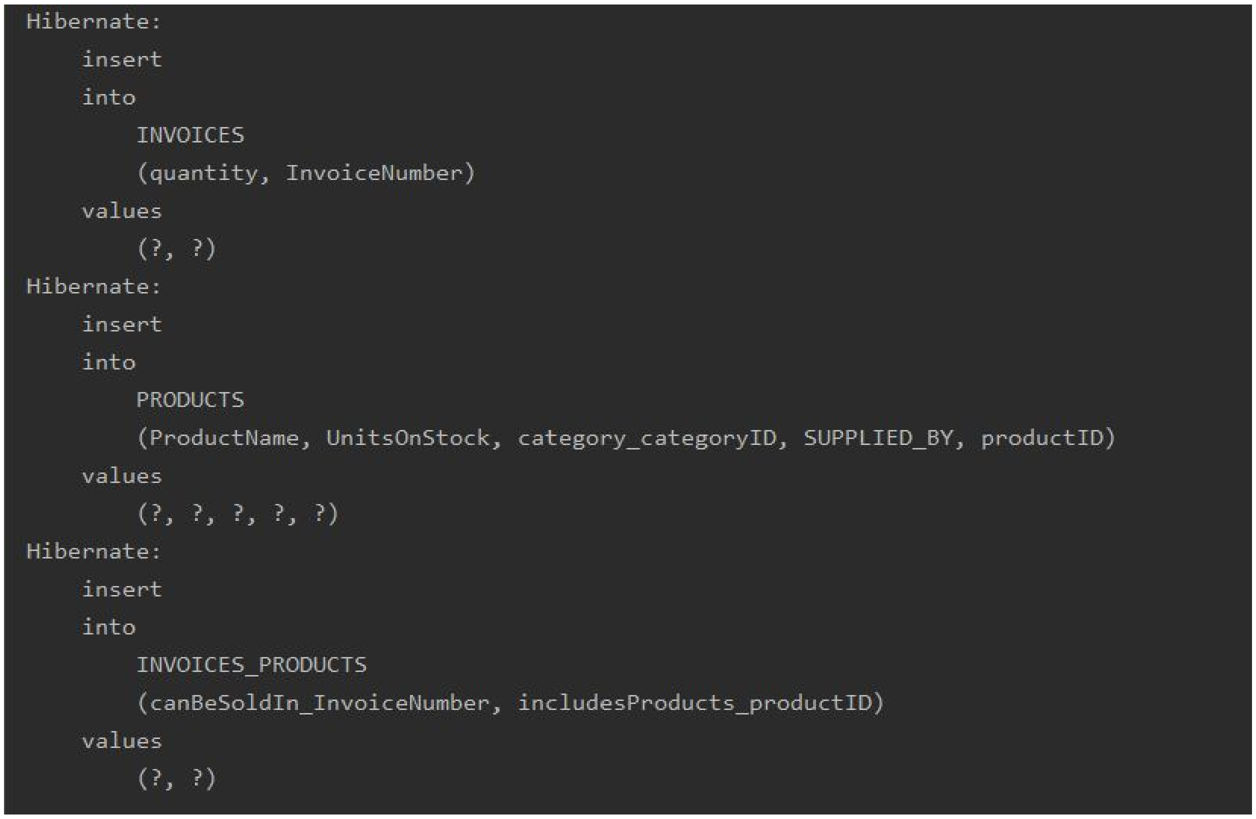
9	28	Mango	100	<null>	23

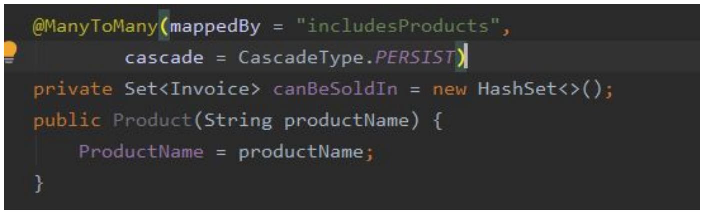


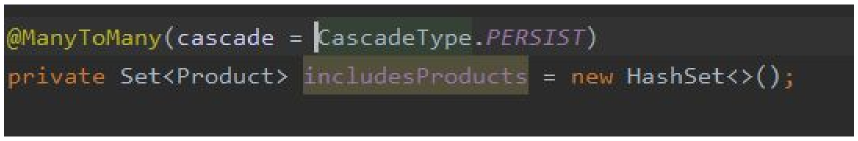
	CANBESOLDIN_INVOICENUMBER	INCLUDESPRODUCTS_PRODUCTID
1	27	28

Efekt:

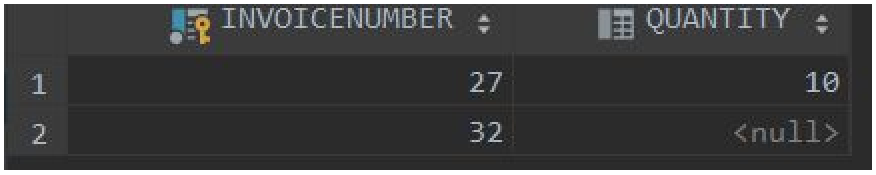

```
Hibernate: 

    alter table CATEGORIES_PRODUCTS 
       drop constraint UK_7w09d90vnr55p3fxhh0rfig8e
Hibernate: 

    alter table CATEGORIES_PRODUCTS 
       add constraint UK_7w09d90vnr55p3fxhh0rfig8e unique (products_productID)
Hibernate: 

    alter table SUPPLIERS_PRODUCTS 
       drop constraint UK_gsnx7so15kdb11873s8h5ncla
Hibernate: 

    alter table SUPPLIERS_PRODUCTS 
       add constraint UK_gsnx7so15kdb11873s8h5ncla unique (suppliedProducts_productID)
```


```
Hibernate: 
    insert 
    into
        INVOICES
        (quantity, InvoiceNumber) 
    values
        (?, ?)
Hibernate: 
    insert 
    into
        PRODUCTS
        (ProductName, UnitsOnStock, category_categoryID, SUPPLIED_BY, productID) 
    values
        (?, ?, ?, ?, ?)
Hibernate: 
    insert 
    into
        INVOICES_PRODUCTS
        (canBeSoldIn_InvoiceNumber, includesProducts_productID) 
    values
        (?, ?)
```


```
@ManyToMany(mappedBy = "includesProducts",
        cascade = CascadeType.PERSIST)
private Set<Invoice> canBeSoldIn = new HashSet<>();
public Product(String productName) {
    ProductName = productName;
}
```



```
@ManyToMany(cascade = CascadeType.PERSIST)
private Set<Product> includesProducts = new HashSet<>();
```



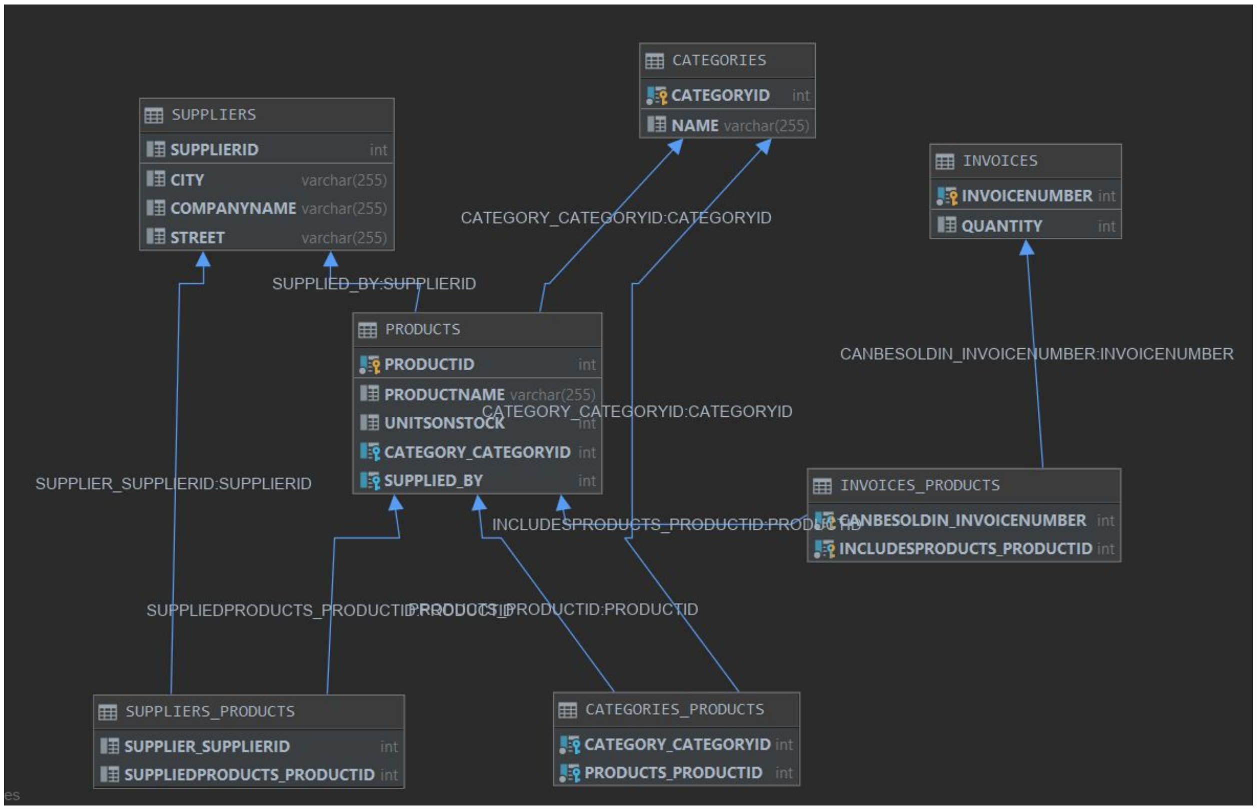
	INVOICENUMBER	QUANTITY
1	27	10
2	32	<null>

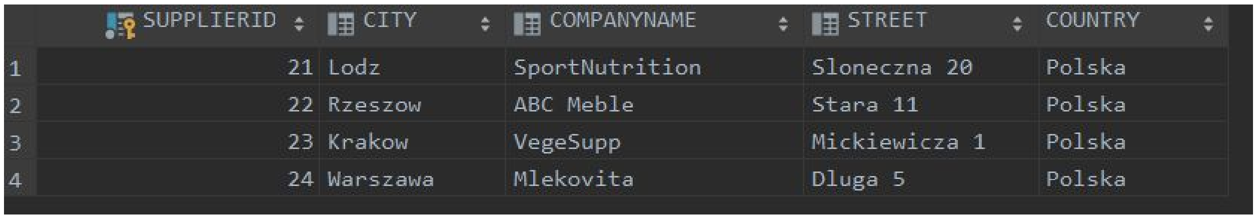
XI Klasa wbudowana - Address

```
import javax.persistence.Embeddable;

@Embeddable
public class Address {
   private String Street;
   private String City;
   private String Country;

   public Address() {
   }

   public Address(String street, String city, String country) {
       Street = street;
       City = city;
       Country = country;
   }

   public String getCity() {
       return City;
   }

   public void setCity(String city) {
       City = city;
   }

   public String getStreet() {
       return Street;
   }

   public void setStreet(String street) {
       Street = street;
   }
```

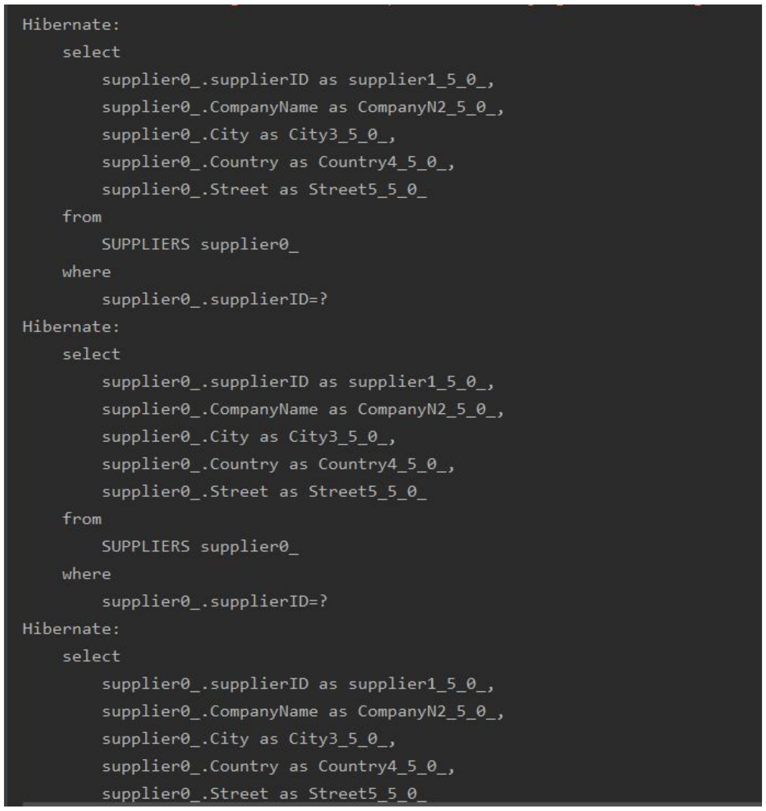
```java
   public String getCountry() {
       return Country;
   }

   public void setCountry(String country) {
       Country = country;
   }


}


import org.hibernate.dialect.ProgressDialect;

import javax.persistence.*;
import java.util.HashSet;
import java.util.List;
import java.util.Set;

@Entity
@Table(name = "SUPPLIERS" )

public class Supplier {
   @Id
   @GeneratedValue(strategy = GenerationType.AUTO)
   private int supplierID;

   private String CompanyName;

   @Embedded
   private Address address;
//    private String Street;
 //  private String City;

   @OneToMany
   private Set<Product> suppliedProducts = new HashSet<>();

   public Supplier(String companyName, String street, String city, String country)
{
       CompanyName = companyName;
       address = new Address(street, city, country);
   }

   public Supplier(){};

   public String getStreet() {
        return this.address.getStreet();
    }

    public void setStreet(String street) {
        this.address.setStreet(street);
    }

    public String getCity() {
        return this.address.getCity();
    }

    public void setCity(String city) {
        this.address.setCity(city);
    }

    public void addProduct(Product product){
        this.suppliedProducts.add(product);
        product.setSuppliedBy(this);
    }

     public void setCountry(String country){
        this.address.setCountry(country);
     }


}
```

Schemat bazy:



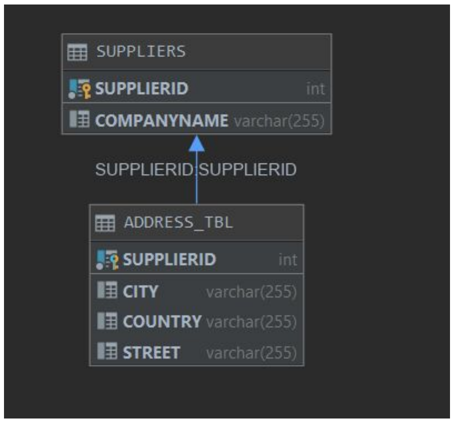


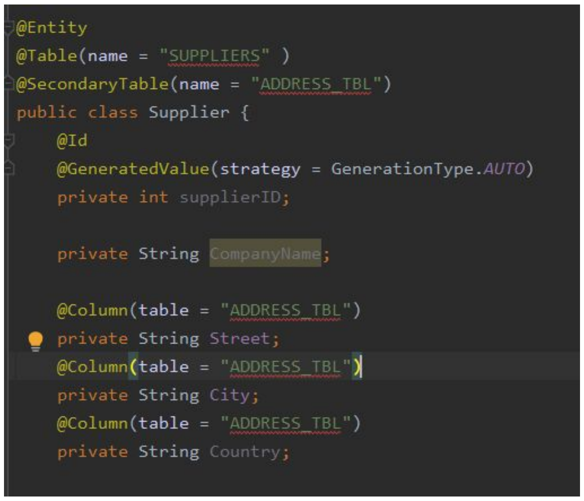
	SUPPLIERID	CITY	COMPANYNAME	STREET	COUNTRY
1	21	Lodz	SportNutrition	Sloneczna 20	Polska
2	22	Rzeszow	ABC Meble	Stara 11	Polska
3	23	Krakow	VegeSupp	Mickiewicza 1	Polska
4	24	Warszawa	Mlekovita	Dluga 5	Polska



```
Hibernate:
    select
        supplier0_.supplierID as supplier1_5_0_,
        supplier0_.CompanyName as CompanyN2_5_0_,
        supplier0_.City as City3_5_0_,
        supplier0_.Country as Country4_5_0_,
        supplier0_.Street as Street5_5_0_
    from
        SUPPLIERS supplier0_
    where
        supplier0_.supplierID=?
Hibernate:
    select
        supplier0_.supplierID as supplier1_5_0_,
        supplier0_.CompanyName as CompanyN2_5_0_,
        supplier0_.City as City3_5_0_,
        supplier0_.Country as Country4_5_0_,
        supplier0_.Street as Street5_5_0_
    from
        SUPPLIERS supplier0_
    where
        supplier0_.supplierID=?
Hibernate:
    select
        supplier0_.supplierID as supplier1_5_0_,
        supplier0_.CompanyName as CompanyN2_5_0_,
        supplier0_.City as City3_5_0_,
        supplier0_.Country as Country4_5_0_,
        supplier0_.Street as Street5_5_0_
```

Mapowanie do dwóch tabel

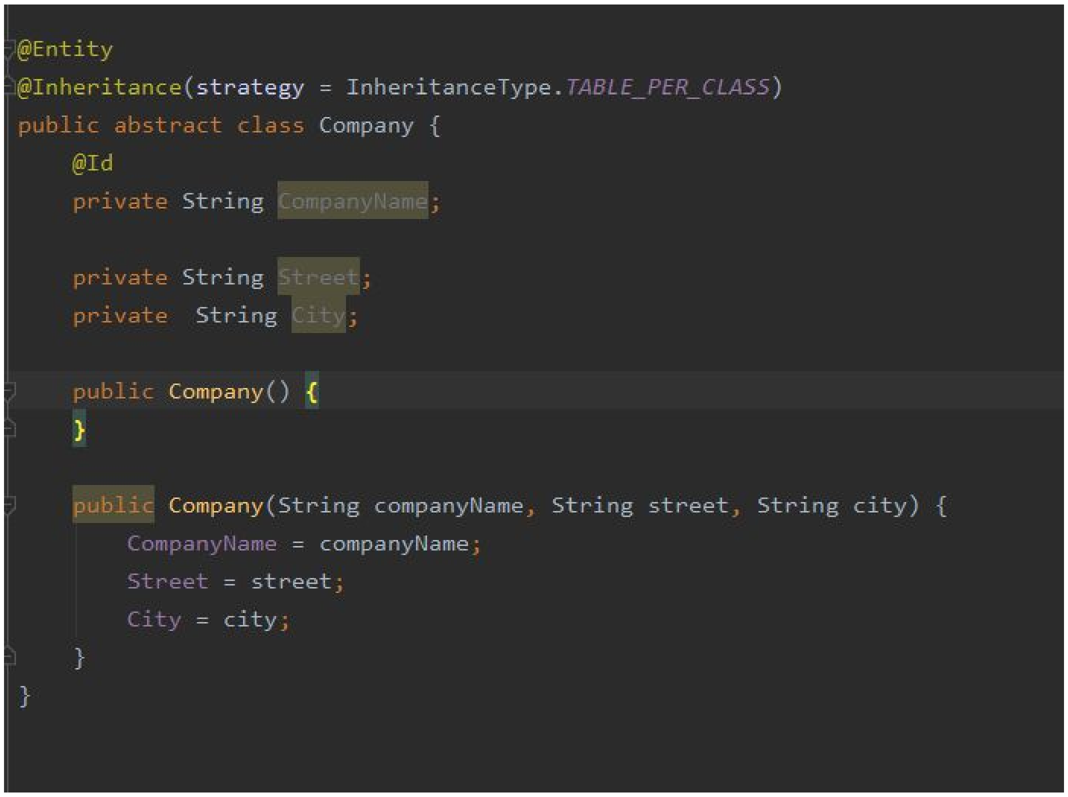




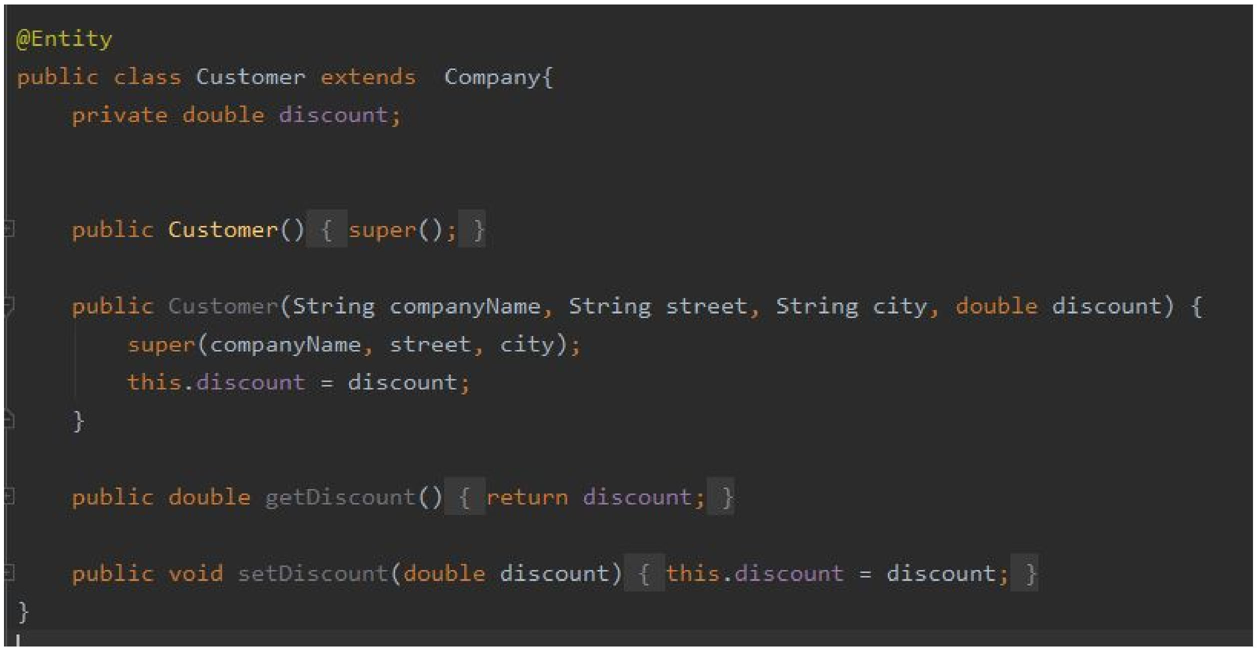
```java
@Entity
@Table(name = "SUPPLIERS" )
@SecondaryTable(name = "ADDRESS_TBL")
public class Supplier {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int supplierID;

    private String CompanyName;

    @Column(table = "ADDRESS_TBL")
    private String Street;
    @Column(table = "ADDRESS_TBL")
    private String City;
    @Column(table = "ADDRESS_TBL")
    private String Country;
```

XII Dziedziczenie

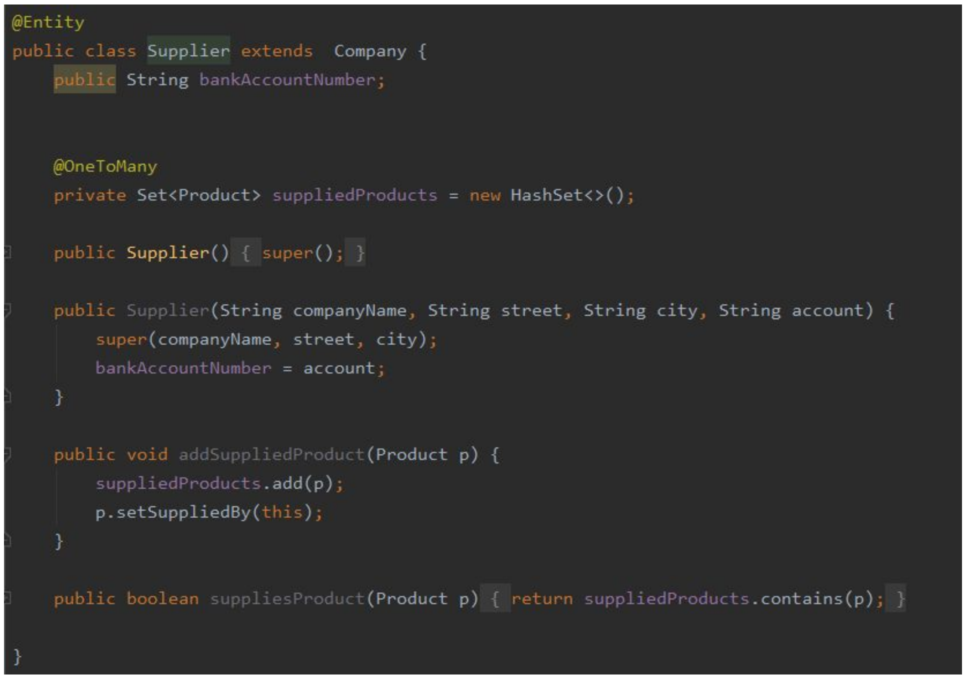
Tabela na klasę



```java
@Entity
@Inheritance(strategy = InheritanceType.TABLE_PER_CLASS)
public abstract class Company {
    @Id
    private String CompanyName;

    private String Street;
    private  String City;

    public Company() {
    }

    public Company(String companyName, String street, String city) {
        CompanyName = companyName;
        Street = street;
        City = city;
    }
}
```

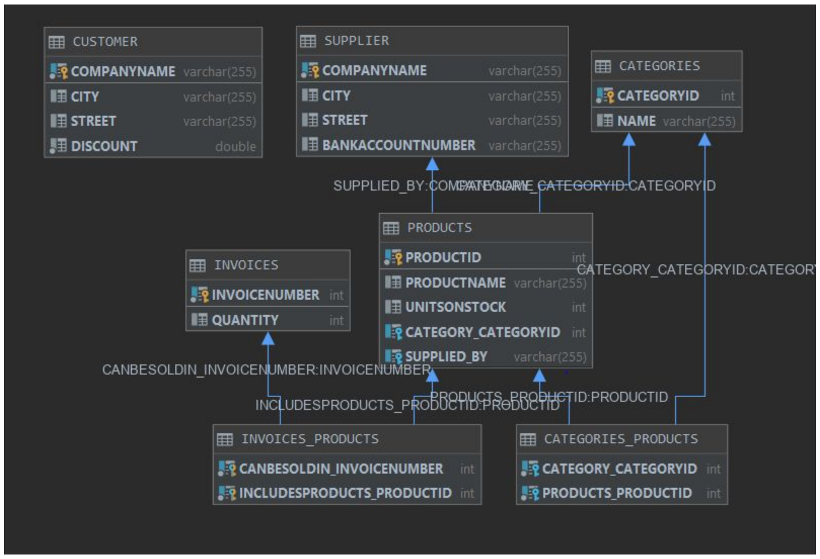


```java
@Entity
public class Customer extends  Company{
    private double discount;


    public Customer() { super(); }

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

    public double getDiscount() { return discount; }

    public void setDiscount(double discount) { this.discount = discount; }
}
```

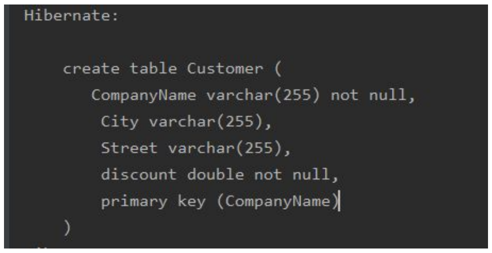


```java
@Entity
public class Supplier extends  Company {
    public String bankAccountNumber;


    @OneToMany
    private Set<Product> suppliedProducts = new HashSet<>();

    public Supplier() { super(); }

    public Supplier(String companyName, String street, String city, String account) {
        super(companyName, street, city);
        bankAccountNumber = account;
    }

    public void addSuppliedProduct(Product p) {
        suppliedProducts.add(p);
        p.setSuppliedBy(this);
    }

    public boolean suppliesProduct(Product p) { return suppliedProducts.contains(p); }

}
```

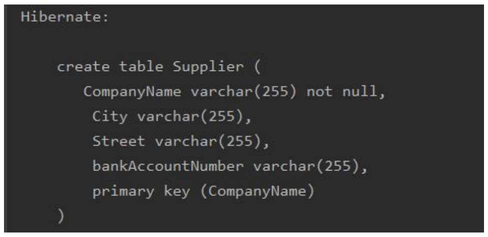
Schemat bazy:



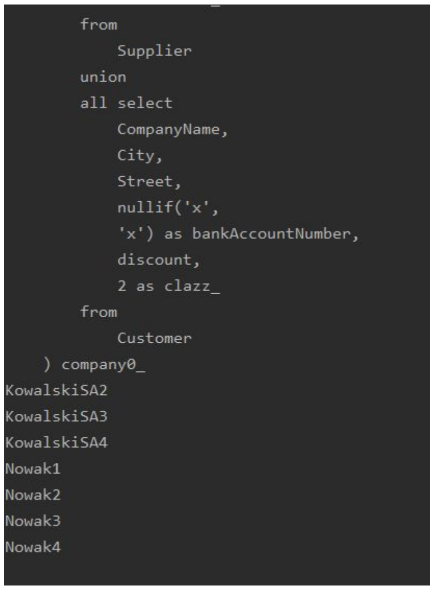
Logi SQLowe:



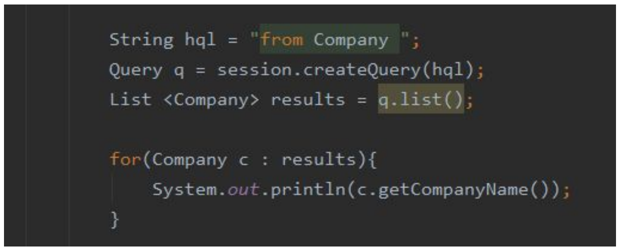
```
Hibernate:

    create table Customer (
       CompanyName varchar(255) not null,
        City varchar(255),
        Street varchar(255),
        discount double not null,
        primary key (CompanyName)
    )
```



```
Hibernate:

    create table Supplier (
       CompanyName varchar(255) not null,
        City varchar(255),
        Street varchar(255),
        bankAccountNumber varchar(255),
        primary key (CompanyName)
    )
```

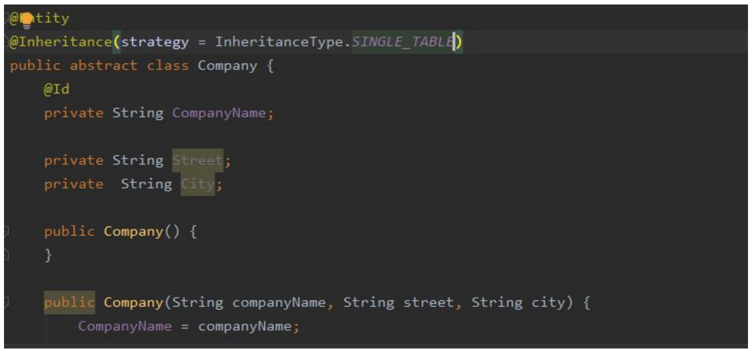


```
        from
            Supplier
        union
        all select
            CompanyName,
            City,
            Street,
            nullif('x',
            'x') as bankAccountNumber,
            discount,
            2 as clazz_
        from
            Customer
    ) company0_
KowalskiSA2
KowalskiSA3
KowalskiSA4
Nowak1
Nowak2
Nowak3
Nowak4
```

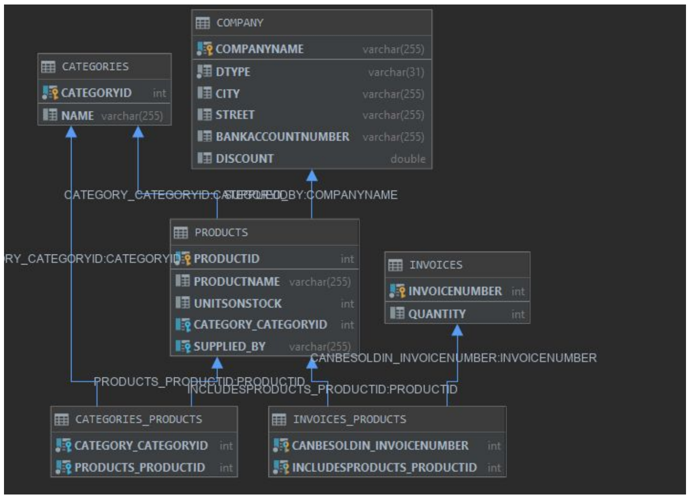


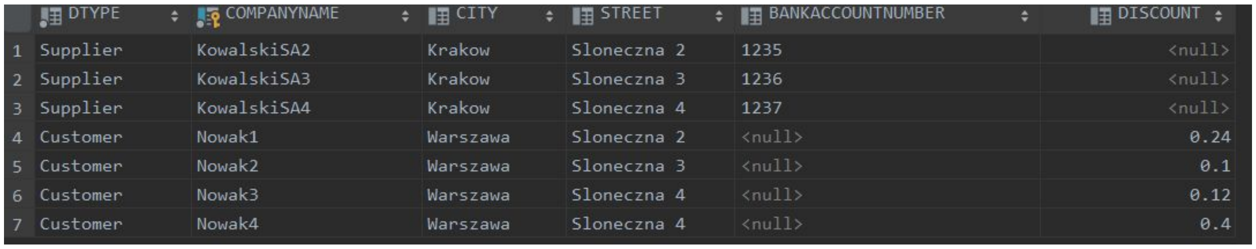
```
String hql = "from Company ";
Query q = session.createQuery(hql);
List <Company> results = q.list();

for(Company c : results){
    System.out.println(c.getCompanyName());
}
```

Pojedyncza tabela



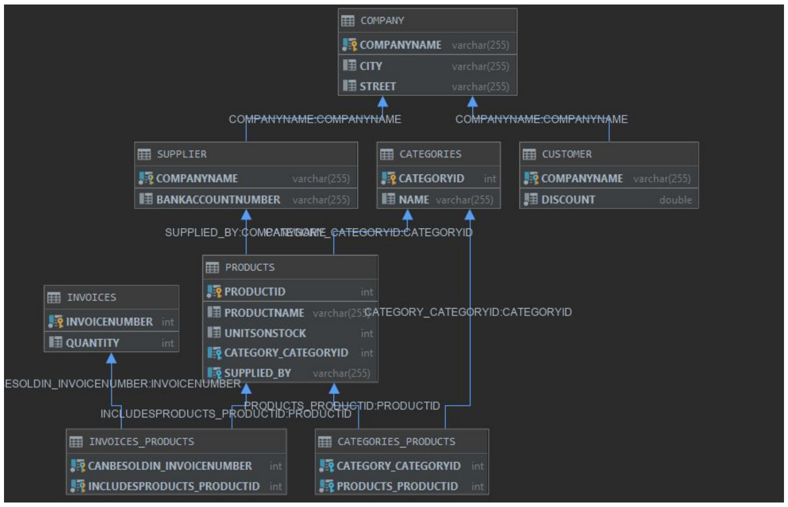
```
@Entity
@Inheritance(strategy = InheritanceType.SINGLE_TABLE)
public abstract class Company {
    @Id
    private String CompanyName;

    private String Street;
    private  String City;

    public Company() {
    }

    public Company(String companyName, String street, String city) {
        CompanyName = companyName;
```



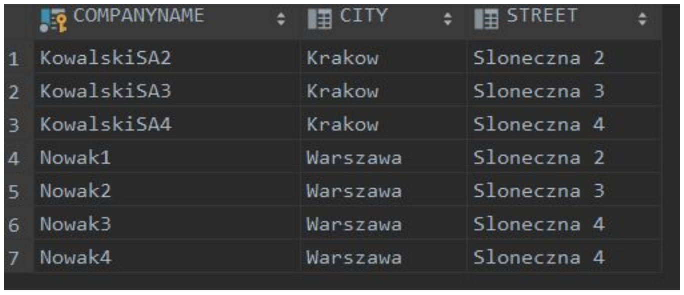


	DTYPE	COMPANYNAME	CITY	STREET	BANKACCOUNTNUMBER	DISCOUNT
1	Supplier	KowalskiSA2	Krakow	Sloneczna 2	1235	<null>
2	Supplier	KowalskiSA3	Krakow	Sloneczna 3	1236	<null>
3	Supplier	KowalskiSA4	Krakow	Sloneczna 4	1237	<null>
4	Customer	Nowak1	Warszawa	Sloneczna 2	<null>	0.24
5	Customer	Nowak2	Warszawa	Sloneczna 3	<null>	0.1
6	Customer	Nowak3	Warszawa	Sloneczna 4	<null>	0.12
7	Customer	Nowak4	Warszawa	Sloneczna 4	<null>	0.4

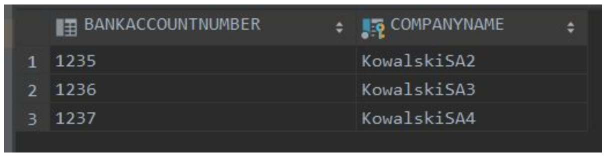
Tabele łączone



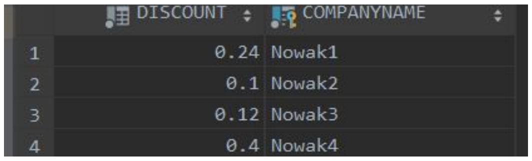
Stan bazy



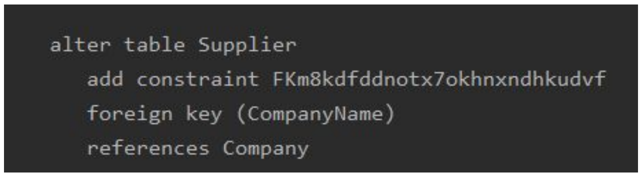
	COMPANYNAME	CITY	STREET
1	KowalskiSA2	Krakow	Sloneczna 2
2	KowalskiSA3	Krakow	Sloneczna 3
3	KowalskiSA4	Krakow	Sloneczna 4
4	Nowak1	Warszawa	Sloneczna 2
5	Nowak2	Warszawa	Sloneczna 3
6	Nowak3	Warszawa	Sloneczna 4
7	Nowak4	Warszawa	Sloneczna 4



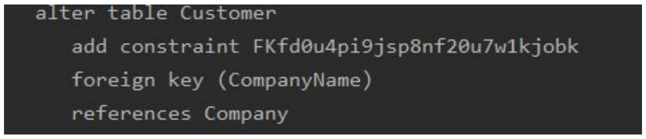
	BANKACCOUNTNUMBER	COMPANYNAME
1	1235	KowalskiSA2
2	1236	KowalskiSA3
3	1237	KowalskiSA4



	DISCOUNT	COMPANYNAME
1	0.24	Nowak1
2	0.1	Nowak2
3	0.12	Nowak3
4	0.4	Nowak4



```
alter table Supplier
   add constraint FKm8kdfddnotx7okhnxndhkudvf
   foreign key (CompanyName)
   references Company
```



```
alter table Customer
   add constraint FKfd0u4pi9jsp8nf20u7w1kjobk
   foreign key (CompanyName)
   references Company
```

Aplikacja do zamawiania

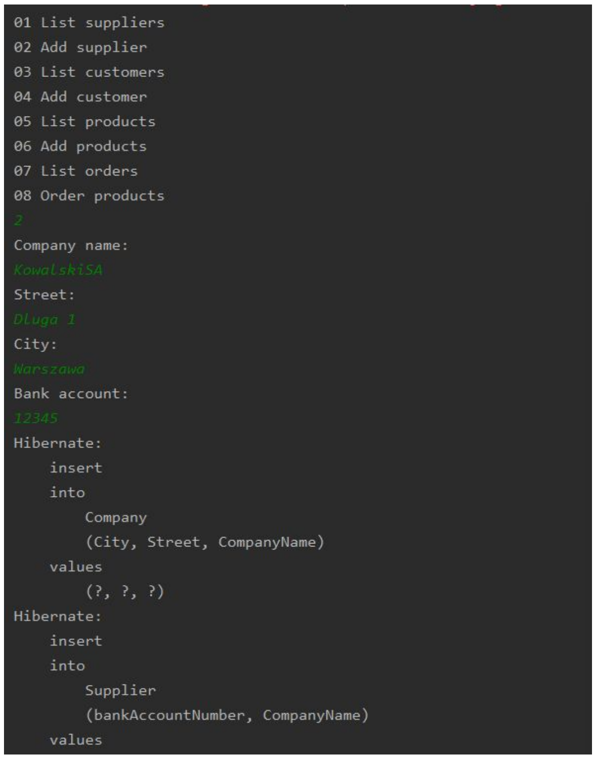
Postanowiłem zrealizować aplikację w formie aplikacji konsolowej.
Dodałem klasę Order.

```java
@Entity
@Table(name = "ORDERS")
public class Order {

   @Id
   @GeneratedValue(strategy = GenerationType.AUTO)
   private int OrderID;
   @ManyToOne(optional = false)
   private Customer customer;

   @ManyToMany
   private Set<Product> products = new HashSet<>();

   public Order() {
   }

   public int getOrderID() {
       return OrderID;
   }

   public Customer getCustomer() {
       return customer;
   }

   public void setCustomer(Customer customer) {
       this.customer = customer;
   }

   public Set<Product> getProducts() {
       return Collections.unmodifiableSet(products);
   }

   public void addProduct(Product product) {
       this.products.add(product);
   }
}
```

```
public class Menu {

   private String info;
   private Map<Integer, MenuOption> menuOptions = new HashMap<>();
   private boolean oneshot = true;

   public void setOneshot(boolean oneshot){
       this.oneshot = oneshot;
   }

   public Menu(String info){
       this.info = info;
   }

   public void addMenuOption(int index, String text, Consumer<Scanner> handler){
       this.menuOptions.put(index, new MenuOption(text, handler));
```

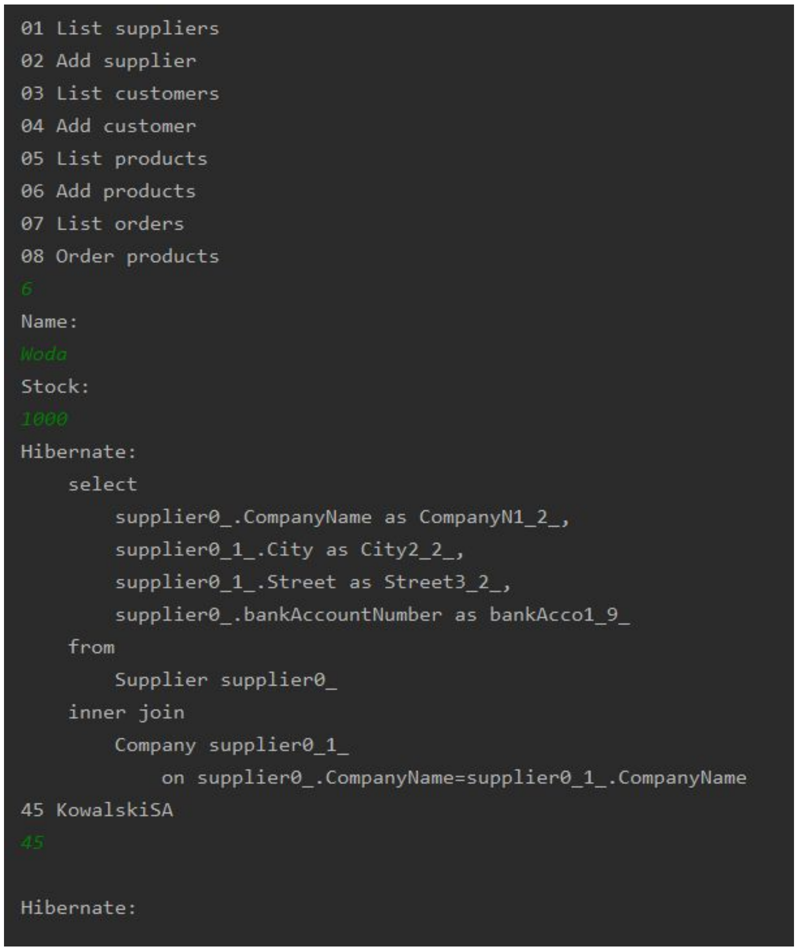
```
    }

    public void printOptions(){
        Scanner inputScanner = new Scanner(System.in);
        Set<Integer> keySet = menuOptions.keySet();
        do {
            menuOptions.forEach((index, option) ->
System.out.println(String.format("%d %s", index, option.getText())));
            Integer choice = null;
            do {
                choice = inputScanner.nextInt();
            } while (!menuOptions.containsKey(choice));

            this.menuOptions.get(choice).getHandler().accept(inputScanner);

            System.out.println();
        }while(!oneshot);
        //System.out.println(getText());
    }
}
```

Dodanie klienta:



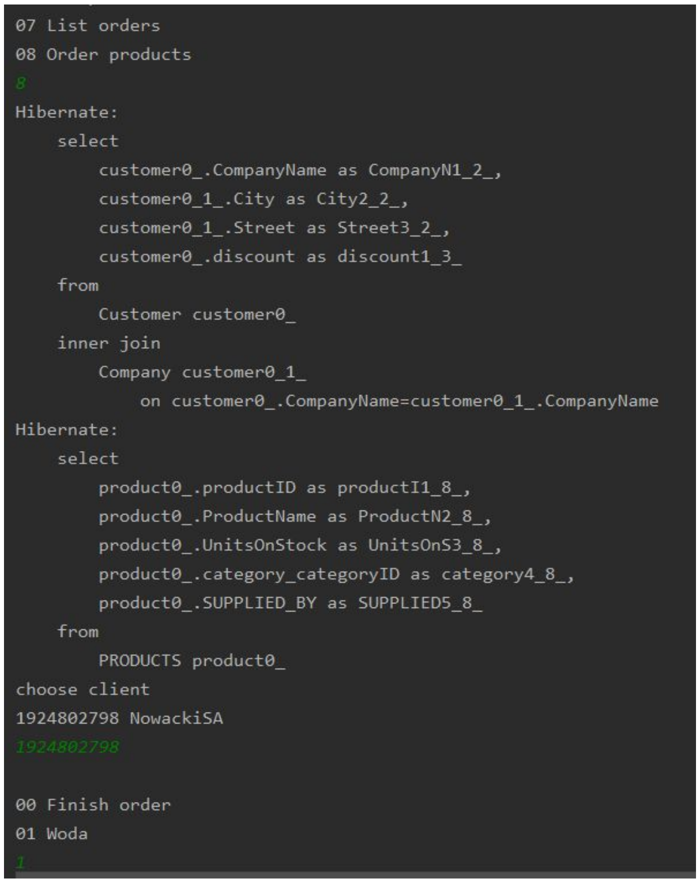
```
01 List suppliers
02 Add supplier
03 List customers
04 Add customer
05 List products
06 Add products
07 List orders
08 Order products
2
Company name:
KowalskiSA
Street:
Dluga 1
City:
Warszawa
Bank account:
12345
Hibernate:
    insert
    into
        Company
        (City, Street, CompanyName)
    values
        (?, ?, ?)
Hibernate:
    insert
    into
        Supplier
        (bankAccountNumber, CompanyName)
    values
```

Dodanie produktu



```
01 List suppliers
02 Add supplier
03 List customers
04 Add customer
05 List products
06 Add products
07 List orders
08 Order products
6
Name:
Woda
Stock:
1000
Hibernate:
    select
        supplier0_.CompanyName as CompanyN1_2_,
        supplier0_1_.City as City2_2_,
        supplier0_1_.Street as Street3_2_,
        supplier0_.bankAccountNumber as bankAcco1_9_
    from
        Supplier supplier0_
    inner join
        Company supplier0_1_
            on supplier0_.CompanyName=supplier0_1_.CompanyName
45 KowalskiSA
45

Hibernate:
```

Dodanie zamówienia:



```
07 List orders
08 Order products
8
Hibernate:
    select
        customer0_.CompanyName as CompanyN1_2_,
        customer0_1_.City as City2_2_,
        customer0_1_.Street as Street3_2_,
        customer0_.discount as discount1_3_
    from
        Customer customer0_
    inner join
        Company customer0_1_
            on customer0_.CompanyName=customer0_1_.CompanyName
Hibernate:
    select
        product0_.productID as productI1_8_,
        product0_.ProductName as ProductN2_8_,
        product0_.UnitsOnStock as UnitsOnS3_8_,
        product0_.category_categoryID as category4_8_,
        product0_.SUPPLIED_BY as SUPPLIED5_8_
    from
        PRODUCTS product0_
choose client
1924802798 NowackiSA
1924802798

00 Finish order
01 Woda
1
```