AGH Akademia Górniczo-Hutnicza
im. Stanisława Staszica w Krakowie

Bazy danych 2022/2023

Hibernate ORM

Laboratorium nr 4

4 maja 2023

Jakub Szaredko

Spis treści

2 Relacja Supplier <- Product

Klasa Product

```java
package model;

import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;

@Entity
public class Product {
    @Id
    private Long ProductID;
    private String ProductName;
    private int UnitsOnStock;
    @ManyToOne
    @JoinColumn(name = "SupplierID")
    private Supplier supplier;

    public Product() {  }

    public Product(String name, int unitsOnStock, Supplier supplier) {
        ProductName = name;
        UnitsOnStock = unitsOnStock;
        this.supplier = supplier;
    }

    public Long getProductID() {
        return ProductID;
    }

    public void setProductID(Long productID) {
        ProductID = productID;
    }

    public String getProductName() {
        return ProductName;
    }

    public void setProductName(String productName) {
        ProductName = productName;
    }

    public int getUnitsOnStock() {
        return UnitsOnStock;
    }

    public void setUnitsOnStock(int unitsOnStock) {
        UnitsOnStock = unitsOnStock;
    }
}
```

Klasa Supplier

```java
package model;

import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.OneToMany;
import java.util.ArrayList;
import java.util.List;
```

```java
@Entity
public class Supplier {
    @Id
    private Long SupplierID;
    private String CompanyName;
    private String Street;
    private String City;
    @OneToMany(mappedBy = "supplier")
    private final List<Product> products = new ArrayList<>();

    public Supplier() {  }

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

    public Long getSupplierID() {
        return SupplierID;
    }

    public void setSupplierID(Long supplierID) {
        SupplierID = supplierID;
    }

    public String getCompanyName() {
        return CompanyName;
    }

    public void setCompanyName(String companyName) {
        CompanyName = companyName;
    }

    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 List<Product> getProducts() {
        return products;
    }
}
```

Klasa główna programu `Main`

```java
import model.Product;
import model.Supplier;
import org.hibernate.HibernateException;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
```

```java
import org.hibernate.Transaction;
import org.hibernate.cfg.Configuration;

import java.util.List;

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();
        Supplier supplier = new Supplier("Kraków HomoTrans", "Stefana Batorego 4", "Kraków");
        supplier.setSupplierID(0L);

        try {
        } finally {
            Transaction tx = session.beginTransaction();

            session.save(supplier);

            // Get all products from Product table,
            // add every available product new created supplier
            List<Product> products = session.createQuery("SELECT a FROM Product a", Product.class)
                .getResultList();
            for (Product p : products) {
                p.setSupplier(supplier);
                supplier.getProducts().add(p);
                session.save(p);
            }

            tx.commit();
            session.close();
        }
    }
}
```

	PRODUCTID	PRODUCTNAME	UNITSONSTOCK	SUPPLIE	SUPPLIERID	CITY	COMPANYNAME	STREET
1	0	Strawberries	2137	1	0	Kraków	Kraków HomoTrans	Stefana Batorego 4

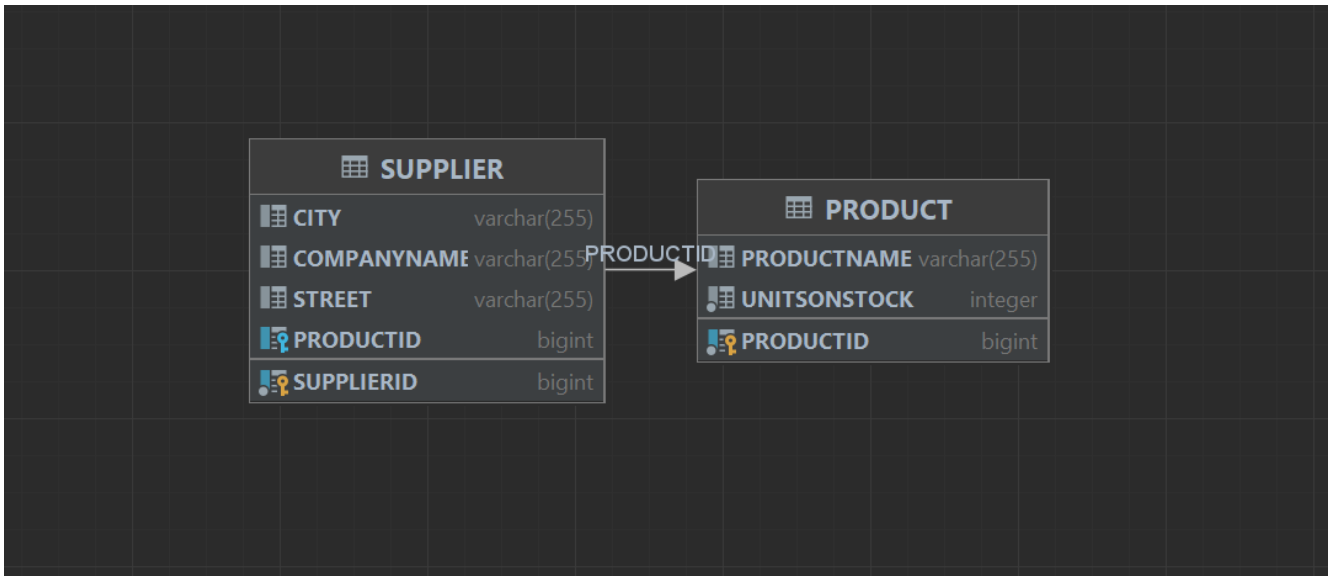
Rysunek 1: Zawartość tabel `Product` i `Supplier`.

3 Relacja `Supplier -> Product`

Odwrócenie relacji polega na odwróceniu odpowiednich typów w klasach `Product` i `Supplier`, tj. dodanie listy do pierwszej, pojedynczego pola obiektu do drugiej.

Klasa główna programu `Main`

```java
import model.Product;
import model.Supplier;
import org.hibernate.HibernateException;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.cfg.Configuration;

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

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();

        List<Product> products = new ArrayList<>();
        products.add(new Product("Bananas", 500));
        products.add(new Product("Pumpkins", 100));
        products.add(new Product("Coconuts", 3000));

        Supplier supplier = new Supplier("Kraków Speed", "Warszawska 33", "Kraków");

        for (Product product : products) {
            product.getSuppliers().add(supplier);
        }

        try {
        } finally {
            Transaction tx = session.beginTransaction();

            // Add to database new supplier and products
            session.save(supplier);
            for (Product product : products) {
                session.save(product);
            }


            tx.commit();
            session.close();
        }
    }
}
```



Rysunek 2: Diagram aktualnego stanu bazy danych.

	PRODUCTID	PRODUCTNAME	UNITSONSTOCK
1	1	Bananas	500
2	2	Pumpkins	100
3	3	Coconuts	3000

	SUPPLIERID	CITY	COMPANYNAME	STREET	PRODUCTID
1	1	Kraków	Kraków Speed	Warszawska 33	<null>

Rysunek 3: Zawartość tabel `Product` i `Supplier`.

4 Dwustronna relacja `Supplier <-> Product`

Aby uzyskać relację obustronną należy zapewnić, by obie klasy, `Product` i `Supplier`, posiadały listę i pojedynczy obiekt przeciwnej klasy.

Fragment klasy `Product`

```
    @ManyToOne
    @JoinColumn(name = "Supplier")
    private Supplier supplier;
    @OneToMany(mappedBy = "product")
    private final List<Supplier> suppliers = new ArrayList<>();
```

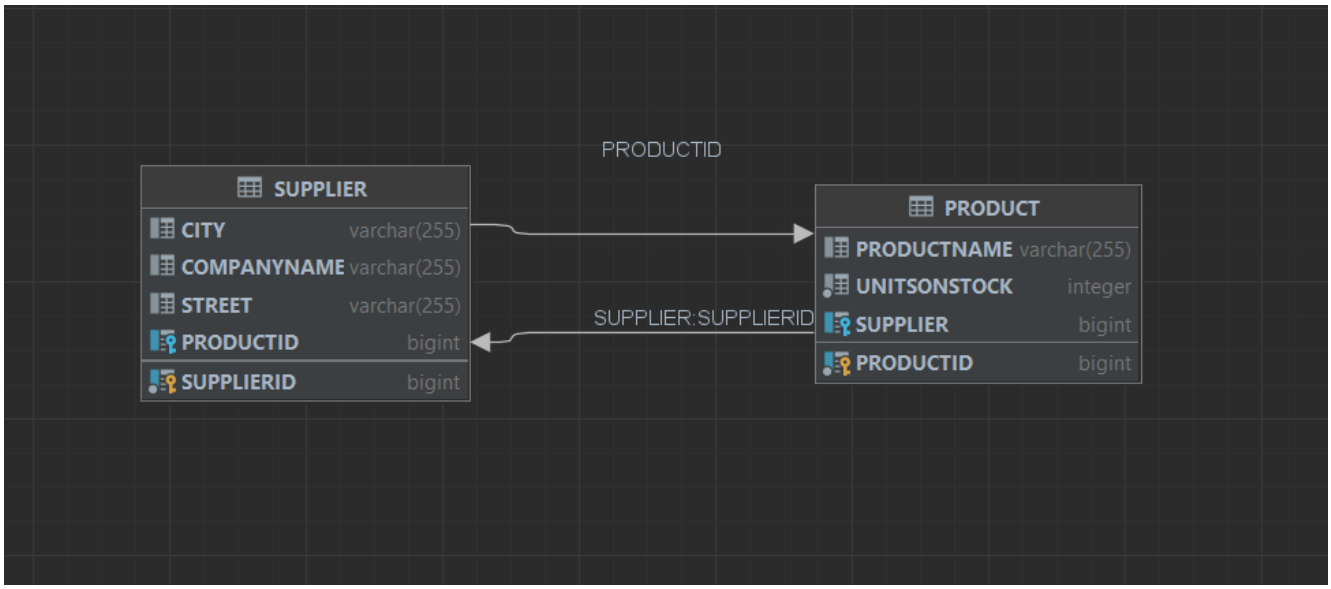
Fragment klasy `Supplier`

```
    @ManyToOne
    @JoinColumn(name = "ProductID")
    private Product product;
    @OneToMany(mappedBy = "supplier")
    private final List<Product> products = new ArrayList<>();
```

Klasa główna programu `Main`

```
    import model.Product;
    import model.Supplier;
    import org.hibernate.HibernateException;
    import org.hibernate.Session;
    import org.hibernate.SessionFactory;
    import org.hibernate.Transaction;
```

```java
import org.hibernate.cfg.Configuration;

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

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();

        List<Product> products = new ArrayList<>();
        products.add(new Product("Bananas", 500));
        products.add(new Product("Pumpkins", 100));
        products.add(new Product("Coconuts", 3000));

        Supplier supplier = new Supplier("Kraków Speed", "Warszawska 33", "Kraków");

        for (Product product : products) {
            product.getSuppliers().add(supplier);
            product.setSupplier(supplier);
            supplier.getProducts().add(product);
            supplier.setProduct(product);
        }

        try {
        } finally {
            Transaction tx = session.beginTransaction();

            // Add to database new supplier and products
            session.save(supplier);
            for (Product product : products) {
                session.save(product);
            }

            tx.commit();
            session.close();
        }
    }
}
```



Rysunek 4: Diagram aktualnego stanu bazy danych.

	PRODUCTID	PRODUCTNAME	UNITSONSTOCK	SUPPLIER
1	1	Bananas	500	1
2	2	Pumpkins	100	1
3	3	Coconuts	3000	1

	SUPPLIERID	CITY	COMPANYNAME	STREET	PRODUCTID
1	1	Kraków	Kraków Speed	Warszawska 33	3

Rysunek 5: Zawartość tabel `Product` i `Supplier`.

5 Klasa `Category`

Klasa `Category`

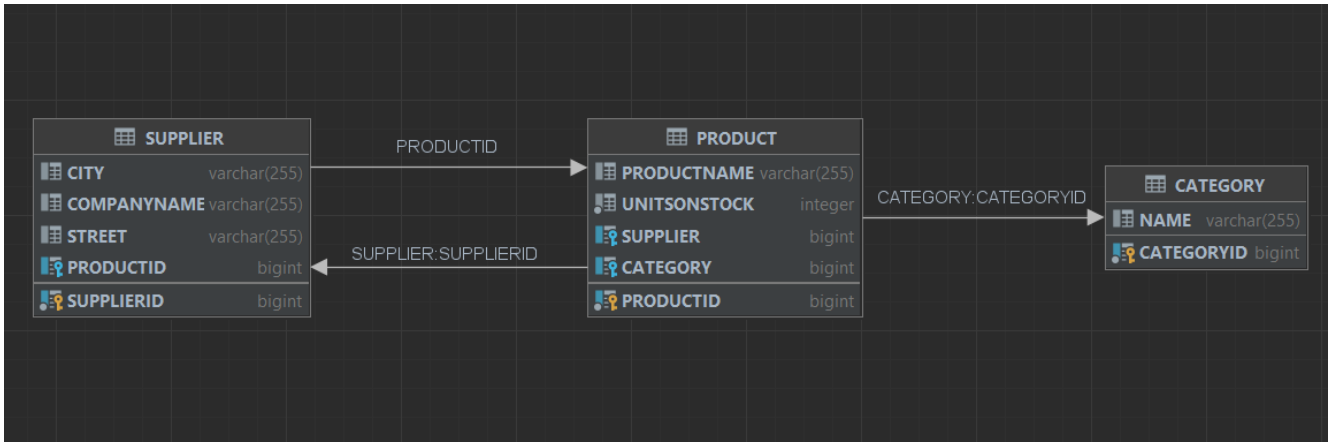
```java
package model;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import java.util.ArrayList;
import java.util.List;

@Entity
public class Category {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long categoryId;
    private String name;
    private final List<Product> products = new ArrayList<>();

    public Category() {  }

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

    public Long getCategoryId() {
        return categoryId;
    }
```

```
27
28          public void setCategoryId(Long categoryId) {
29              this.categoryId = categoryId;
30          }
31
32          public String getName() {
33              return name;
34          }
35
36          public void setName(String name) {
37              this.name = name;
38          }
39
40          public List<Product> getProducts() {
41              return products;
42          }
43      }
```

Klasa główna programu `Main`

```
1       import model.Category;
2       import model.Product;
3       import model.Supplier;
4       import org.hibernate.HibernateException;
5       import org.hibernate.Session;
6       import org.hibernate.SessionFactory;
7       import org.hibernate.Transaction;
8       import org.hibernate.cfg.Configuration;
9
10      import java.util.ArrayList;
11      import java.util.List;
12
13      public class Main {
14          private static final SessionFactory ourSessionFactory;
15
16          static {
17              try {
18                  Configuration configuration = new Configuration();
19                  configuration.configure();
20
21                  ourSessionFactory = configuration.buildSessionFactory();
22              } catch (Throwable ex) {
23                  throw new ExceptionInInitializerError(ex);
24              }
25          }
26
27          public static Session getSession() throws HibernateException {
28              return ourSessionFactory.openSession();
29          }
30
31          public static void main(final String[] args) throws Exception {
32              final Session session = getSession();
33
34              Supplier supplier = new Supplier("Kraków Speed", "Warszawska 33", "Kraków");
35
36              List<Category> categories = new ArrayList<>();
37              categories.add(new Category("Fruit"));
38              categories.add(new Category("Vegetable"));
39
40              Product product = new Product("Cucumbers", 10);
41              product.setCategory(categories.get(1));
42
43              try {
```

```
44              } finally {
45                  Transaction tx = session.beginTransaction();
46
47                  session.save(product);
48
49                  for (Category category : categories) {
50                      session.save(category);
51                  }
52
53                  product = session.load(Product.class, 1L);
54                  product.setCategory(categories.get(0));
55                  session.save(product);
56
57                  product = session.load(Product.class, 2L);
58                  product.setCategory(categories.get(0));
59                  session.save(product);
60
61                  product = session.load(Product.class, 3L);
62                  product.setCategory(categories.get(0));
63                  session.save(product);
64
65                  session.save(supplier);
66
67                  tx.commit();
68                  session.close();
69              }
70          }
71      }
```



Rysunek 6: Diagram aktualnego stanu bazy danych.

	PRODUCTID	PRODUCTNAME	UNITSONSTOCK	SUPPLIER	CATEGORY
1	1	Bananas	500	1	1
2	2	Pumpkins	100	1	1
3	3	Coconuts	3000	1	1
4	7	Cucumbers	10	<null>	2

	CATEGORYID	NAME
1	1	Fruit
2	2	Vegetable

Rysunek 7: Zawartość tabel `Product` i `Category`.

6 Dwustronna relacja `Invoice <-> Product`

Klasa `Invoice`

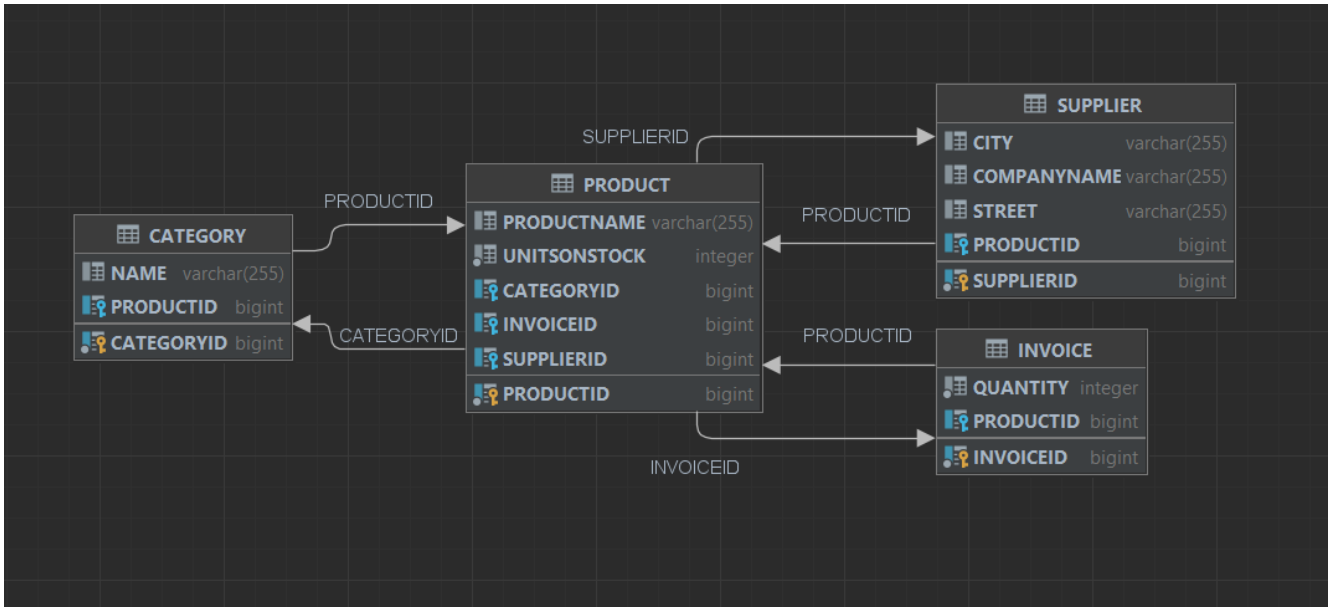
```java
package model;

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

@Entity
public class Invoice {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long InvoiceID;
    private int quantity;
    @ManyToOne
    @JoinColumn(name = "ProductID")
    private Product product;
    @OneToMany(mappedBy = "invoice")
    private final List<Product> products = new ArrayList<>();

    public Invoice() {  }

    public Invoice(int quantity) {
        this.quantity = quantity;
    }

    public Long getInvoiceID() {
        return InvoiceID;
    }

    public void setInvoiceID(Long invoiceID) {
        InvoiceID = invoiceID;
    }

    public int getQuantity() {
        return quantity;
    }

    public void setQuantity(int quantity) {
        this.quantity = quantity;
    }

    public Product getProduct() {
        return product;
    }

    public void setProduct(Product product) {
        this.product = product;
    }

    public List<Product> getProducts() {
        return products;
    }
}
```

Klasa główna programu `Main`

```java
import model.Category;
import model.Invoice;
import model.Product;
import model.Supplier;
import org.hibernate.HibernateException;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
```

```
9   import org.hibernate.cfg.Configuration;
10
11  import java.util.ArrayList;
12  import java.util.List;
13
14  public class Main {
15      private static final SessionFactory ourSessionFactory;
16
17      static {
18          try {
19              Configuration configuration = new Configuration();
20              configuration.configure();
21
22              ourSessionFactory = configuration.buildSessionFactory();
23          } catch (Throwable ex) {
24              throw new ExceptionInInitializerError(ex);
25          }
26      }
27
28      public static Session getSession() throws HibernateException {
29          return ourSessionFactory.openSession();
30      }
31
32      public static void main(final String[] args) throws Exception {
33          final Session session = getSession();
34
35          List<Product> products = new ArrayList<>();
36          products.add(new Product("Bananas", 500));
37          products.add(new Product("Pumpkins", 100));
38          products.add(new Product("Coconuts", 3000));
39
40          Supplier supplier = new Supplier("Kraków Speed", "Warszawska 33", "Kraków");
41
42          List<Category> categories = new ArrayList<>();
43          categories.add(new Category("Fruit"));
44          categories.add(new Category("Vegetable"));
45          for (Category category : categories) {
46              category.setProduct(products.get(0));
47          }
48
49          for (Product product : products) {
50              product.getSuppliers().add(supplier);
51              product.setSupplier(supplier);
52              product.setCategory(categories.get(0));
53              supplier.getProducts().add(product);
54              supplier.setProduct(product);
55          }
56
57          List<Invoice> invoices = new ArrayList<>();
58          invoices.add(new Invoice(1));
59          invoices.add(new Invoice(5));
60
61          invoices.get(0).setProduct(products.get(0));
62          invoices.get(0).getProducts().addAll(products);
63          invoices.get(1).setProduct(products.get(2));
64          invoices.get(1).getProducts().add(products.get(2));
65
66          try {
67          } finally {
68              Transaction tx = session.beginTransaction();
69
70              // Add to database new records
```

```
                session.save(supplier);
                for (Product product : products) {
                    session.save(product);
                }
                for (Category category : categories) {
                    session.save(category);
                }
                for (Invoice invoice : invoices) {
                    session.save(invoice);
                }

                tx.commit();
                session.close();
            }
        }
    }
```



Rysunek 8: Diagram aktualnego stanu bazy danych.

Zapytanie pokazujące wszystkie produkty z faktury o `ID` równym 2.

```
SELECT P.* FROM PRODUCT P
INNER JOIN INVOICE I on I.PRODUCTID = P.PRODUCTID
WHERE I.INVOICEID = 2;
```

	PRODUCTID	PRODUCTNAME	UNITSONSTOCK	CATEGORYID	INVOICEID	SUPPLIERID
1	3	Coconuts	3000	1	<null>	1

Rysunek 9: Wynik zapytania.

Zapytanie pokazujące wszystkie faktury, które zawierają produkt o `ID` równym 1.

```
SELECT I.* FROM INVOICE I
INNER JOIN PRODUCT P on I.PRODUCTID = P.PRODUCTID
WHERE I.PRODUCTID = 1;
```

	INVOICEID	QUANTITY	PRODUCTID
1	1	1	1

Rysunek 10: Wynik zapytania.

7 Wykorzystanie JPA

Nowo utworzona klasa główna programu `Main`

```
import model.Category;
import model.Invoice;
import model.Product;
import model.Supplier;
import org.hibernate.HibernateException;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.cfg.Configuration;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;
import java.util.ArrayList;
import java.util.List;

public class MainJpa {
    public static void main(final String[] args) throws Exception {
        EntityManagerFactory emFactory = Persistence.createEntityManagerFactory("JakubSzaredkoJPA");
        EntityManager entityManager = emFactory.createEntityManager();

        entityManager.getTransaction().begin();

        List<Product> products = new ArrayList<>();
        products.add(new Product("Bananas", 500));
        products.add(new Product("Pumpkins", 100));
        products.add(new Product("Coconuts", 3000));

        Supplier supplier = new Supplier("Kraków Speed", "Warszawska 33", "Kraków");

        List<Category> categories = new ArrayList<>();
        categories.add(new Category("Fruit"));
        categories.add(new Category("Vegetable"));
        for (Category category : categories) {
            category.setProduct(products.get(0));
        }

        for (Product product : products) {
            product.getSuppliers().add(supplier);
            product.setSupplier(supplier);
            product.setCategory(categories.get(0));
            supplier.getProducts().add(product);
            supplier.setProduct(product);
        }

        List<Invoice> invoices = new ArrayList<>();
        invoices.add(new Invoice(1));
```

```
48              invoices.add(new Invoice(5));
49
50              invoices.get(0).setProduct(products.get(0));
51              invoices.get(0).getProducts().addAll(products);
52              invoices.get(1).setProduct(products.get(2));
53              invoices.get(1).getProducts().add(products.get(2));
54
55              // Add to database new records
56              entityManager.persist(supplier);
57              for (Product product : products) {
58                  entityManager.persist(product);
59              }
60              for (Category category : categories) {
61                  entityManager.persist(category);
62              }
63              for (Invoice invoice : invoices) {
64                  entityManager.persist(invoice);
65              }
66
67              entityManager.getTransaction().commit();
68              entityManager.close();
69              emFactory.close();
70          }
71      }
```

Plik konfiguracyjny `persistence.xml`

```
1   <?xml version="1.0"?>
2   <persistence xmlns="http://java.sun.com/xml/ns/persistence"
3                xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4                xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
5   http://java.sun.com/xml/ns/persistence/persistence_2_0.xsd"
6                version="2.0">
7     <persistence-unit name="JakubSzaredkoJPA"
8                       transaction-type="RESOURCE_LOCAL">
9       <properties>
10        <property name="hibernate.connection.driver_class"
11                  value="org.apache.derby.jdbc.ClientDriver"/>
12        <property name="hibernate.connection.url"
13                  value="jdbc:derby://127.0.0.1/JakubSzaredkoJPA"/>
14        <property name="hibernate.show_sql" value="true" />
15        <property name="hibernate.format_sql" value="true" />
16        <property name="hibernate.hbm2ddl.auto" value="create" />
17      </properties>
18    </persistence-unit>
19  </persistence>
```

8 Utworzenie kaskadowej relacji

Do odpowiednich list występujących w klasach `Product` i `Invoice` dodałem odpowiednie opcje `cascade` do adnotacji `@OneToMany`.
Fragment klasy `Product`

```
1   @OneToMany(mappedBy = "product", cascade = CascadeType.REFRESH)
2   private final List<Invoice> invoices = new ArrayList<>();
```

Fragment klasy `Invoice`

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

9 Klasa Embedded

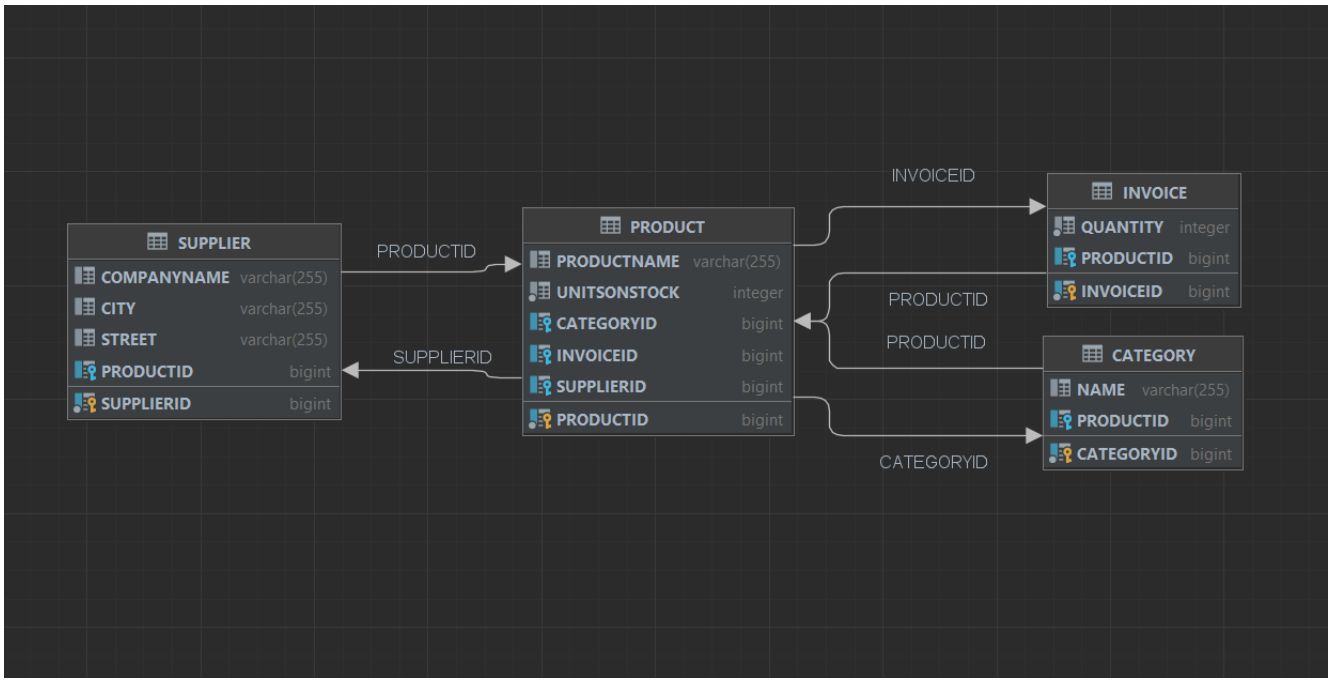
Klasa `Address`

```
package model;

import javax.persistence.Embeddable;

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

    public Address() {  }

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

    public String getStreet() {
        return street;
    }

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

    public String getCity() {
        return city;
    }

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

Klasa `Supplier`

```
package model;

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

@Entity
public class Supplier {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long SupplierID;
    private String CompanyName;
    @Embedded
    private Address address;
    @ManyToOne
    @JoinColumn(name = "ProductID")
    private Product product;
```

```
18          @OneToMany(mappedBy = "supplier")
19          private final List<Product> products = new ArrayList<>();
20
21          public Supplier() {  }
22
23          public Supplier(String companyName, String street, String city) {
24              CompanyName = companyName;
25              address = new Address(street, city);
26          }
27
28          public Long getSupplierID() {
29              return SupplierID;
30          }
31
32          public void setSupplierID(Long supplierID) {
33              SupplierID = supplierID;
34          }
35
36          public String getCompanyName() {
37              return CompanyName;
38          }
39
40          public void setCompanyName(String companyName) {
41              CompanyName = companyName;
42          }
43
44          public Address getAddress() {
45              return address;
46          }
47
48          public void setAddress(Address address) {
49              this.address = address;
50          }
51
52          public Product getProduct() {
53              return product;
54          }
55
56          public void setProduct(Product product) {
57              this.product = product;
58          }
59
60          public List<Product> getProducts() {
61              return products;
62          }
63      }
```



Rysunek 11: Diagram aktualnego stanu bazy danych.

Zmapowanie klasy Supplier z klasą `Address`, tak aby powstały dwie oddzielne tabele, możemy uzyskać za pomocą relacji `OneToOne` lub `ManyToOne`.

10 Dziedziczenie

10.1 `@MappedSuperclass`

Klasa `Company`

```
1   package model;
2
3   import javax.persistence.*;
4
5   @MappedSuperclass
6   public class Company {
7       @Id
8       @GeneratedValue(strategy = GenerationType.IDENTITY)
9       private Long CompanyID;
10      private String companyName;
11      @Embedded
12      private Address address;
13
14      public Company() {  }
15      public Company(String companyName, String street, String city, String zipCode) {
16          this.companyName = companyName;
17          address = new Address(street, city, zipCode);
18      }
19
20      public Long getCompanyID() {
21          return CompanyID;
22      }
23
24      public void setCompanyID(Long companyID) {
25          CompanyID = companyID;
```

```
26          }
27
28          public String getCompanyName() {
29              return companyName;
30          }
31
32          public void setCompanyName(String companyName) {
33              this.companyName = companyName;
34          }
35
36          public Address getAddress() {
37              return address;
38          }
39
40          public void setAddress(Address address) {
41              this.address = address;
42          }
43      }
```

Klasa `Supplier`

```
1       package model;
2
3       import javax.persistence.*;
4       import java.util.ArrayList;
5       import java.util.List;
6
7       @Entity
8       public class Supplier extends Company {
9           private String bankAccountNumber;
10          @ManyToOne
11          @JoinColumn(name = "ProductID")
12          private Product product;
13          @OneToMany(mappedBy = "supplier")
14          private final List<Product> products = new ArrayList<>();
15
16          public Supplier() {  }
17
18          public Supplier(String companyName, String street, String city, String zipCode, String bankAccountNumber) {
19              super(companyName, street, city, zipCode);
20              this.bankAccountNumber = bankAccountNumber;
21          }
22
23          public String getBankAccountNumber() {
24              return bankAccountNumber;
25          }
26
27          public void setBankAccountNumber(String bankAccountNumber) {
28              this.bankAccountNumber = bankAccountNumber;
29          }
30
31          public Product getProduct() {
32              return product;
33          }
34
35          public void setProduct(Product product) {
36              this.product = product;
37          }
38
39          public List<Product> getProducts() {
40              return products;
41          }
42      }
```

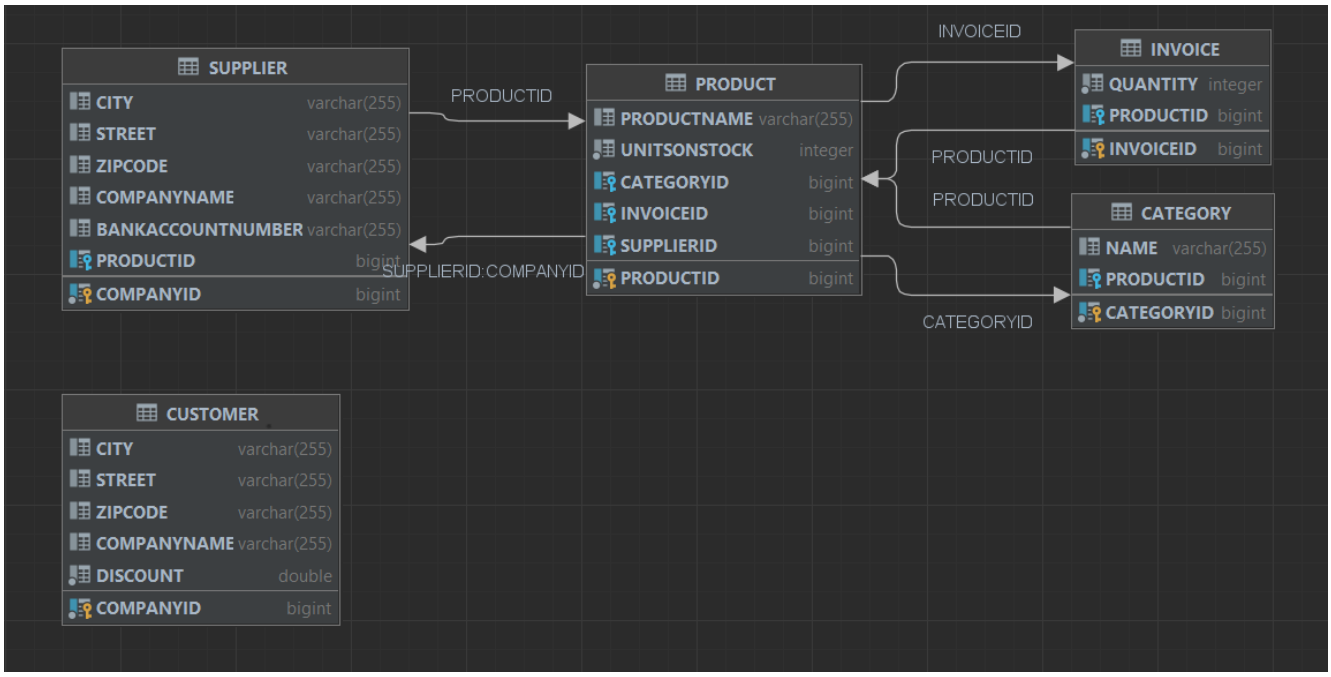
Klasa `Customer`

```java
package model;

import javax.persistence.Entity;

@Entity
public class Customer extends Company {
    private float discount;

    public Customer() {  }

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

    public float getDiscount() {
        return discount;
    }

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

Klasa główna programu `Main`

```java
import model.*;
import org.hibernate.HibernateException;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.cfg.Configuration;

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

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();

        List<Product> products = new ArrayList<>();
        products.add(new Product("Bananas", 500));
        products.add(new Product("Pumpkins", 100));
        products.add(new Product("Coconuts", 3000));
```

```
36
37              Supplier supplier = new Supplier(
38                  "Kraków Speed",
39                  "Warszawska 33",
40                  "Kraków",
41                  "02-137",
42                  "002137420"
43              );
44              Customer customer = new Customer(
45                  "Mszana Dolna Lubogoszcz",
46                  "Spadochroniarzy 3",
47                  "Mszana Dolna",
48                  "34-730",
49                  .1f
50              );
51
52              List<Category> categories = new ArrayList<>();
53              categories.add(new Category("Fruit"));
54              categories.add(new Category("Vegetable"));
55              for (Category category : categories) {
56                  category.setProduct(products.get(0));
57              }
58
59              for (Product product : products) {
60                  product.getSuppliers().add(supplier);
61                  product.setSupplier(supplier);
62                  product.setCategory(categories.get(0));
63                  supplier.getProducts().add(product);
64                  supplier.setProduct(product);
65              }
66
67              List<Invoice> invoices = new ArrayList<>();
68              invoices.add(new Invoice(1));
69              invoices.add(new Invoice(5));
70
71              invoices.get(0).setProduct(products.get(0));
72              invoices.get(0).getProducts().addAll(products);
73              invoices.get(1).setProduct(products.get(2));
74              invoices.get(1).getProducts().add(products.get(2));
75
76              try {
77              } finally {
78                  Transaction tx = session.beginTransaction();
79
80                  // Add to database new records
81                  session.save(supplier);
82                  session.save(customer);
83                  for (Product product : products) {
84                      session.save(product);
85                  }
86                  for (Category category : categories) {
87                      session.save(category);
88                  }
89                  for (Invoice invoice : invoices) {
90                      session.save(invoice);
91                  }
92
93                  tx.commit();
94                  session.close();
95              }
96          }
97      }
```

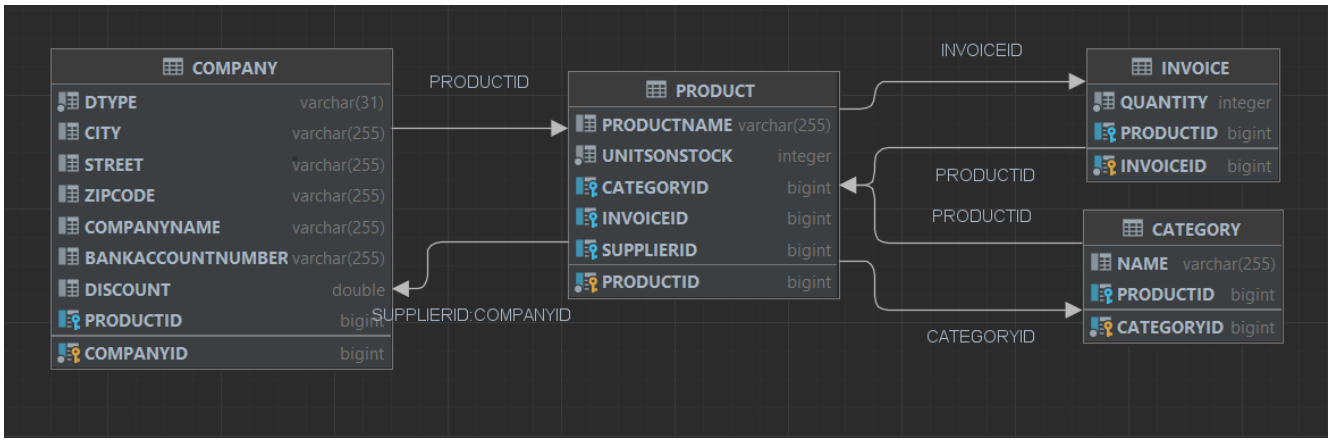


Rysunek 12: Diagram aktualnego stanu bazy danych.

10.2 `@InheritanceType.SINGLE_TABLE`

Fragment klasy `Company`

```
1  package model;
2
3  import javax.persistence.*;
4
5  @Entity
6  @Inheritance(strategy = InheritanceType.SINGLE_TABLE)
7  public class Company {
8      // ...
9  }
```

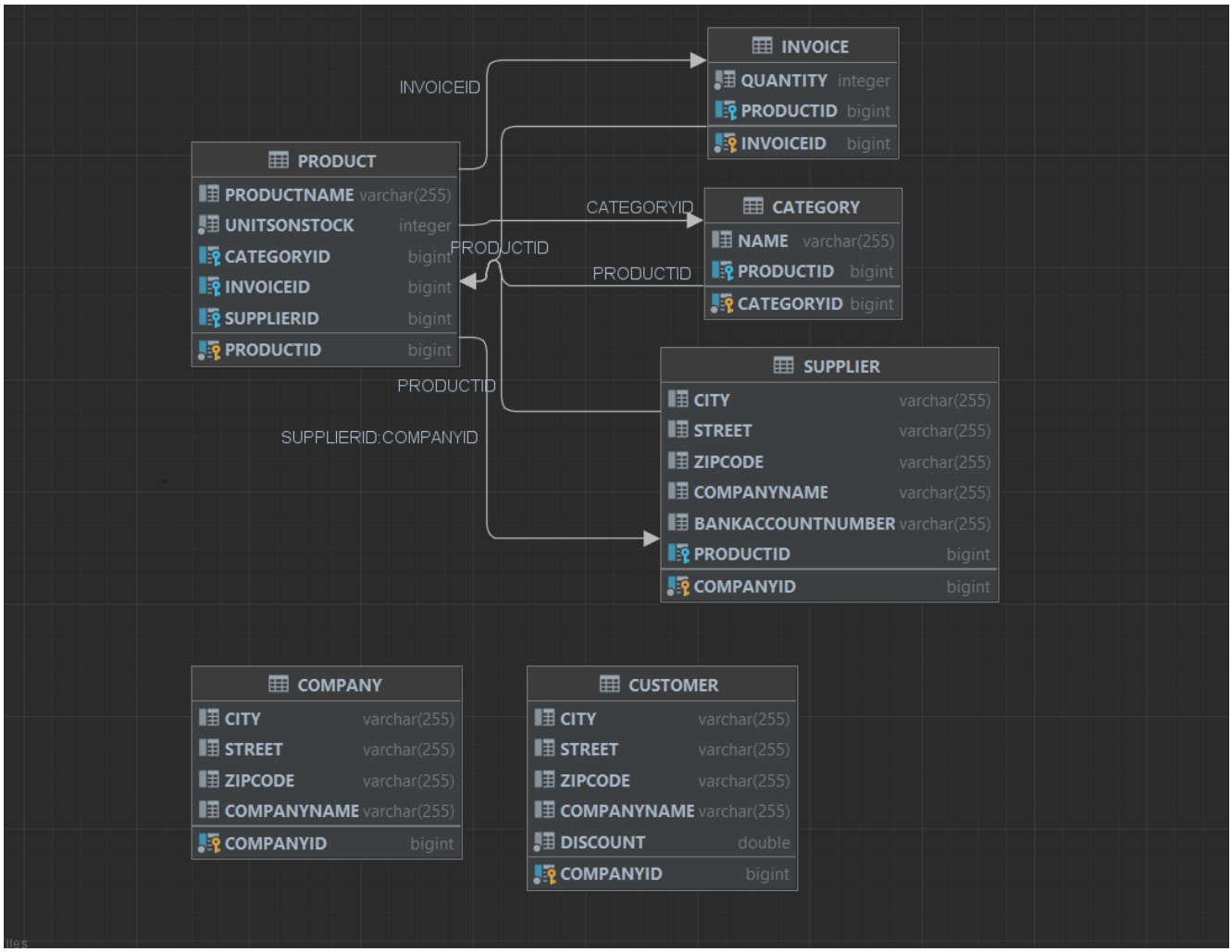


Rysunek 13: Diagram aktualnego stanu bazy danych.

10.3 `@InheritanceType.TABLE_PER_CLASS`

Fragment klasy `Company`

```java
package model;

import javax.persistence.*;

@Entity
@Inheritance(strategy = InheritanceType.TABLE_PER_CLASS)
public class Company {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    // ...
}
```



Rysunek 14: Diagram aktualnego stanu bazy danych.