I. Basics

a. Uruchomienie servera Derby

```
C:\Users\Krzys\Desktop\hibernate\db-derby-10.15.2.0-bin\db-derby-10.15.2.0-bin\bin>startNetworkServer.bat
Thu May 07 11:37:42 CEST 2020 : Security manager installed using the Basic server security policy.
Thu May 07 11:37:42 CEST 2020 : Serwer sieciowy Apache Derby - 10.15.2.0 - (1873585) uruchomiony i gotowy do zaakceptowa
nia po╣cze˜ na porcie 1527 w {3}
```

b. Uruchom konsole ij

```
C:\Users\Krzys\Desktop\hibernate\db-derby-10.15.2.0-bin\db-derby-10.15.2.0-bin\bin>ij
wersja ij 10.15
ij>
```

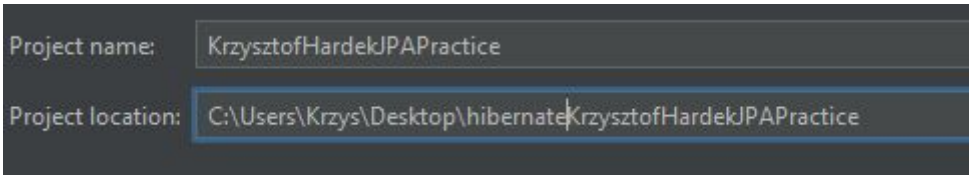
c. Podepnij się do serwera zakladajac baze INazwiskoJPA

```
C:\Users\Krzys\Desktop\hibernate\db-derby-10.15.2.0-bin\db-derby-10.15.2.0-bin\bin>ij
wersja ij 10.15
ij> connect 'jdbc:derby://127.0.0.1/KrzysztofHardekJPA;create=true'
```

d. Polecenie show tables

```
ij> show tables;
TABLE_SCHEM         |TABLE_NAME                     |REMARKS
------------------------------------------------------------------------------
SYS                 |SYSALIASES                     |
SYS                 |SYSCHECKS                      |
SYS                 |SYSCOLPERMS                    |
SYS                 |SYSCOLUMNS                     |
SYS                 |SYSCONGLOMERATES               |
SYS                 |SYSCONSTRAINTS                 |
SYS                 |SYSDEPENDS                     |
SYS                 |SYSFILES                       |
SYS                 |SYSFOREIGNKEYS                 |
SYS                 |SYSKEYS                        |
SYS                 |SYSPERMS                       |
SYS                 |SYSROLES                       |
SYS                 |SYSROUTINEPERMS                |
SYS                 |SYSSCHEMAS                     |
SYS                 |SYSSEQUENCES                   |
SYS                 |SYSSTATEMENTS                  |
SYS                 |SYSSTATISTICS                  |
SYS                 |SYSTABLEPERMS                  |
SYS                 |SYSTABLES                      |
SYS                 |SYSTRIGGERS                    |
SYS                 |SYSUSERS                       |
SYS                 |SYSVIEWS                       |
SYSIBM              |SYSDUMMY1                      |

23 wierszy wybranych
```

e. Stwórz projekt



f. Stwórz klase produktu z polami ProductName, UnitsOnStock

```
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)
  private int dbID;

  private String productName;
  private int unitsOnStock;

  public Product(){}

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

  public String getProductName() {
     return this.productName;
  }

  public int getUnitsOnStock() {
     return this.unitsOnStock;
  }

  public void setUnitsOnStock(int unitsOnStock) {
     this.unitsOnStock = unitsOnStock;
  }

  public void setProductName(String productName) {
     this.productName = productName;
  }
```

```
  public int getDbID() {
    return dbID;
  }

}
```

g. Uzupełnij potrzebne property w konfiguracji hibernate’a

```
<?xml version='1.0' encoding='utf-8'?>
      <!DOCTYPE hibernate-configuration PUBLIC
            "-//Hibernate/Hibernate Configuration DTD//EN"
            "http://www.hibernate.org/dtd/hibernate-configuration3.0.dtd">
<hibernate-configuration>
  <session-factory>
      <property
name="connection.url">jdbc:derby://127.0.0.1/KrzysztofHardekJPA</property>
      <property
name="connection.driver_class">org.apache.derby.jdbc.ClientDriver</property>
      <property name="dialect">org.hibernate.dialect.DerbyTenSevenDialect</property>
      <property name="format_sql">true</property>
      <property name="show_sql">true</property>
      <property name="use_sql_comments">true</property>
      <!-- DB schema will be updated if needed -->
      <property name="hibernate.hbm2ddl.auto">update</property>
      <mapping class="Product"></mapping>
  </session-factory>
</hibernate-configuration
```

h. Stwórz w mainie przykładowy produkt

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

public class HibRunner {
  private static SessionFactory sessionFactory = null;
  public static void main(String[] args) {
      Product product = new Product("Krokiet",
            10);
      sessionFactory = getSessionFactory();
      Session session = sessionFactory.openSession();
      Transaction tx = session.beginTransaction();
      session.save(product);
      tx.commit();
```

```
        session.close();
    }
    private static SessionFactory getSessionFactory() {
        if (sessionFactory == null) {
            Configuration configuration = new Configuration();
            sessionFactory =
                    configuration.configure().buildSessionFactory();
        }
        return sessionFactory;
    }
}
```

Efekt:

```
Hibernate:

    create table Product (
       dbID integer not null,
        productName varchar(255),
        unitsOnStock integer not null,
        primary key (dbID)
    )
Hibernate: create sequence hibernate_sequence start with 1 increment by 1
```

```
Hibernate:

values
    next value for hibernate_sequence
Hibernate:
    /* insert Product
        */ insert
        into
            Product
            (productName, unitsOnStock, dbID)
        values
            (?, ?, ?)
```

i. Widok z datagrip

Schemat bazy:

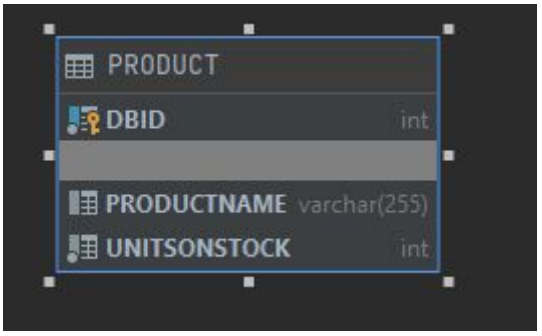


Tabela Products:

	DBID	PRODUCTNAME	UNITSONSTOCK
1	1	Krokiet	10

II. Wprowadzenie dostawcy

a. Stworzenie nowej klasy

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

  private String companyName;
  private String street;
  private String city;

  public Supplier(){}

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

  public int getDbID() {
     return dbID;
  }

  public String getCity() {
     return city;
  }

  public String getCompanyName() {
     return companyName;
  }
```

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

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

    public void setCompanyName(String companyName) {
        this.companyName = companyName;
    }

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

b. Dodanie pola supplier oraz jego setera i getera w Product

```
@ManyToOne
private Supplier supplier;

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

public Supplier getSupplier() {
    return supplier;
}
```

c. Dodanie dostawcy i ustawienie go w produkcie

```
public class HibRunner {
    private static SessionFactory sessionFactory = null;
    public static void main(String[] args) {
        Supplier supplier = new Supplier("Google", "Bema", "Oswiecim");
        sessionFactory = getSessionFactory();
        Session session = sessionFactory.openSession();
        Transaction tx = session.beginTransaction();

        session.save(supplier);
        Product foundProduct = session.get(Product.class, 1);
        foundProduct.setSupplier(supplier);
        tx.commit();
        session.close();
    }
```

Efekt:

```
Hibernate:
    /* insert Supplier
        */ insert
        into
            Supplier
            (city, companyName, street, dbID)
        values
            (?, ?, ?, ?)
Hibernate:
    /* update
        Product */ update
            Product
        set
            productName=?,
            supplier_dbID=?,
            unitsOnStock=?
        where
            dbID=?
```

d. Widok z datagrip

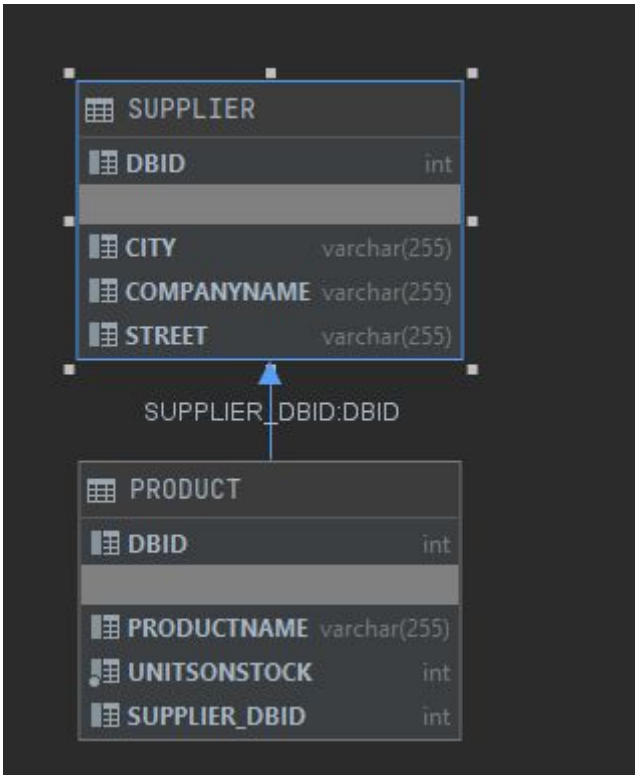
Tabela supplier:

	DBID	CITY	COMPANYNAME	STREET
1	2	Oswiecim	Google	Bema

Tabela products:

	DBID	PRODUCTNAME	UNITSONSTOCK	SUPPLIER_DBID
1	1	Krokiet	10	2

Schemat bazy:



III. Odwrócenie relacji

a. Z tabelą łącznikową

Dodałem pole oraz geter i seter w klasie Supplier oraz zakomentowałem poprzednio dodane linijki i nadpisałem metodę compareTo w klasie Product:

Klasa Supplier:

```
private Set<Product> productSet;

public Set<Product> getProductSet() {
  return productSet;
}

public void setProductSet(Set<Product> productSet) {
  this.productSet = productSet;
}
```

Klasa Product:

```
@Override
public int compareTo(Object o) {
  if(o instanceof Product){
```

```
        return ((Product) o).getProductName().compareTo(this.productName);
    }

    return -1;
}
```

Main:

```
public static void main(String[] args) {
    Product product1 = new Product("Ogorek", 3);
    Product product2 = new Product("Zupa", 4);
    Product product3 = new Product("Pies", 5);
    Set<Product> productSet = new TreeSet<>();
    productSet.add(product1);
    productSet.add(product2);
    productSet.add(product3);
    Supplier supplier = new Supplier("Facebook", "Borowego", "Krakow");
    supplier.setProductSet(productSet);

    sessionFactory = getSessionFactory();
    Session session = sessionFactory.openSession();
    Transaction tx = session.beginTransaction();


    session.save(supplier);
    session.save(product1);
    session.save(product2);
    session.save(product3);
    tx.commit();
    session.close();
}
```

efekt:

```
Hibernate:

    create table Supplier_Product (
       Supplier_dbID integer not null,
        productSet_dbID integer not null,
        primary key (Supplier_dbID, productSet_dbID)
    )
Hibernate:

    alter table Supplier_Product
       drop constraint UK_jl61xkmi6tf7rq7bq24s623ru
Hibernate:

    alter table Supplier_Product
       add constraint UK_jl61xkmi6tf7rq7bq24s623ru unique (productSet_dbID)
Hibernate:

    alter table Supplier_Product
       add constraint FKbuapa41oers6q19syovqvnumc
       foreign key (productSet_dbID)
       references Product
Hibernate:

    alter table Supplier_Product
       add constraint FKoug7pvmk6ld50lhl9u4bs1x1v
       foreign key (Supplier_dbID)
       references Supplier
```

b. Widok z datagrip:

Tabela Product:

	DBID	PRODUCTNAME	UNITSONSTOCK
1	107	Ogorek	3
2	108	Zupa	4
3	109	Pies	5

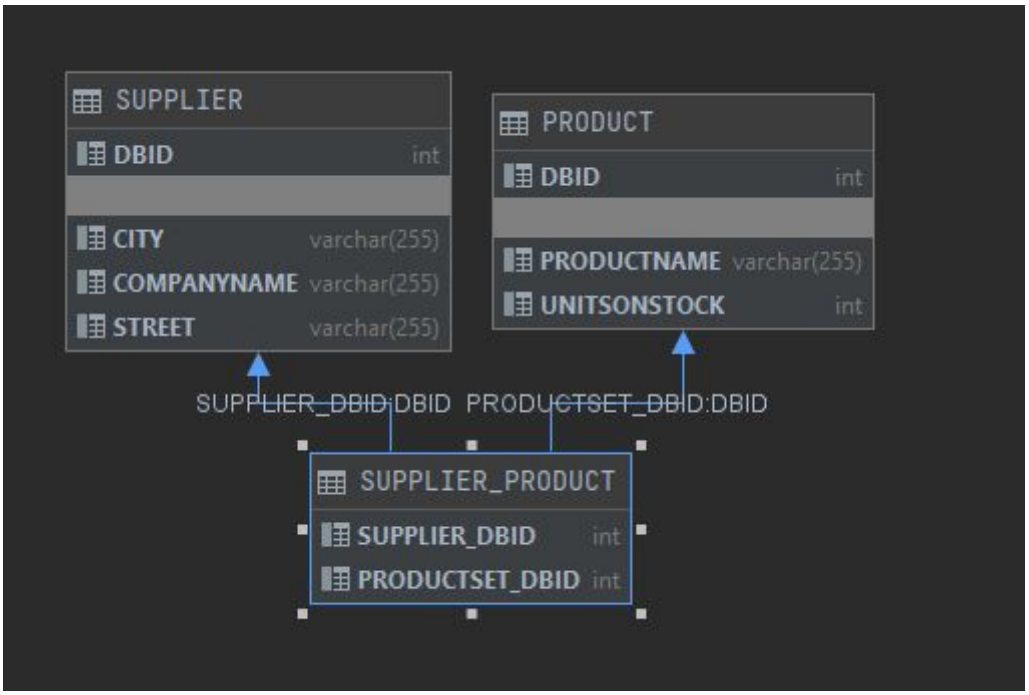
Tabela Supplier:

	DBID	CITY	COMPANYNAME	STREET
1	106	Krakow	Facebook	Borowego

Tabela Product_Supplier:

	SUPPLIER_DBID	PRODUCTSET_DBID
1	106	107
2	106	108
3	106	109

Diagram bazy:



c. Bez tabeli łącznikowej

Dopisałem jedną linijkę w klasie Supplier likwidującą dodatkową tabelę

```
@OneToMany
@JoinColumn(name="SUPPLIER_FK")
private Set<Product> productSet;
```

d. Widok z datagrip

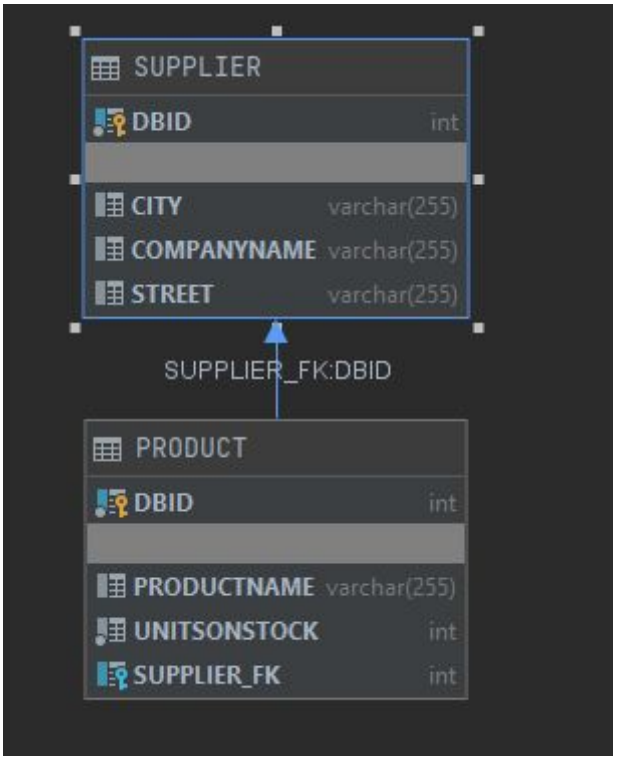
Tabela Product:

	DBID	PRODUCTNAME	UNITSONSTOCK	SUPPLIER_FK
1	111	Ogorek	3	110
2	112	Zupa	4	110
3	113	Pies	5	110

Tabela Supplier:

	DBID	CITY	COMPANYNAME	STREET
1	110	Krakow	Facebook	Borowego

Diagram bazy:



IV. Relacja dwustronna

a. Modyfikacja adnotacji

Klasa Product:

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

Klasa Supplier:

```
@OneToMany(mappedBy = "supplier")
private Set<Product> productSet;
```

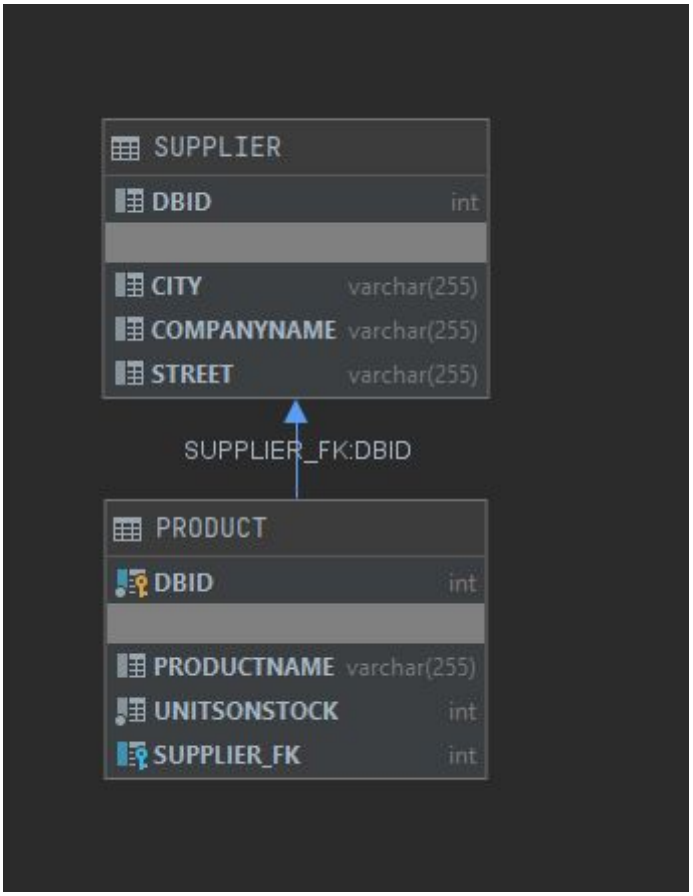
Tabela Supplier:

	DBID	CITY	COMPANYNAME	STREET
1	134	Krakow	Facebook	Borowego

Tabela Product:

	DBID	PRODUCTNAME	UNITSONSTOCK	SUPPLIER_FK
1	135	Ogorek	3	134
2	136	Zupa	4	134
3	137	Pies	5	134

Diagram bazy:



VI. Klasa Category

a. Stworzenie klasy

```
@Entity
public class Category {
  @Id
```

```
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int categoryID;

    private String name;

    @OneToMany
    @JoinColumn(name = "CATEGORY_FK")
    private List<Product> productList;

    public Category(){}

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

    public int getCategoryID() {
        return categoryID;
    }

    public List<Product> getProductList() {
        return productList;
    }

    public void setName(String name) {
        this.name = name;
    }

    public void setProductList(List<Product> productList) {
        this.productList = productList;
    }

    public String getName() {
        return name;
    }
}
```

b. Dodanie kategorii, nowych produktów oraz przypisanie kategorii starym

```
public class HibRunner {
    private static SessionFactory sessionFactory = null;
    public static void main(String[] args) {
        Category category1 = new Category("Jedzenie");
        Category category2 = new Category("Kosmetyki");

        Product product1 = new Product("Szminka", 6);
        Product product2 = new Product("Chusteczki", 7);
        Product product3 = new Product("Burak", 8);
```

```
        Set<Product> productSet = new TreeSet<>();
        productSet.add(product1);
        productSet.add(product2);
        productSet.add(product3);

        Supplier supplier = new Supplier("Biedronka", "Rynek", "Wrocław");

        product1.setSupplier(supplier);
        product2.setSupplier(supplier);
        product3.setSupplier(supplier);
        supplier.setProductSet(productSet);

        List<Product> productList = new ArrayList<>();
        productList.add(product1);
        productList.add(product2);
        productList.add(product3);
        category2.setProductList(productList);

        sessionFactory = getSessionFactory();
        Session session = sessionFactory.openSession();
        Transaction tx = session.beginTransaction();

        productList = new ArrayList<>();
        for(int i = 135; i < 138; i++){
            Product product = session.get(Product.class, i);
            productList.add(product);
        }

        category1.setProductList(productList);

        session.save(supplier);
        session.save(product1);
        session.save(product2);
        session.save(product3);
        session.save(category1);
        session.save(category2);
        tx.commit();
        session.close();
    }
```

c. Wydobądź produkty z wybranej kategorii oraz kategorię do której należy wybrany produkt

Usunąłem wyświetlanie SQLa w configu aby pokazać efekt w konsola:

```
<property name="show_sql">false</property>
```

```java
public class HibRunner {
    private static SessionFactory sessionFactory = null;
    public static void main(String[] args) {
        sessionFactory = getSessionFactory();
        Session session = sessionFactory.openSession();
        Transaction tx = session.beginTransaction();

        String hql = "FROM Category";
        Query query = session.createQuery(hql);
        List<Category> categories = (List<Category>) query.list();

        for(Category cat : categories){
            System.out.printf("Kategoria: %s\n", cat.getName());
            for(Product prod : cat.getProductList()){
                System.out.printf("Produkt: %s\n", prod.getProductName());
            }
            System.out.println("");
        }
        System.out.println("------------------");

        hql = "FROM Product";
        query = session.createQuery(hql);
        List<Product> results = (List<Product>) query.list();

        for(Product prod :  results){
            System.out.printf("Produkt: %s\n", prod.getProductName());
            for(Category cat : categories){
                if(cat.getProductList().contains(prod)){
                    System.out.printf("Kategoria: %s\n", cat.getName());
                }
            }
            System.out.println("");
        }

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

Efekt:

```
Kategoria: Jedzenie
Produkt: Ogorek
Produkt: Zupa
Produkt: Pies

Kategoria: Kosmetyki
Produkt: Szminka
Produkt: Chusteczki
Produkt: Burak

--------------------
Produkt: Ogorek
Kategoria: Jedzenie

Produkt: Zupa
Kategoria: Jedzenie

Produkt: Pies
Kategoria: Jedzenie

Produkt: Szminka
Kategoria: Kosmetyki

Produkt: Chusteczki
Kategoria: Kosmetyki

Produkt: Burak
Kategoria: Kosmetyki
```

d. Widok z datagrip

Tabela Product:

	DBID	PRODUCTNAME	UNITSONSTOCK	SUPPLIER_FK	CATEGORY_FK
1	135	Ogorek	3	134	254
2	136	Zupa	4	134	254
3	137	Pies	5	134	254
4	251	Szminka	6	250	255
5	252	Chusteczki	7	250	255
6	253	Burak	8	250	255

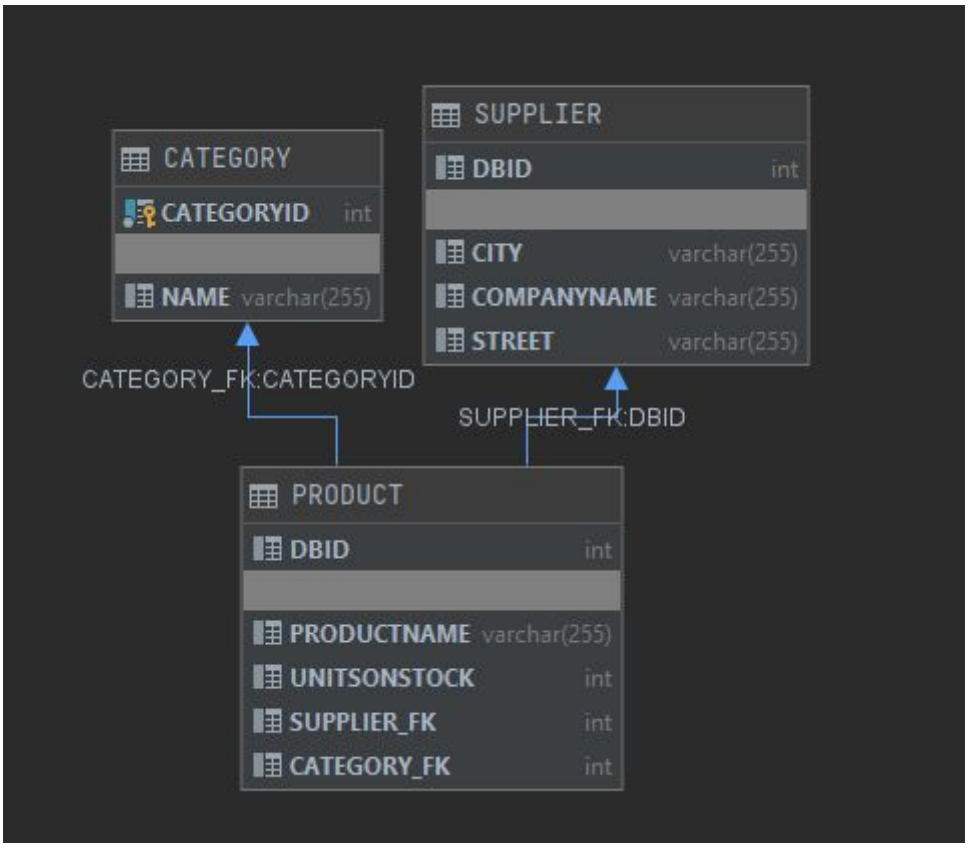
Tabela Supplier:

	DBID	CITY	COMPANYNAME	STREET
1	134	Krakow	Facebook	Borowego
2	250	Wrocław	Biedronka	Rynek

Tabela Category:

	CATEGORYID	NAME
1	254	Jedzenie
2	255	Kosmetyki

Diagram bazy:



VII. Relacja wiele do wielu

a. Stworzenie klasy

```
@Entity
public class Invoice implements Comparable{
  @Id
  @GeneratedValue(strategy = GenerationType.AUTO)
  private int dbID;
```

```
    private int invoiceNumber;
    private int quantity;

    @ManyToMany(mappedBy = "invoiceSet")
    private Set<Product> productSet;

    public Invoice(){}

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

    public int getDbID() {
        return dbID;
    }

    public int getInvoiceNumber() {
        return invoiceNumber;
    }

    public int getQuantity() {
        return quantity;
    }

    public void setInvoiceNumber(int invoiceNumber) {
        this.invoiceNumber = invoiceNumber;
    }

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

    public Set<Product> getProductSet() {
        return productSet;
    }

    public void setProductSet(Set<Product> productSet) {
        this.productSet = productSet;
    }

    public int compareTo(Object o){
        if(o instanceof Invoice){
            return Integer.compare(((Invoice) o).invoiceNumber, this.invoiceNumber);
        }

        return -1;
    }
}
```

b. Zmiany w Product

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

public Set<Invoice> getInvoiceSet() {
  return invoiceSet;
}

public void setInvoiceSet(Set<Invoice> invoiceSet) {
  this.invoiceSet = invoiceSet;
}
```

c. Dodanie nowych produktów i sprzedanie ich na różnych fakturach

```
public static void main(String[] args) {
  Category category1 = new Category("Ubrania");

  Product product1 = new Product("Bluza", 9);
  Product product2 = new Product("Spodnie", 10);
  Product product3 = new Product("Czapka", 11);

  Set<Product> productSet = new TreeSet<>();
  productSet.add(product1);
  productSet.add(product2);
  productSet.add(product3);

  Supplier supplier = new Supplier("HM", "Dluga", "Poznan");

  product1.setSupplier(supplier);
  product2.setSupplier(supplier);
  product3.setSupplier(supplier);
  supplier.setProductSet(productSet);

  List<Product> productList = new ArrayList<>();
  productList.add(product1);
  productList.add(product2);
  productList.add(product3);
  category1.setProductList(productList);

  Invoice invoice1 = new Invoice(1, 1);
  Invoice invoice2 = new Invoice(2, 2);
  Invoice invoice3 = new Invoice(3, 3);

  productSet = new TreeSet<>(productSet);
  invoice3.setProductSet(productSet);

  productSet = new TreeSet<>(productSet);
  productSet.remove(product1);
```

```
    invoice2.setProductSet(productSet);

    productSet = new TreeSet<>(productSet);
    productSet.remove(product2);
    invoice1.setProductSet(productSet);

    Set<Invoice> invoiceSet = new TreeSet<>();
    invoiceSet.add(invoice1);
    invoiceSet.add(invoice2);
    invoiceSet.add(invoice3);

    product3.setInvoiceSet(invoiceSet);

    invoiceSet = new TreeSet<>(invoiceSet);
    invoiceSet.remove(invoice1);
    product2.setInvoiceSet(invoiceSet);

    invoiceSet = new TreeSet<>(invoiceSet);
    invoiceSet.remove(invoice2);
    product1.setInvoiceSet(invoiceSet);


    sessionFactory = getSessionFactory();
    Session session = sessionFactory.openSession();
    Transaction tx = session.beginTransaction();

    session.save(product1);
    session.save(product2);
    session.save(product3);
    session.save(invoice1);
    session.save(invoice2);
    session.save(invoice3);
    session.save(supplier);
    session.save(category1);

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

}
```

d. Pokaż produkty sprzedane w ramach wybranej faktury/transakcji oraz faktury w ramach których był sprzedany wybrany produkt

```
public class HibRunner {
    private static SessionFactory sessionFactory = null;
    public static void main(String[] args) {

        sessionFactory = getSessionFactory();
```

```
    Session session = sessionFactory.openSession();
    Transaction tx = session.beginTransaction();

    String hql = "FROM Invoice";
    Query query = session.createQuery(hql);
    List<Invoice> invoices = (List<Invoice>) query.list();

    for(Invoice inv : invoices){
        System.out.printf("Faktura: %d\n", inv.getInvoiceNumber());
        for(Product prod : inv.getProductSet()){
            System.out.printf("Produkt: %s\n", prod.getProductName());
        }
        System.out.println("");
    }

    hql = "FROM Product";
    query = session.createQuery(hql);
    List<Product> products = (List<Product>) query.list();

    for(Product prod : products){
        System.out.printf("Produkt: %s\n", prod.getProductName());
        for(Invoice inv : prod.getInvoiceSet()){
            System.out.printf("Faktura: %d\n", inv.getInvoiceNumber());
        }
        System.out.println("");
    }

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

}
```

Efekt:

```
Faktura: 1
Produkt: Czapka

Faktura: 2
Produkt: Czapka
Produkt: Spodnie

Faktura: 3
Produkt: Czapka
Produkt: Bluza
Produkt: Spodnie

Produkt: Ogorek

Produkt: Zupa

Produkt: Pies

Produkt: Szminka

Produkt: Chusteczki

Produkt: Burak

Produkt: Bluza
Faktura: 3

Produkt: Spodnie
Faktura: 3
Faktura: 2

Produkt: Czapka
Faktura: 1
Faktura: 3
Faktura: 2
```

e. Widok z datagrip

Tabela Product:

	DBID	PRODUCTNAME	UNITSONSTOCK	SUPPLIER_FK	CATEGORY_FK
1	135	Ogorek	3	134	254
2	136	Zupa	4	134	254
3	137	Pies	5	134	254
4	251	Szminka	6	250	255
5	252	Chusteczki	7	250	255
6	253	Burak	8	250	255
7	373	Bluza	9	377	376
8	374	Spodnie	10	377	376
9	375	Czapka	11	377	376

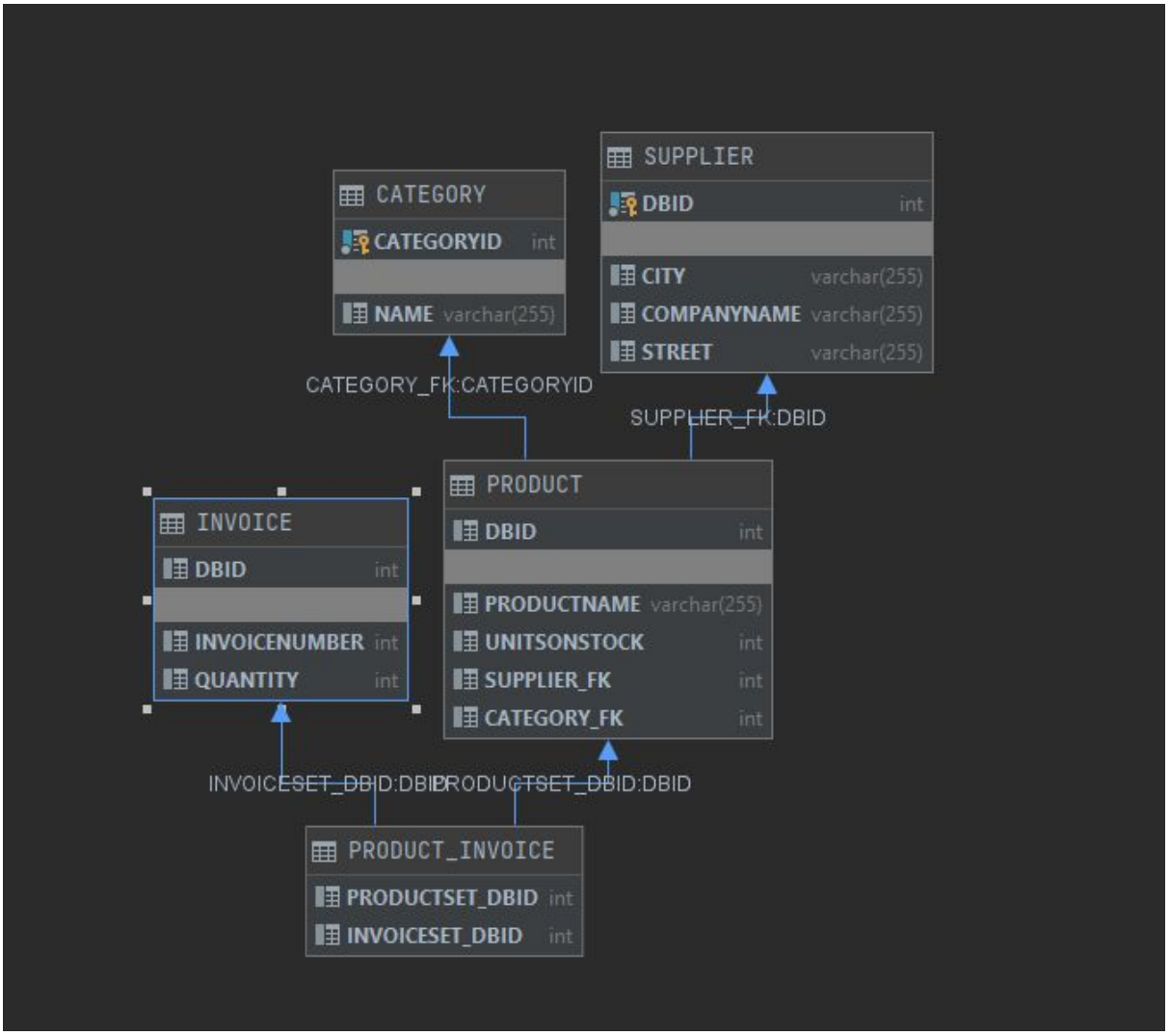
Tabela Invoice:

	DBID	INVOICENUMBER	QUANTITY
1	378	1	1
2	379	2	2
3	380	3	3

Tabela Product_Invoice:

	PRODUCTSET_DBID	INVOICESET_DBID
1	373	380
2	374	379
3	374	380
4	375	378
5	375	379
6	375	380

Diagram bazy:



VIII. JPA

a. Dodanie nowych produktów oraz połączenie ich z dostawcą

W klasie Product oraz Supplier nie trzeba było nic zmieniać, ponieważ adnotacje pozostają takie same

```
public class HibRunner {
  public static void main(String[] args) {
     Product product1 = new Product("Zelki", 10);
     Product product2 = new Product("Cukierki", 11);
     Product product3 = new Product("Grzyby", 12);
     Set<Product> productSet = new TreeSet<>();
     productSet.add(product1);
     productSet.add(product2);
     productSet.add(product3);

     Supplier supplier = new Supplier("Microsoft", "Bema", "Oświęcim");
     supplier.setProductSet(productSet);
```

```
        product1.setSupplier(supplier);
        product2.setSupplier(supplier);
        product3.setSupplier(supplier);

        EntityManagerFactory emf =
Persistence.createEntityManagerFactory("KrzysztofHardekJPA");
        EntityManager em = emf.createEntityManager();
        EntityTransaction etx = em.getTransaction();
        etx.begin();

        em.persist(product1);
        em.persist(product2);
        em.persist(product3);
        em.persist(supplier);

        etx.commit();
        em.close();
    }
}
```

b. Nowy plik konfiguracyjny

```
<?xml version="1.0"?>
<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="KrzysztofHardekJPA"
                    transaction-type="RESOURCE_LOCAL">
    <properties>
      <property name="hibernate.connection.driver_class"
                value="org.apache.derby.jdbc.ClientDriver"/>
      <property name="hibernate.connection.url"
                value="jdbc:derby://localhost/KrzysztofHardekJPA"/>
      <property name="hibernate.show_sql" value="true" />
      <property name="hibernate.format_sql" value="true" />
      <property name="hibernate.hbm2ddl.auto" value="create" />
    </properties>
  </persistence-unit>
</persistence>
```

c. dok z datagrip
d. Wi

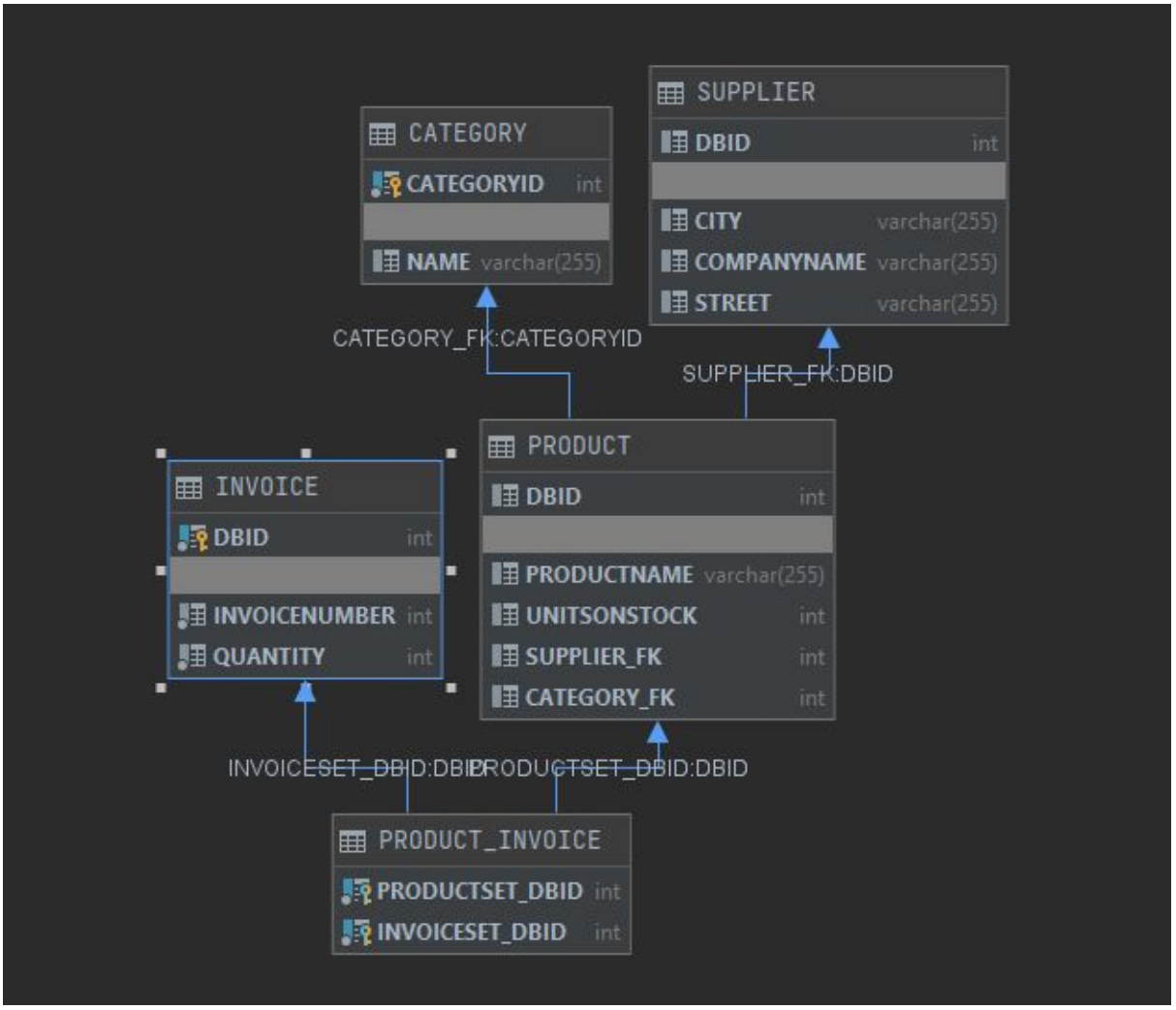
Tabela Product:

	DBID	PRODUCTNAME	UNITSONSTOCK	SUPPLIER_FK	CATEGORY_FK
1	1	Zelki	10	4	<null>
2	2	Cukierki	11	4	<null>
3	3	Grzyby	12	4	<null>

Tabela Supplier:

	DBID	CITY	COMPANYNAME	STREET
1	4	Oświęcim	Microsoft	Bema

Diagram bazy:



IX. Kaskady

a. Zmiany w adnotacjach

Klasa Product:

```
@ManyToMany(cascade = CascadeType.PERSIST)
private Set<Invoice> invoiceSet;
```

Klasa Invoice:

```
@ManyToMany(mappedBy = "invoiceSet", cascade = CascadeType.PERSIST)
private Set<Product> productSet;
```

b. Kaskadowe Tworzenie faktur oraz produktów

```
public class HibRunner {
  public static void main(String[] args) {
     Product product1 = new Product("Papier", 13);
     Product product2 = new Product("Chusteczki", 14);
     Product product3 = new Product("Recznik", 15);
     Set<Product> productSet = new TreeSet<>();
     productSet.add(product1);
     productSet.add(product2);
     productSet.add(product3);

     Supplier supplier = new Supplier("Kaufland", "Dluga", "Wroclaw");
     supplier.setProductSet(productSet);

     product1.setSupplier(supplier);
     product2.setSupplier(supplier);
     product3.setSupplier(supplier);

     Invoice invoice1 = new Invoice(1, 1);
     Invoice invoice2 = new Invoice(2, 2);
     Invoice invoice3 = new Invoice(3, 3);

     productSet = new TreeSet<>(productSet);
     invoice3.setProductSet(productSet);

     productSet = new TreeSet<>(productSet);
     productSet.remove(product1);
     invoice2.setProductSet(productSet);

     productSet = new TreeSet<>(productSet);
     productSet.remove(product2);
     invoice1.setProductSet(productSet);

     Set<Invoice> invoiceSet = new TreeSet<>();
     invoiceSet.add(invoice1);
     invoiceSet.add(invoice2);
     invoiceSet.add(invoice3);
```

```
        product3.setInvoiceSet(invoiceSet);

        invoiceSet = new TreeSet<>(invoiceSet);
        invoiceSet.remove(invoice1);
        product2.setInvoiceSet(invoiceSet);

        invoiceSet = new TreeSet<>(invoiceSet);
        invoiceSet.remove(invoice2);
        product1.setInvoiceSet(invoiceSet);

        EntityManagerFactory emf =
Persistence.createEntityManagerFactory("KrzysztofHardekJPA");
        EntityManager em = emf.createEntityManager();
        EntityTransaction etx = em.getTransaction();
        etx.begin();

        em.persist(product3);
        em.persist(supplier);

        etx.commit();
        em.close();
    }
}
```

c. Widok z datagrip

Tabela Product:

	DBID	PRODUCTNAME	UNITSONSTOCK	SUPPLIER_FK	CATEGORY_FK
1	1	Recznik	15	7	<null>
2	3	Papier	13	7	<null>
3	4	Chusteczki	14	7	<null>

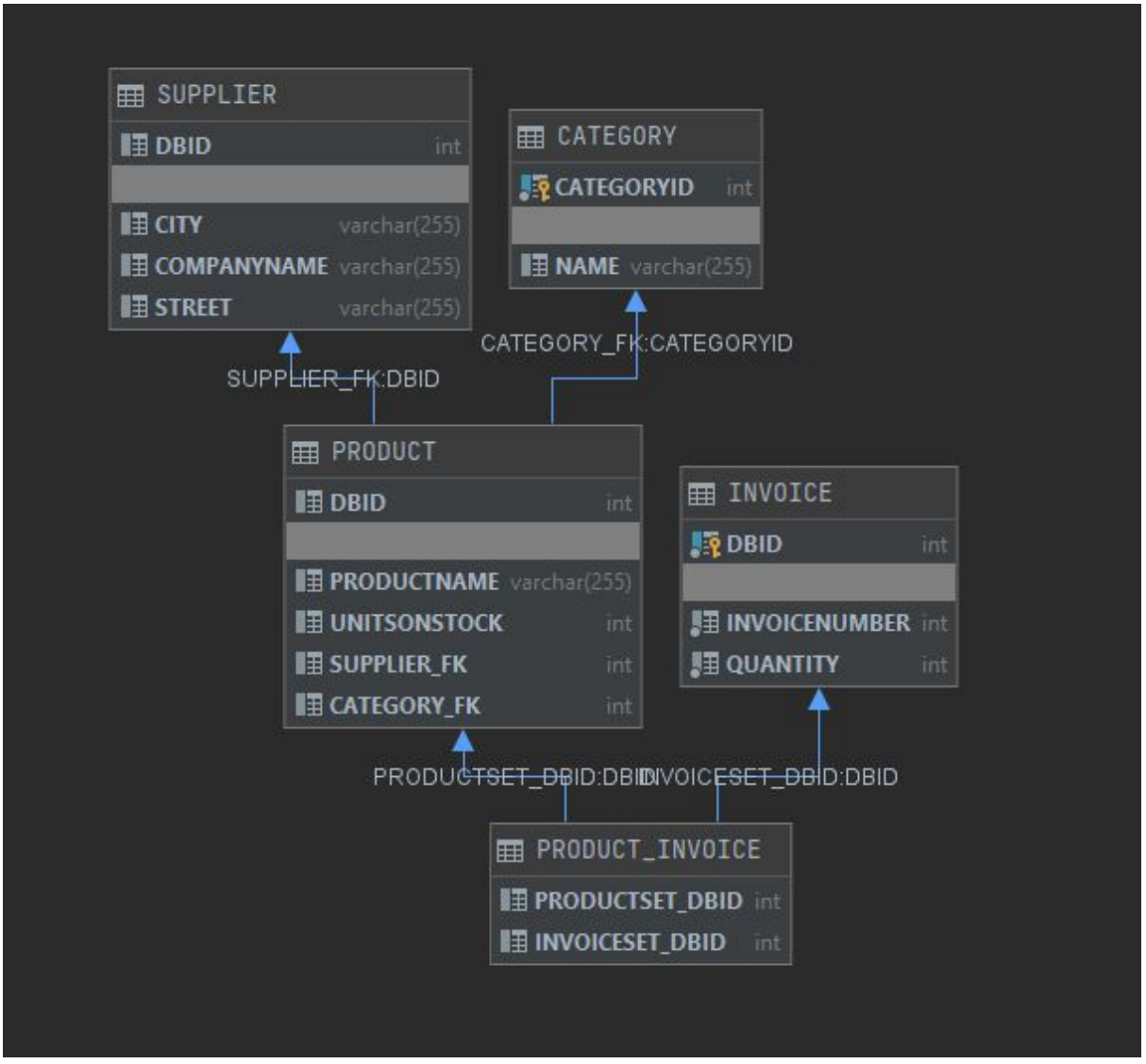
Tabela Invoice:

	DBID	INVOICENUMBER	QUANTITY
1	2	3	3
2	5	2	2
3	6	1	1

Tabela Product_Invoice:

	PRODUCTSET_DBID	INVOICESET_DBID
1	1	2
2	1	5
3	1	6
4	3	2
5	4	2
6	4	5

Diagram bazy:



X. Embedded class

a. Dodanie i wbudowanie klasy

```java
@Embeddable
public class Address {
    private String street;
    private String city;
    private String zipCode;

    public Address(){}

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

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

    public String getCity() {
        return city;
    }

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

    public String getStreet() {
        return street;
    }

    public String getZipCode() {
        return zipCode;
    }

    public void setZipCode(String zipCode) {
        this.zipCode = zipCode;
    }
}
```

W klasie Supplier:

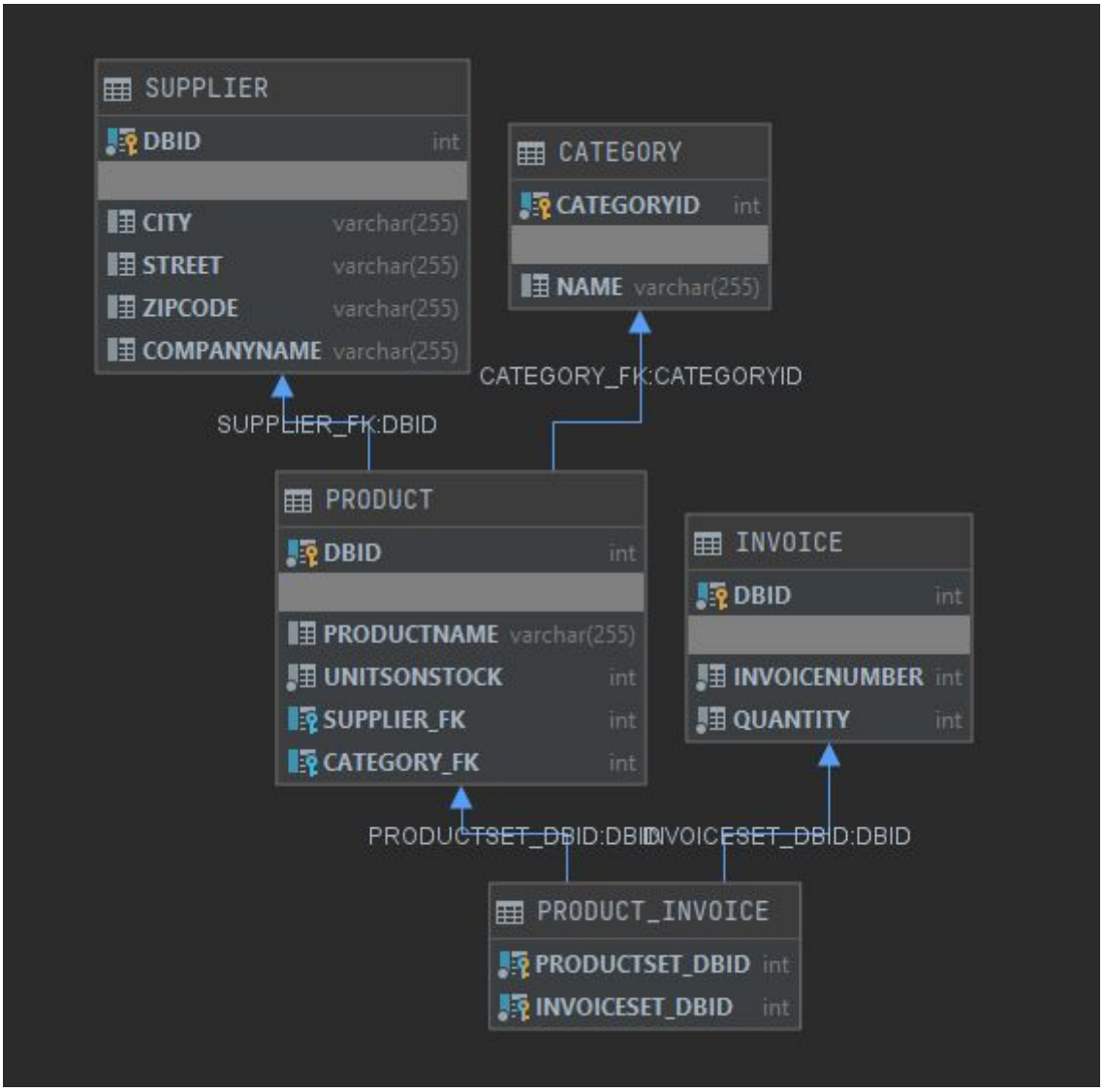
```java
@Embedded
private Address address;
```

b. Widok z datagrip

Tabela Supplier:

	DBID	CITY	STREET	ZIPCODE	COMPANYNAME
1	1	Krakow	Florianska	32-613	McDonald

Diagram bazy:



c. Mapowanie do osobnych tabel

Zmiany w Klasie Supplier:

```
@Entity
@SecondaryTable(name = "ADDRESS_TBL")
public class Supplier {
  @Id
  @GeneratedValue(strategy = GenerationType.AUTO)
```

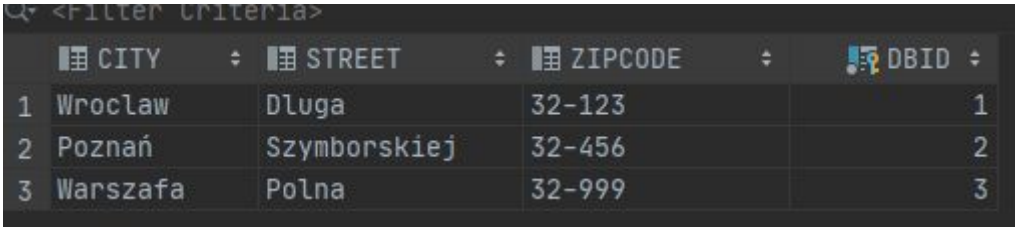
```
private int dbID;

private String companyName;

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

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

@Column(table = "ADDRESS_TBL")
private String zipCode;
```

d. Widok z datagrip

Tabela Supplier:

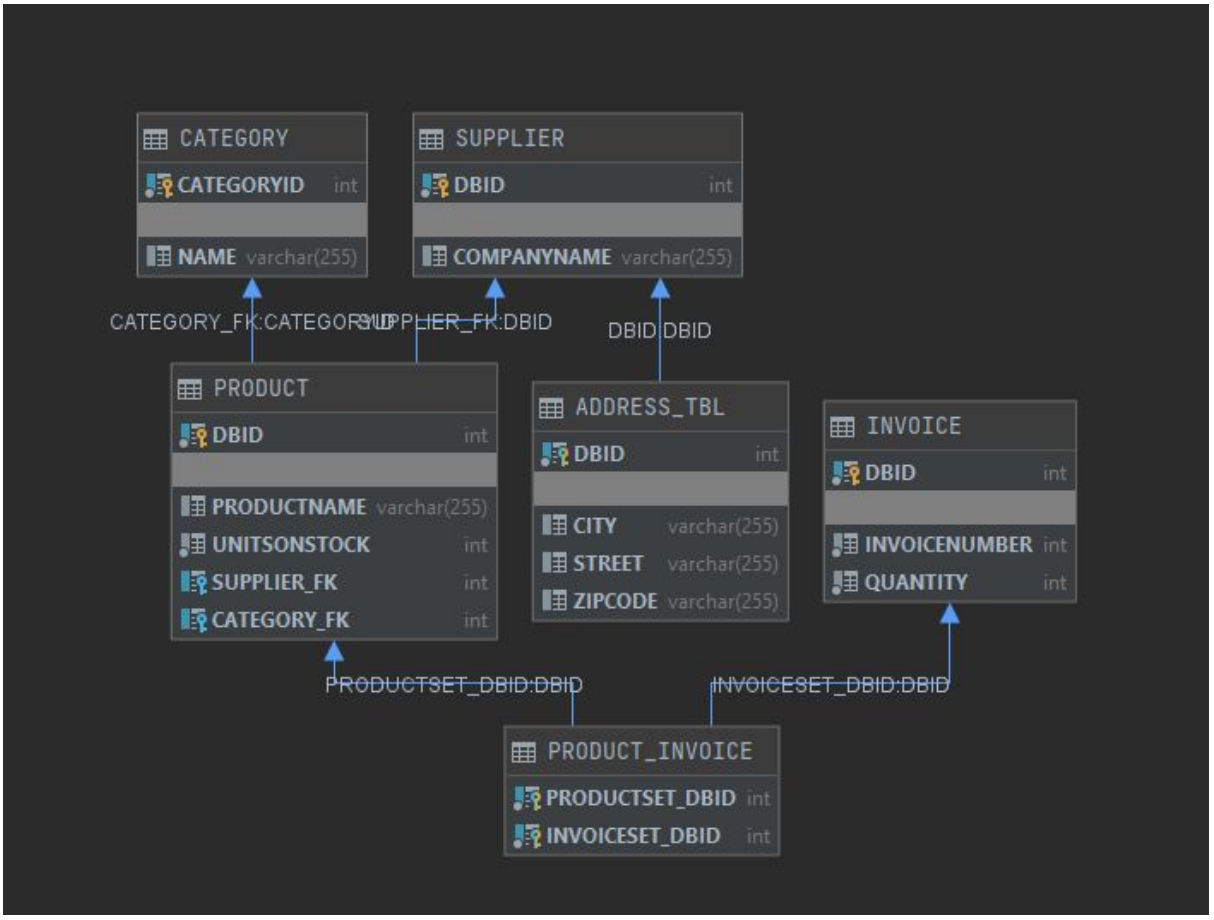
	DBID	COMPANYNAME
1	1	Kaufland
2	2	Lidl
3	3	Tesco

Tabela Address_Tbl:



	CITY	STREET	ZIPCODE	DBID
1	Wroclaw	Dluga	32-123	1
2	Poznań	Szymborskiej	32-456	2
3	Warszafa	Polna	32-999	3

Schemat bazy:



XI. Dziedziczenie

a. Tworzenie obiektów

```
public class HibRunner {
   public static void main(String[] args) {
      Customer customer1 = new Customer(0.5, "Super firma", "Krotka", "Gdansk",
"32-123");
      Customer customer2 = new Customer(0.6, "Grynn", "Dluga", "Nysa", "32-456");
      Customer customer3 = new Customer(0.6, "Tortex", "Biala", "Zgorzelsk", "32-789");

      Supplier supplier1 = new Supplier("831239-1237810", "Drutex", "Mokra", "Lodz",
"32-743");
      Supplier supplier2 = new Supplier("1293-12389", "ASD", "Czerwona", "Wlosienica",
"32-937");
      Supplier supplier3 = new Supplier("17283-343912", "FKKO", "Czarna", "Osiek",
"32-867");

      EntityManagerFactory emf =
Persistence.createEntityManagerFactory("KrzysztofHardekJPA");
      EntityManager em = emf.createEntityManager();
      EntityTransaction etx = em.getTransaction();
      etx.begin();
```

```
        em.persist(supplier1);
        em.persist(supplier2);
        em.persist(supplier3);
        em.persist(customer1);
        em.persist(customer2);
        em.persist(customer3);

        etx.commit();
        em.close();
    }
}
```

b. Tworzenie i modyfikacja klas

Klasa Company:

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

    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;
    }

    public String getZipCode() {
        return zipCode;
    }

    public void setZipCode(String zipCode) {
        this.zipCode = zipCode;
    }

    public String getStreet() {
        return street;
    }

    public void setCity(String city) {
```

```
        this.city = city;
    }

    public String getCity() {
        return city;
    }

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

    public int getDbID() {
        return dbID;
    }

    public void setCompanyName(String companyName) {
        this.companyName = companyName;
    }

    public String getCompanyName() {
        return companyName;
    }
}
```

Klasa Customer:

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

    public Customer(){}

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

    public double getDiscount() {
        return discount;
    }

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

Klasa Supplier:

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

   @OneToMany(mappedBy = "supplier")
   private Set<Product> productSet;

   public Supplier(){}

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

   public Set<Product> getProductSet() {
      return productSet;
   }

   public void setProductSet(Set<Product> productSet) {
      this.productSet = productSet;
   }
}
```

c. Strategia single table

Wstawiam adnotacje przed nazwą klasy Company

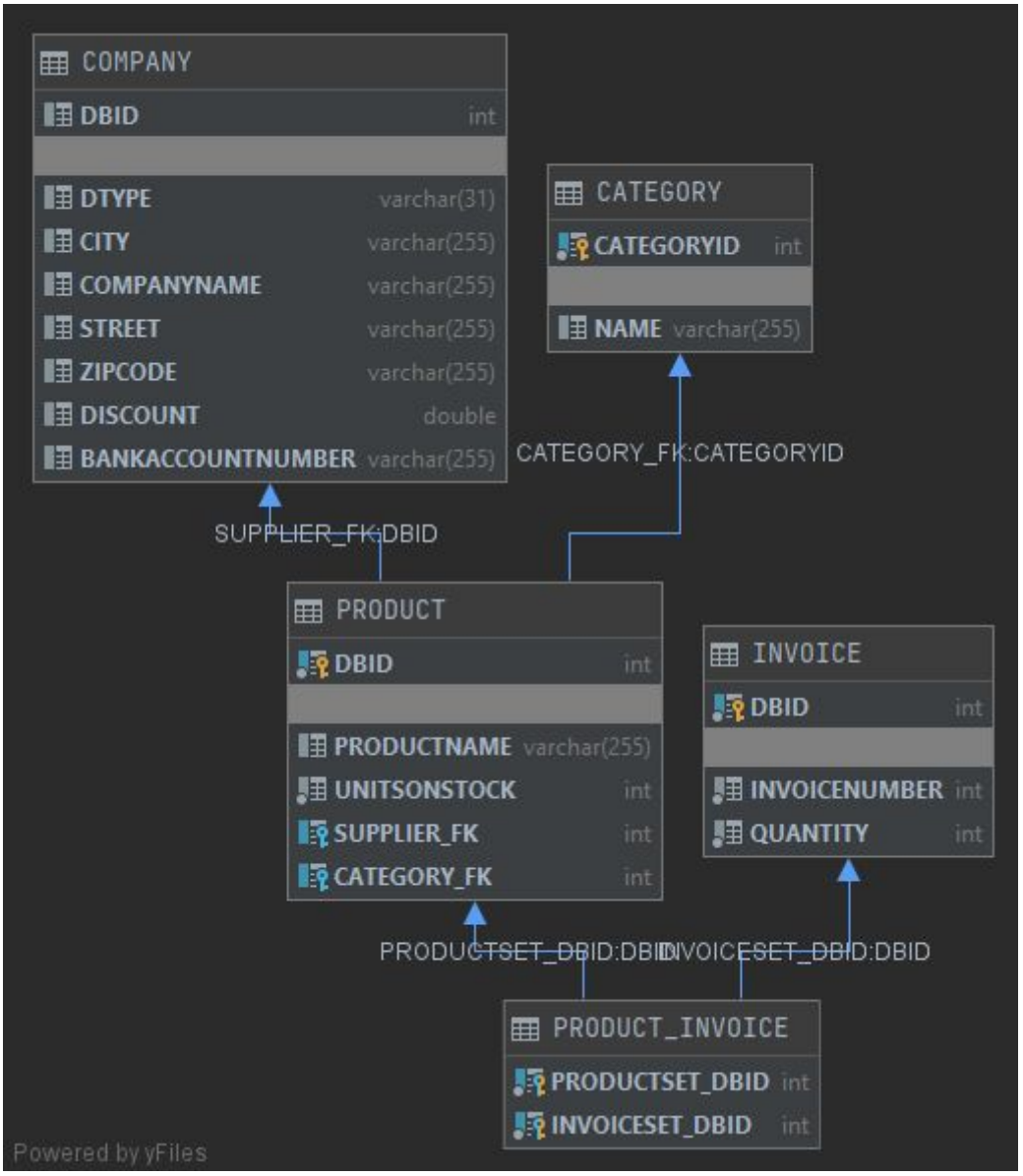
```java
@Inheritance(strategy = InheritanceType.SINGLE_TABLE)
```

d. Widok z datagrip

Tabela Company:

	DTYPE	DBID	CITY	COMPANYNAME	STREET	ZIPCODE	DISCOUNT	BANKACCOUNTNUMBER
1	Supplier	1	Lodz	Drutex	Mokra	32-743	<null>	831239-1237810
2	Supplier	2	Wlosienica	ASD	Czerwona	32-937	<null>	1293-12389
3	Supplier	3	Osiek	FKKO	Czarna	32-867	<null>	17283-343912
4	Customer	4	Gdansk	Super firma	Krotka	32-123	0.5	<null>
5	Customer	5	Nysa	Grynn	Dluga	32-456	0.6	<null>
6	Customer	6	Zgorzelsk	Tortex	Biala	32-789	0.6	<null>

Diagram bazy:



e. Strategia Joined

Wstawiam adnotacje przed nazwą klasy Company:

```
@Inheritance(strategy = InheritanceType.JOINED)
```

f. Widok z datagrip

Tabela Company:

	DBID	CITY	COMPANYNAME	STREET	ZIPCODE
1	1	Lodz	Drutex	Mokra	32-743
2	2	Wlosienica	ASD	Czerwona	32-937
3	3	Osiek	FKKO	Czarna	32-867
4	4	Gdansk	Super firma	Krotka	32-123
5	5	Nysa	Grynn	Dluga	32-456
6	6	Zgorzelsk	Tortex	Biala	32-789

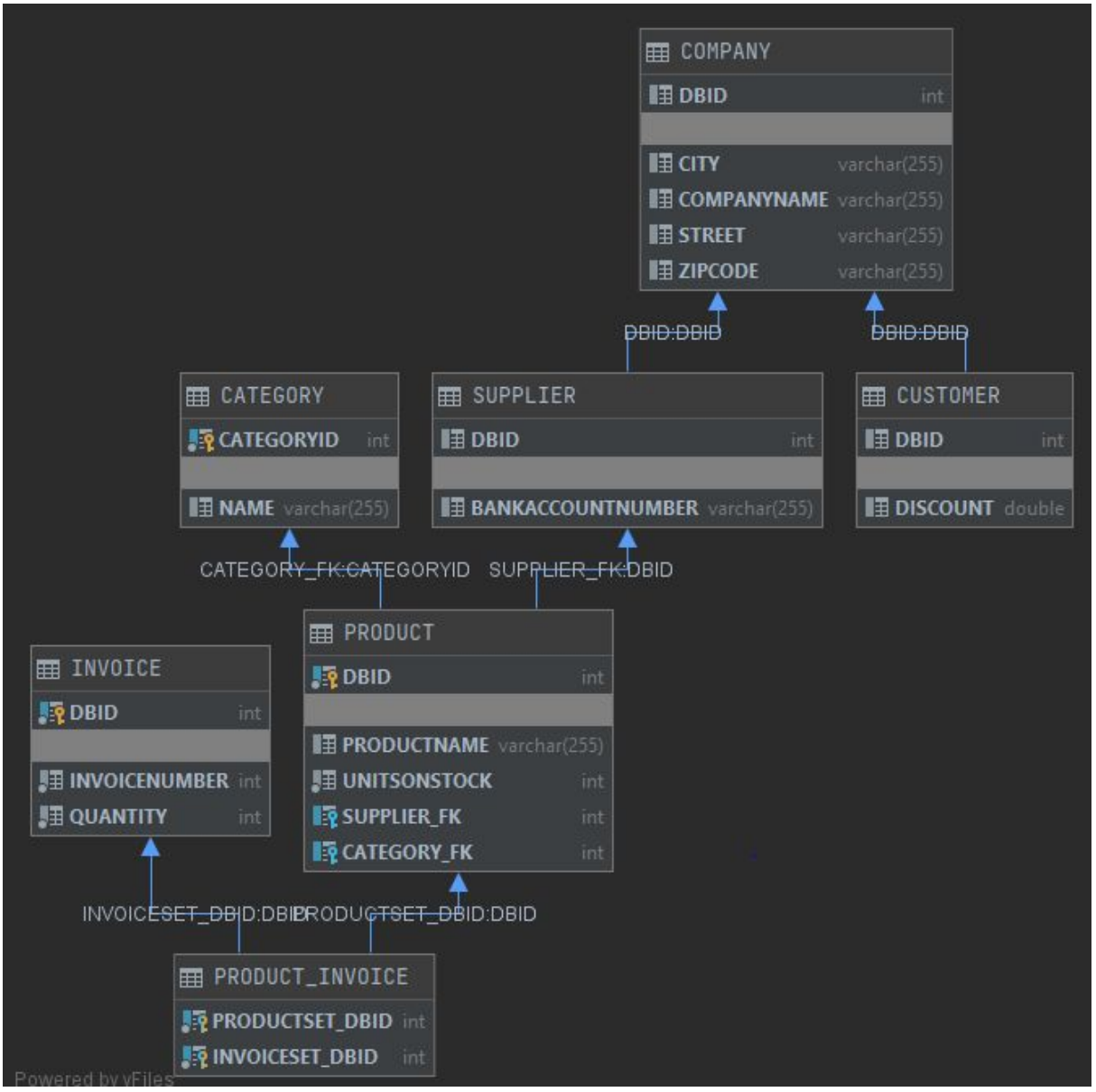
Tabela Supplier:

	BANKACCOUNTNUMBER	DBID
1	831239-1237810	1
2	1293-12389	2
3	17283-343912	3

Tabela Customer:

	DISCOUNT	DBID
1	0.5	4
2	0.6	5
3	0.6	6

Diagram bazy:



g. Strategia Table per class

Wstawiam adnotacje przed nazwą klasy Company:

```
@Inheritance(strategy = InheritanceType.TABLE_PER_CLASS)
```

h. Widok z datagrip

Tabela Company:

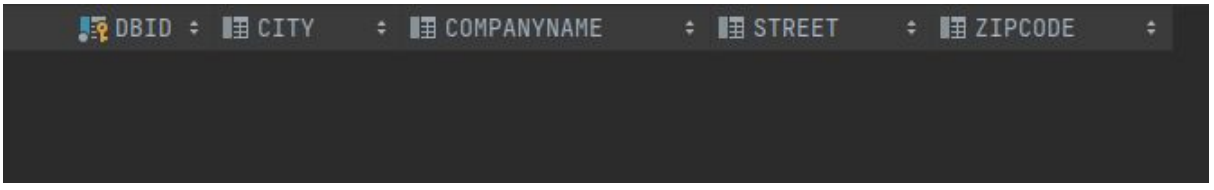


Tabela Customer:

	DBID	CITY	COMPANYNAME	STREET	ZIPCODE	DISCOUNT
1	4	Gdansk	Super firma	Krotka	32-123	0.5
2	5	Nysa	Grynn	Dluga	32-456	0.6
3	6	Zgorzelsk	Tortex	Biala	32-789	0.6

Tabela Supplier:

	DBID	CITY	COMPANYNAME	STREET	ZIPCODE	BANKACCOUNTNUMBER
1	1	Lodz	Drutex	Mokra	32-743	831239-1237810
2	2	Wlosienica	ASD	Czerwona	32-937	1293-12389
3	3	Osiek	FKKO	Czarna	32-867	17283-343912

Schemat bazy: