Hibernate

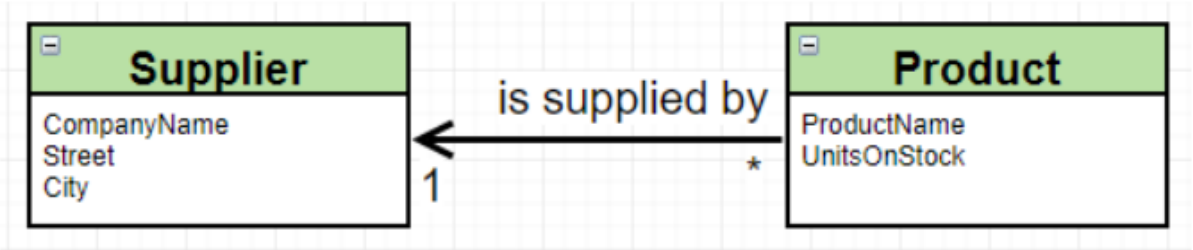
Jakub Białecki, Przemysław Popowski, Jakub Worek

0. Wstępne zadania:

```
Hibernate:
    drop sequence Product_SEQ restrict
Hibernate:
    create sequence Product_SEQ start with 1 increment by 50
Hibernate:
    create table Product (
        productID integer not null,
        unitsOnStock integer not null,
        productName varchar(255),
        primary key (productID)
    )
Hibernate:

values
    next value for Product_SEQ
Hibernate:
    /* insert for
        org.example.Product */insert
    into
        Product (productName, unitsOnStock, productID)
    values
        (?, ?, ?)

Process finished with exit code 0
```

	PRODUCTID	UNITSONSTOCK	PRODUCTNAME
1	1	4	Flamaster

1. Zmodyfikuj model wprowadzając pojęcie Dostawcy jak poniżej



Main.java

```java
package org.example;

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

public class Main {

    5 usages
    private static SessionFactory sessionFactory = null;

    public static void main(String[] args) {
        System.out.println("Hello world!");
        sessionFactory = getSessionFactory();
        Session session = sessionFactory.openSession();
        Product product = new Product( productName: "Flamaster", unitsOnStock: 4);
        Transaction tx = session.beginTransaction();
        session.save(product);
        tx.commit();
        session.close();
    }

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

Product.java

```java
package org.example;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;

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

    public Product() {
    }

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

hibernate.cfg.xml | Supplier.java | PRODUCT | SUPPLIER | Produc

2 rows | Tx: Auto | DDL | CSV

WHERE | ORDER BY SUPPLIERID DES

	SUPPLIERID	CITY	STREET	COMPANYNAME
1	4952	Kraków	Budryka	Kapitol

```java
package org.example;

import jakarta.persistence.*;

import java.util.HashSet;
import java.util.Set;

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

    public Supplier() {
    }

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


    @Override
    public String toString() {
        return "Supplier{" +
                "supplierID=" + supplierID +
                ", street='" + street + '\'' +
                ", city='" + city + '\'' +
                '}';
    }
}
```

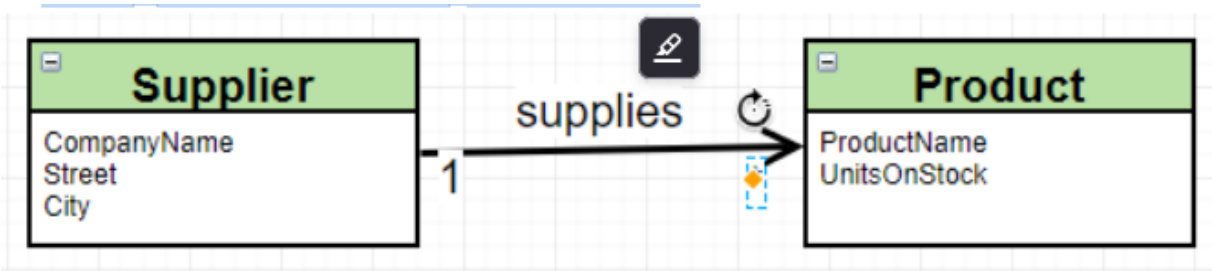
```java
package org.example;

import jakarta.persistence.*;

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

    1 usage
    public Supplier getSupplier() {
        return supplier;
    }

    3 usages
    @ManyToOne
    private Supplier supplier;

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

    public Product() {
    }

    no usages
    public Product(String productName, int unitsOnStock) {
        this.productName = productName;
        this.unitsOnStock = unitsOnStock;
    }

    @Override
    public String toString() {
        return "Product{" +
                "productID=" + productID +
                ", productName='" + productName + '\'' +
                ", unitsOnStock=" + unitsOnStock +
                ", supplier=" + supplier +
                '}';
    }


}
```

```java
public static void main(String[] args) {
    sessionFactory = getSessionFactory();
    Session session = sessionFactory.openSession();
    Transaction tx = session.beginTransaction();
    /* znajdujemy ostatni produkt i dodajemy nowego dostawce */
    Product foundProduct = session.get(Product.class, o: 1);
    Supplier supplier = new Supplier( companyName: "Kapitol", street: "Budryka", city: "Kraków");
    session.save(supplier);
    /*ustawiamy relacje jeden do wielu*/
    foundProduct.setSupplier(supplier);
    System.out.println(foundProduct.getSupplier());
    tx.commit();
    session.close();
}
```

```
cze 04, 2024 4:03:13 PM org.hibernate.Version logVersion
INFO: HHH000412: Hibernate ORM core version 6.4.4.Final
cze 04, 2024 4:03:13 PM org.hibernate.cache.internal.RegionFactoryInitiator initiateService
INFO: HHH000026: Second-level cache disabled
cze 04, 2024 4:03:13 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl configur
WARN: HHH10001002: Using built-in connection pool (not intended for production use)
cze 04, 2024 4:03:13 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCre
INFO: HHH10001005: Loaded JDBC driver class: org.apache.derby.jdbc.ClientDriver
cze 04, 2024 4:03:13 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCre
INFO: HHH10001012: Connecting with JDBC URL [jdbc:derby://127.0.0.1/MyLabDatabase]
cze 04, 2024 4:03:13 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCre
INFO: HHH10001001: Connection properties: {}
cze 04, 2024 4:03:13 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCre
INFO: HHH10001003: Autocommit mode: false
cze 04, 2024 4:03:13 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl$PooledCo
INFO: HHH10001115: Connection pool size: 20 (min=1)
cze 04, 2024 4:03:14 PM org.hibernate.engine.transaction.jta.platform.internal.JtaPlatformInitiator initiateService
INFO: HHH000489: No JTA platform available (set 'hibernate.transaction.jta.platform' to enable JTA platform integra
cze 04, 2024 4:03:14 PM org.hibernate.resource.transaction.backend.jdbc.internal.DdlTransactionIsolatorNonJtaImpl g
INFO: HHH10001501: Connection obtained from JdbcConnectionAccess [org.hibernate.engine.jdbc.env.internal.JdbcEnviro
Hibernate:
    alter table Supplier
       add column companyName varchar(255)
Hibernate:
    select
        p1_0.productID,
        p1_0.productName,
        s1_0.supplierID,
        s1_0.city,
        s1_0.companyName,
        s1_0.street,
        p1_0.unitsOnStock
    from
        Product p1_0
    left join
        Supplier s1_0
            on s1_0.supplierID=p1_0.supplier_supplierID
    where
        p1_0.productID=?
Hibernate:

values
    next value for Supplier_SEQ
Supplier{supplierID=4952, street='Budryka', city='Kraków'}
Hibernate:
    /* insert for
        org.example.Supplier */insert
    into
        Supplier (city, companyName, street, supplierID)
    values
        (?, ?, ?, ?)
Hibernate:
    /* update
        for org.example.Product */update Product
    set
        productName=?,
        supplier_supplierID=?,
        unitsOnStock=?
    where
        productID=?

Process finished with exit code 0
```

```
<mapping class="org.example.Supplier"/>
```

2.Odwróć relacje zgodnie z poniższym schematem

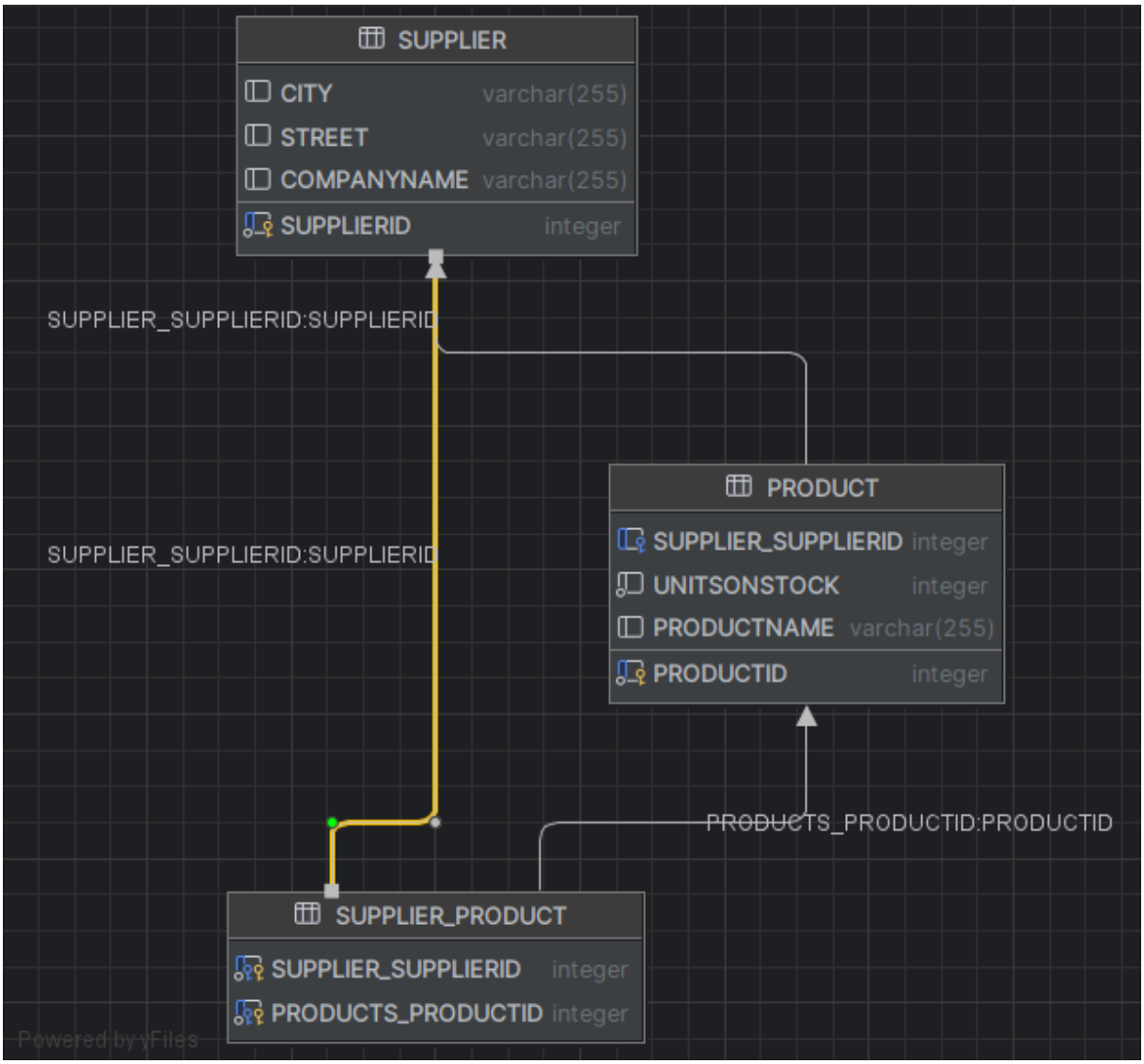


2.1 z tabelą łącznikową:

```
Hibernate:

values
    next value for Product_SEQ
Hibernate:

values
    next value for Supplier_SEQ
Hibernate:
    /* insert for
        org.example.Product */insert
    into
        Product (productName, unitsOnStock, productID)
    values
        (?, ?, ?)
Hibernate:
    /* insert for
        org.example.Product */insert
    into
        Product (productName, unitsOnStock, productID)
    values
        (?, ?, ?)
Hibernate:
    /* insert for
        org.example.Product */insert
    into
        Product (productName, unitsOnStock, productID)
    values
        (?, ?, ?)
Hibernate:
    /* insert for
        org.example.Product */insert
    into
        Product (productName, unitsOnStock, productID)
    values
        (?, ?, ?)
Hibernate:
    /* insert for
        org.example.Supplier */insert
    into
        Supplier (city, companyName, street, supplierID)
    values
        (?, ?, ?, ?)
Hibernate:
    /* insert for
        org.example.Supplier.products */insert
    into
        Supplier_Product (Supplier_supplierID, products_productID)
    values
        (?, ?)
Hibernate:
    /* insert for
        org.example.Supplier.products */insert
    into
        Supplier_Product (Supplier_supplierID, products_productID)
    values
        (?, ?)
Hibernate:
    /* insert for
        org.example.Supplier.products */insert
    into
        Supplier_Product (Supplier_supplierID, products_productID)
    values
        (?, ?)
```

```
4 usages
@OneToMany
private Set<Product> products;

4 usages
public void addProduct(Product product){
    if (products == null){
        products = new HashSet<>();
    }
    else{
        products.add(product);
    }}
```

```
public static void main(String[] args) {
    sessionFactory = getSessionFactory();
    Session session = sessionFactory.openSession();
    Transaction tx = session.beginTransaction();
    Product product = new Product( productName: "Klawiatura", unitsOnStock: 3);
    Product product1 = new Product( productName: "Myszka", unitsOnStock: 2);
    Product product2 = new Product( productName: "Monitor", unitsOnStock: 16);
    Product product3 = new Product( productName: "Jednostka Centralna", unitsOnStock: 7);
    session.save(product);
    session.save(product1);
    session.save(product2);
    session.save(product3);
    Supplier supplier = new Supplier( companyName: "D-17", street: "Kawiory", city: "Kraków");
    session.save(supplier);
    supplier.addProduct(product);
    supplier.addProduct(product1);
    supplier.addProduct(product2);
    supplier.addProduct(product3);
    tx.commit();
    session.close();
}
```



yLabDatabase@127.0.0.1] SUPPLIER_PRODUCT [MyLabDatabase@127.0.0.1]

3 rows Tx: Auto DDL

WHERE ORDER BY PRODUCTS_PRODUCT

	SUPPLIER_SUPPLIERID	PRODUCTS_PRODUCTID
1	9952	4953
2	9952	4954
3	9952	4955

RODUCT [MyLabDatabase@127.0.0.1] SUPPLIER [MyLabDatabase@127.0.0.1]

3 rows Tx: Auto DDL

WHERE ORDER BY

	SUPPLIERID	CITY	STREET	COMPANYNAME
1	1	Kraków	Budryka	<null>
2	4952	Kraków	Budryka	Kapitol
3	9952	Kraków	Kawiory	D-17

PRODUCT [MyLabDatabase@127.0.0.1] SUPPLIER [MyLabDatabase@127.0.0.1

5 rows Tx: Auto DDL

WHERE ORDER BY

	PRODUCTID	SUPPLIER_SUPPL...	UNITSONSTOCK	PRODUCTNAME
1	1	4952	1	Kredki
2	4952	<null>	3	Klawiatura
3	4953	<null>	2	Myszka
4	4954	<null>	16	Monitor
5	4955	<null>	7	Jednostka Centralna

2.2 Bez tabeli łącznikowej:

```
4 usages
@OneToMany
@JoinColumn(name = "Supplier_FK")
private Set<Product> products;
```

```
Hibernate:
    drop table Supplier
Hibernate:
    drop sequence Product_SEQ restrict
Hibernate:
    drop sequence Supplier_SEQ restrict
Hibernate:
    create sequence Product_SEQ start with 1 increment by 50
Hibernate:
    create sequence Supplier_SEQ start with 1 increment by 50
```

```
    values
        (?, ?, ?, ?)
Hibernate:
    update
        Product
    set
        Supplier_FK=?
    where
        productID=?
Hibernate:
    update
        Product
    set
        Supplier_FK=?
    where
        productID=?
Hibernate:
    update
        Product
    set
        Supplier_FK=?
    where
        productID=?
```

```
Hibernate:
    create table Product (
        Supplier_FK integer,
        productID integer not null,
        unitsOnStock integer not null,
        productName varchar(255),
        primary key (productID)
    )
Hibernate:
    create table Supplier (
        supplierID integer not null,
        city varchar(255),
        companyName varchar(255),
        street varchar(255),
        primary key (supplierID)
    )
Hibernate:
    alter table Product
       add constraint FKve96qacvsr1a50rgwl94enru
       foreign key (Supplier_FK)
       references Supplier
Hibernate:

values
    next value for Product_SEQ
Hibernate:

values
    next value for Product_SEQ
Hibernate:

values
    next value for Supplier_SEQ
Hibernate:
    /* insert for
        org.example.Product */insert
    into
        Product (productName, unitsOnStock, productID)
    values
        (?, ?, ?)
Hibernate:
    /* insert for
        org.example.Product */insert
    into
        Product (productName, unitsOnStock, productID)
    values
        (?, ?, ?)
Hibernate:
    /* insert for
        org.example.Product */insert
    into
        Product (productName, unitsOnStock, productID)
    values
        (?, ?, ?)
Hibernate:
    /* insert for
        org.example.Product */insert
    into
        Product (productName, unitsOnStock, productID)
    values
        (?, ?, ?)
Hibernate:
    /* insert for
        org.example.Supplier */insert
    into
        Supplier (city, companyName, street, supplierID)
```

console [MyLab3Database@127.0.0.1] | PRODUCT [MyLab3Database@127.0.0.1] | SUPPLIER [MyLab3Database@127.0.0.1]

1 row | Tx: Auto | DDL | CSV

WHERE | ORDER BY

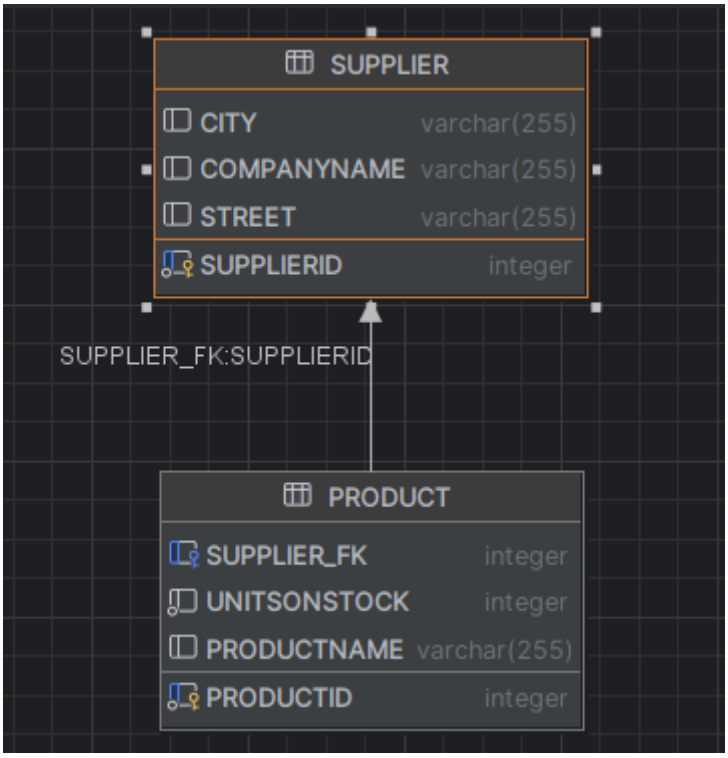
	SUPPLIERID	CITY	COMPANYNAME	STREET
1	1	Kraków	D-17	Kawiory

console [MyLab3Database@127.0.0.1] | PRODUCT [MyLab3Database@127.0.0.1] | SUPPL

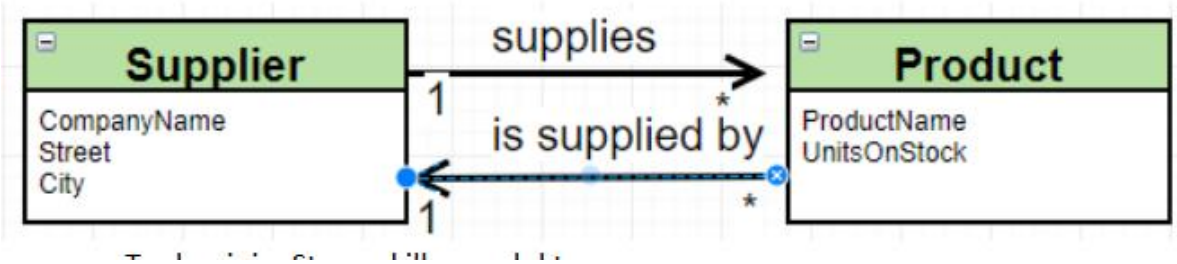
4 rows | Tx: Auto | DDL

WHERE | ORDER BY

	SUPPLIER_FK	PRODUCTID	UNITSONSTOCK	PRODUCTNAME
1	<null>	1	3	Klawiatura
2	1	2	2	Myszka
3	1	3	16	Monitor
4	1	4	7	Jednostka Centralna



3.Zamodeluj relację dwustronną jak poniżej:




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

1 usage
public void setSupplier(Supplier supplier) {
    this.supplier = supplier;
    if (!supplier.containProduct(this)) {
        supplier.addProduct(this);
    }
}
```

```
5 usages
@OneToMany
@JoinColumn(name = "Supplier_FK")
private Set<Product> products;

5 usages
public void addProduct(Product product) {
    if (products == null) {
        products = new HashSet<>();
    } else {
        products.add(product);
        product.setSupplier(this);
    }
}

1 usage
public boolean containProduct(Product product){
    return products.contains(product);
}
```

PRODUCT [MyLab4Database@127.0.0.1] SUPPLIER [MyLab4Database@127.0.0.1]

1 row Tx: Auto DDL

WHERE ORDER BY

	SUPPLIERID	CITY	COMPANYNAME	STREET
1	1	Kraków	Samogłoski:((	Kawiory

PRODUCT [MyLab4Database@127.0.0.1]

4 rows Tx: Auto DDL

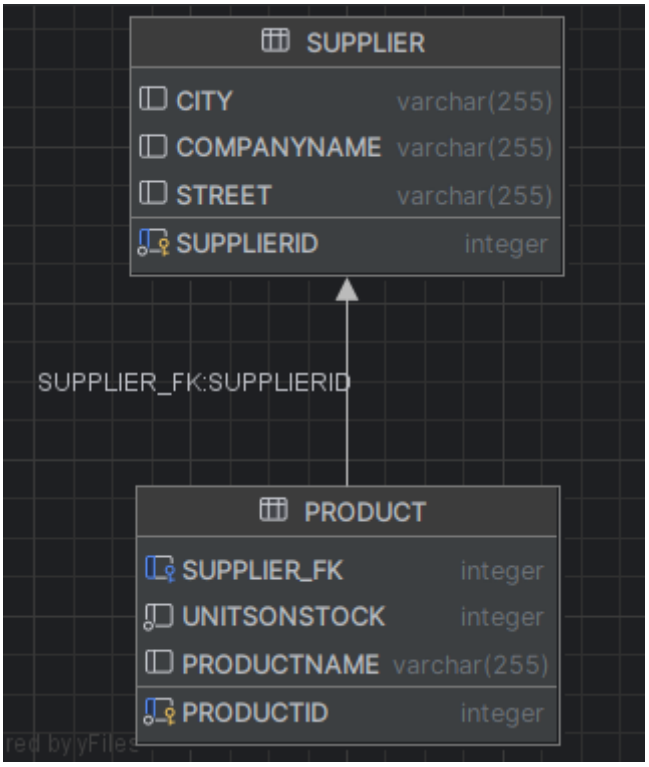
WHERE ORDER BY

	SUPPLIER_FK	PRODUCTID	UNITSONSTOCK	PRODUCTNAME
1	<null>	1	3	Klawiatura2
2	1	2	2	Myszka2
3	1	3	16	Monitor2
4	1	4	7	Jednostka Centralna2

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

```
Hibernate:
    create table Supplier (
        supplierID integer not null,
        city varchar(255),
        companyName varchar(255),
        street varchar(255),
        primary key (supplierID)
    )
Hibernate:
    alter table Product
       add constraint FKve96qacvsr1a50rgwl94enru
       foreign key (Supplier_FK)
       references Supplier
Hibernate:

values
    next value for Product_SEQ
Hibernate:

values
    next value for Product_SEQ
Hibernate:

values
    next value for Supplier_SEQ
Hibernate:
    /* insert for
        org.example.Product */insert
    into
        Product (productName, Supplier_FK, unitsOnStock, productID)
    values
        (?, ?, ?, ?)
Hibernate:
    /* insert for
        org.example.Product */insert
    into
        Product (productName, Supplier_FK, unitsOnStock, productID)
    values
        (?, ?, ?, ?)
Hibernate:
    /* insert for
        org.example.Product */insert
    into
        Product (productName, Supplier_FK, unitsOnStock, productID)
    values
        (?, ?, ?, ?)
Hibernate:
    /* insert for
        org.example.Product */insert
    into
        Product (productName, Supplier_FK, unitsOnStock, productID)
    values
        (?, ?, ?, ?)
Hibernate:
    /* insert for
        org.example.Supplier */insert
    into
        Supplier (city, companyName, street, supplierID)
    values
        (?, ?, ?, ?)
Hibernate:
    /* update
        for org.example.Product */update Product
    set
        productName=?,
        Supplier_FK=?,
```

```
Hibernate:
    drop table Supplier
Hibernate:
    drop sequence Product_SEQ restrict
Hibernate:
    drop sequence Supplier_SEQ restrict
Hibernate:
    create sequence Product_SEQ start with 1 increment by 50
Hibernate:
    create sequence Supplier_SEQ start with 1 increment by 50
Hibernate:
    create table Product (
        Supplier_FK integer,
        productID integer not null,
        unitsOnStock integer not null,
        productName varchar(255),
        primary key (productID)
    )
```



4.Dodaj klase Category z property int CategoryID, String Name oraz listą produktow

List<Product> Products

```
public static void main(String[] args) {
    sessionFactory = getSessionFactory();
    Session session = sessionFactory.openSession();
    Transaction tx = session.beginTransaction();
    Product product = new Product( productName: "Klawiatura", unitsOnStock: 3);
    Product product1 = new Product( productName: "Myszka", unitsOnStock: 2);
    Product product2 = new Product( productName: "Monitor", unitsOnStock: 16);
    Product product3 = new Product( productName: "Jednostka Centralna", unitsOnStock: 7);
    session.save(product);
    session.save(product1);
    session.save(product2);
    session.save(product3);
    Category category = new Category( name: "Klikalne");
    Category category1 = new Category( name: "Drogie");
    session.save(category);
    session.save(category1);
    category.addProduct(product1);
    category.addProduct(product2);
    category1.addProduct(product);
    category1.addProduct(product3);

    for (Product pro: category.getProducts()){
        System.out.println(pro);
    }
    tx.commit();
    session.close();
}
```

```
1 usage
public void setCategory(Category category) {
    this.category = category;
    category.addProduct(this);
}

1 usage
public boolean containCategory() { return category != null; }


1 usage
public void setSupplier(Supplier supplier) {
    this.supplier = supplier;
    if (!supplier.containProduct(this)) {
        supplier.addProduct(this);
    }
}
```

```
2 usages
@ManyToOne
private Category category;
1 usage
```

```java
package org.example;

import jakarta.persistence.*;

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

8 usages
@Entity
public class Category {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int categoryID;
    2 usages
    private String name;
    3 usages
    @OneToMany(mappedBy = "category")
    private final List<Product> products = new ArrayList<>();

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

    public Category() {
    }

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

    @Override
    public String toString() {
        return "Category{" + "categoryID=" + categoryID + ", name='" + name + '\'' + ", products=" + products +
    }

    5 usages
    public void addProduct(Product product) {
        products.add(product);
        if (!product.containCategory()) {
            product.setCategory(this);
        }
    }
}
```

PRODUCT [MyLab5Database@127.0.0.1] SUPPLIER [MyLab5Database@127.0.0.1]

4 rows Tx: Auto DDL

WHERE ORDER BY

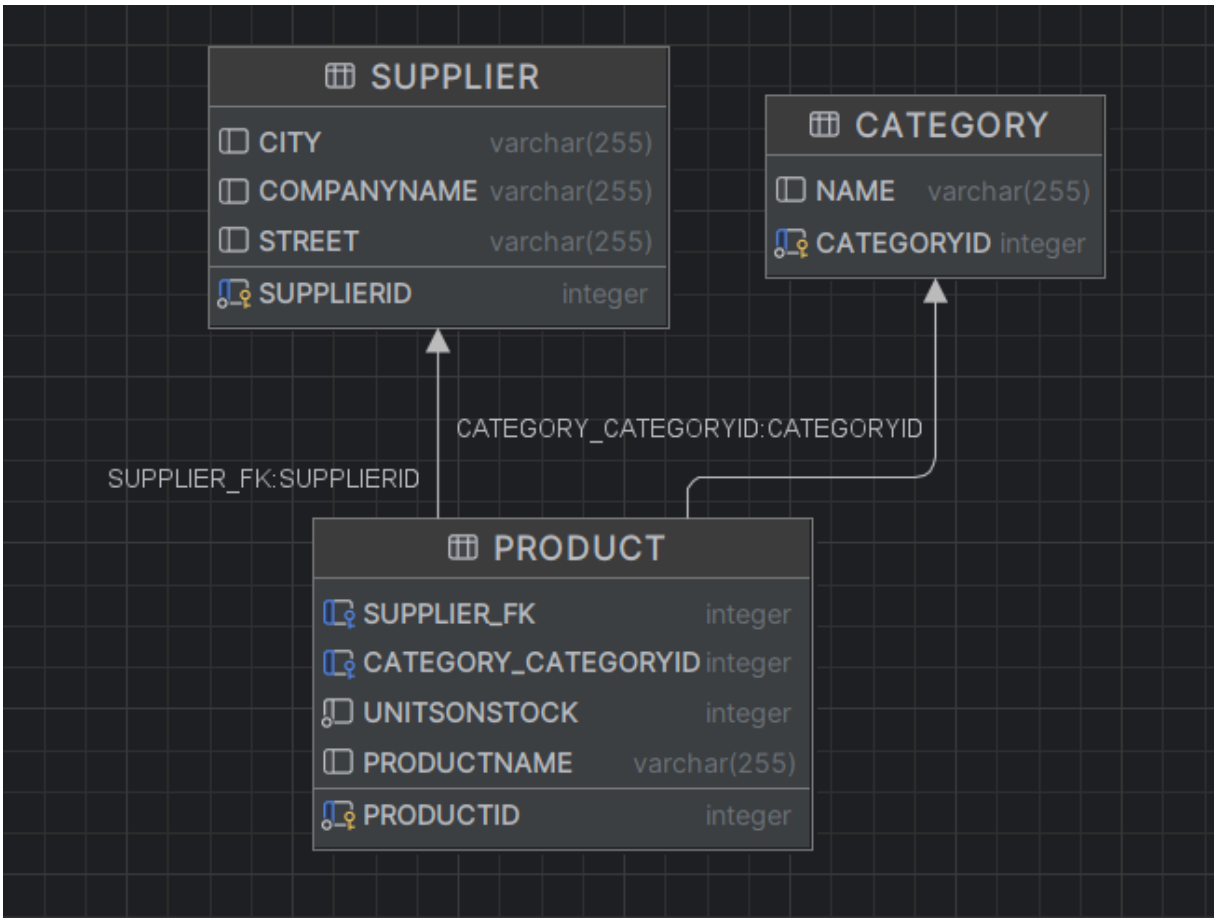
	SUPPLIER_FK	CATEGORY_CATEGORYID	PRODUCTID	UNITSONSTOCK	PRODUCTNAME
1	<null>	2	1	3	Klawiatura
2	<null>	1	2	2	Myszka
3	<null>	1	3	16	Monitor
4	<null>	2	4	7	Jednostka Centralna

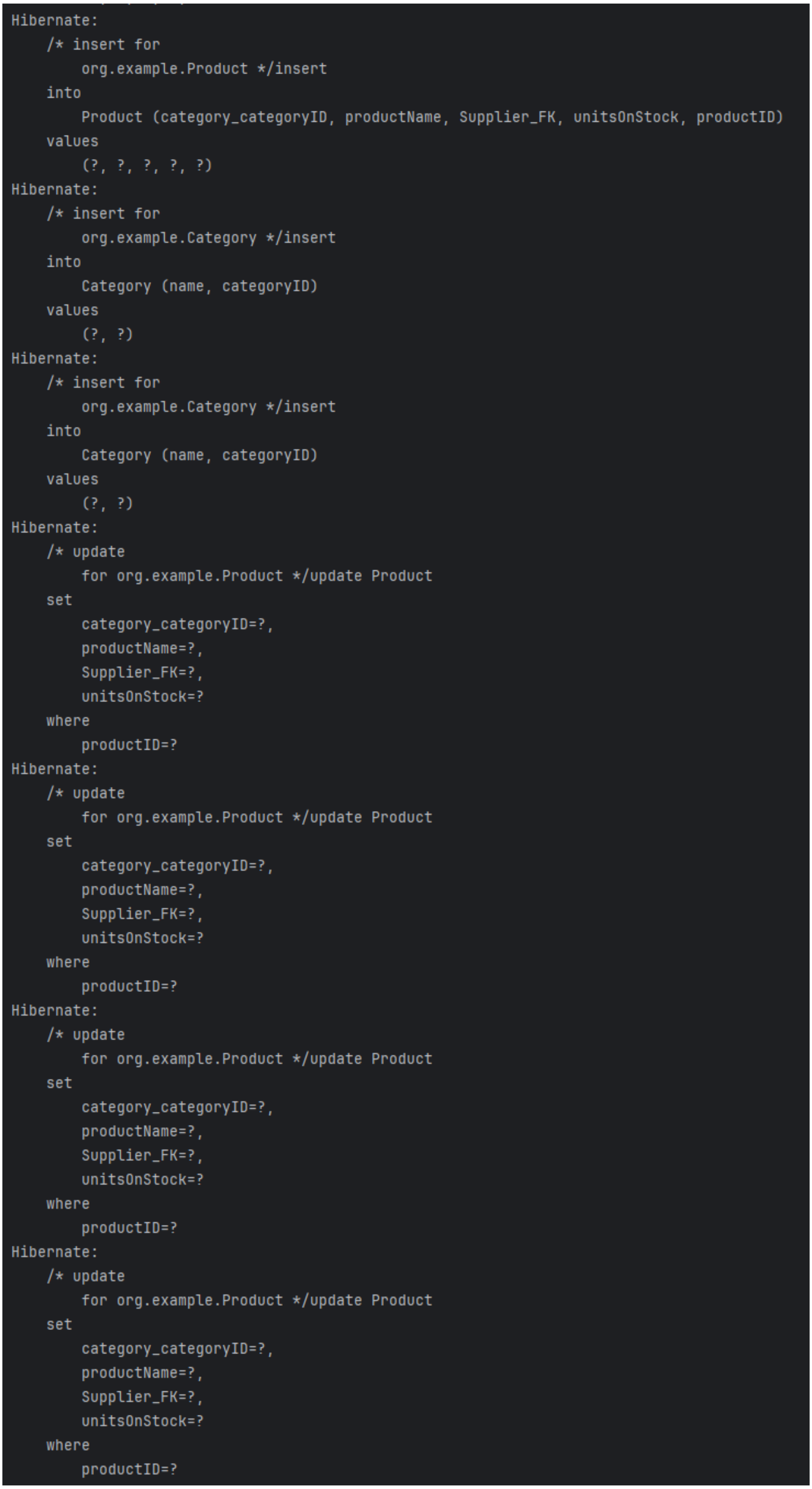
CATEGORY [MyLab5Database@127.0.0.1]

2 rows

WHERE

	CATEGORYID	NAME
1	1	Klikalne
2	2	Drogie

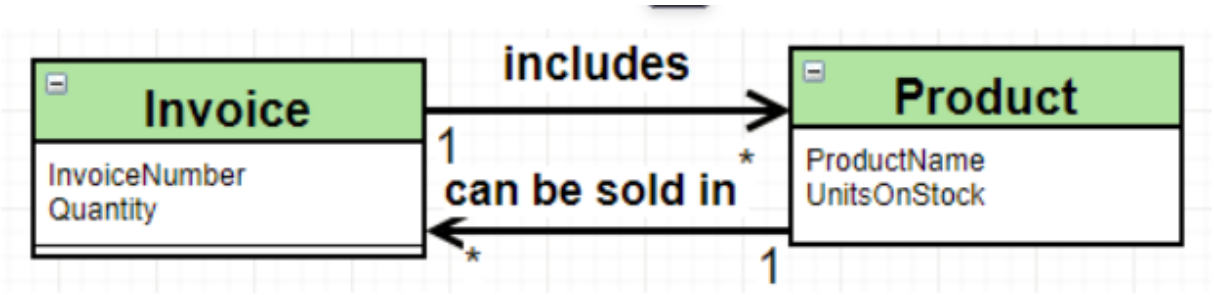




```
Hibernate:
    /* insert for
        org.example.Product */insert
    into
        Product (category_categoryID, productName, Supplier_FK, unitsOnStock, productID)
    values
        (?, ?, ?, ?, ?)
Hibernate:
    /* insert for
        org.example.Category */insert
    into
        Category (name,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

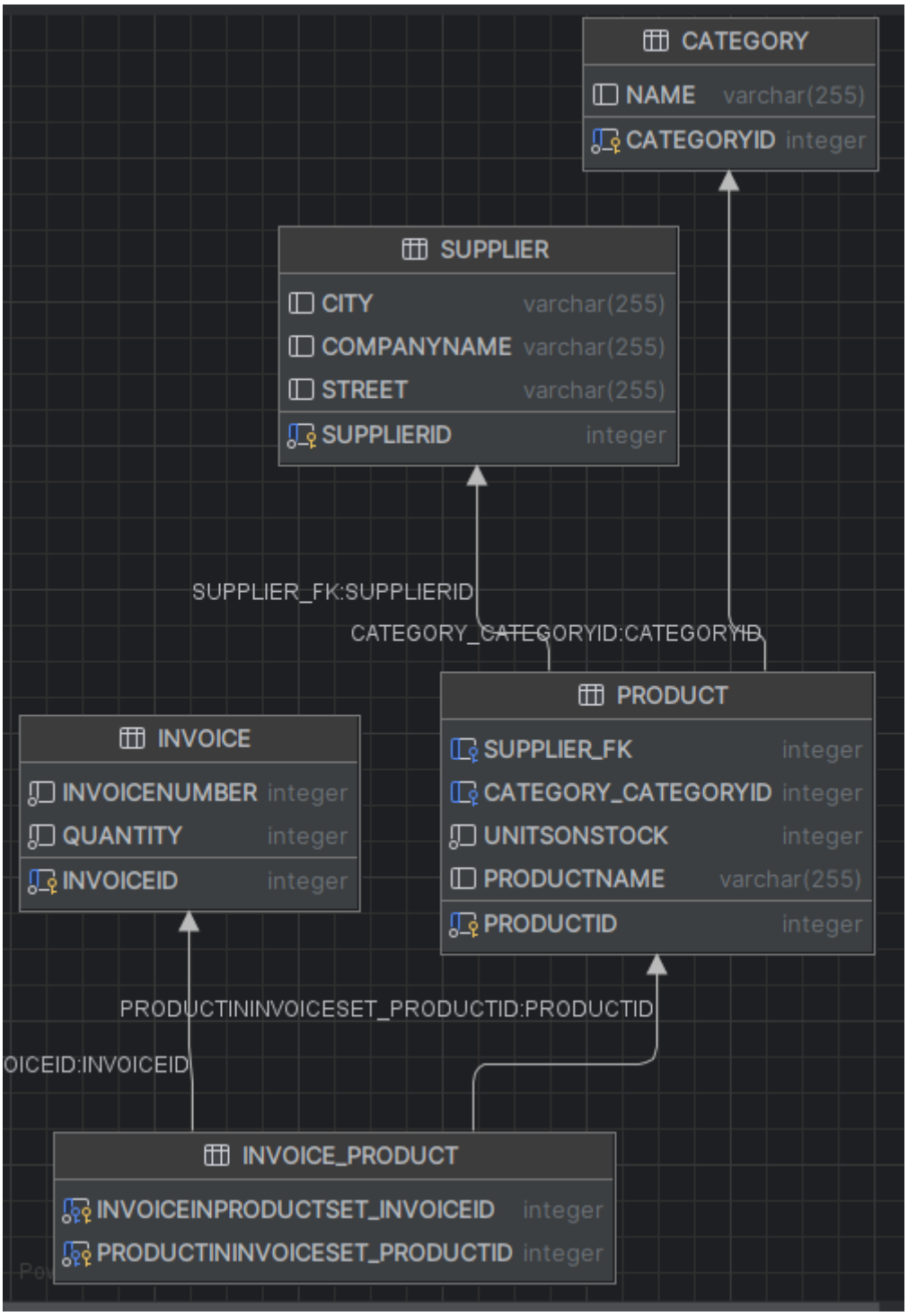
```
Hibernate:
    create table Product (
        Supplier_FK integer,
        category_categoryID integer,
        productID integer not null,
        unitsOnStock integer not null,
        productName varchar(255),
        primary key (productID)
    )
Hibernate:
    create table Supplier (
        supplierID integer not null,
        city varchar(255),
        companyName varchar(255),
        street varchar(255),
        primary key (supplierID)
    )
Hibernate:
    alter table Product
       add constraint FK987q0koesbyk7oqky7lg431xr
       foreign key (category_categoryID)
       references Category
Hibernate:
    alter table Product
       add constraint FKve96qacvsr1a50rgwl94enru
       foreign key (Supplier_FK)
       references Supplier
Hibernate:

values
    next value for Product_SEQ
Hibernate:

values
    next value for Product_SEQ
Hibernate:

values
    next value for Category_SEQ
Hibernate:

values
    next value for Category_SEQ
Product{productID=2, productName='Myszka', unitsOnStock=2}
Product{productID=2, productName='Myszka', unitsOnStock=2}
Product{productID=3, productName='Monitor', unitsOnStock=16}
Product{productID=3, productName='Monitor', unitsOnStock=16}
Hibernate:
    /* insert for
        org.example.Product */insert
    into
        Product (category_categoryID, productName, Supplier_FK, unitsOnStock, productID)
    values
        (?, ?, ?, ?, ?)
Hibernate:
    /* insert for
        org.example.Product */insert
    into
        Product (category_categoryID, productName, Supplier_FK, unitsOnStock, productID)
    values
        (?, ?, ?, ?, ?)
Hibernate:
    /* insert for
        org.example.Product */insert
    into
        Product (category_categoryID, productName, Supplier_FK, unitsOnStock, productID)
```

5.Zamodeluj relacje wiele-do-wielu, jak poniżej:



```java
public static void main(String[] args) throws InvalidAttributeValueException {
    sessionFactory = getSessionFactory();
    Session session = sessionFactory.openSession();
    Transaction tx = session.beginTransaction();
    Product product = new Product( productName: "Klawiatura",  unitsOnStock: 3);
    Product product1 = new Product( productName: "Myszka",  unitsOnStock: 2);
    Product product2 = new Product( productName: "Monitor",  unitsOnStock: 16);
    Product product3 = new Product( productName: "Jednostka Centralna",  unitsOnStock: 7);
    session.save(product);
    session.save(product1);
    session.save(product2);
    session.save(product3);
    Invoice invoice = new Invoice( invoiceNumber: 1);
    Invoice invoice1 = new Invoice( invoiceNumber: 2);
    session.save(invoice);
    session.save(invoice1);

    // sprzedajemy wszystko

    try {
        product.sell(invoice, quantity: 3);
        product1.sell(invoice1, quantity: 1);
        product2.sell(invoice1, quantity: 10);
        product3.sell(invoice, quantity: 5);
        product2.sell(invoice, quantity: 6);
    } catch (InvalidAttributeValueException invalidAttributeValueException){
        invalidAttributeValueException.printStackTrace();
    }
    tx.commit();

    // produkty sprzedane w ramach faktury nr.1
    List<Invoice> productsSold = findInvoicesByNumber( invoiceNumber: 1,session);
    for(Invoice invoicee:productsSold) {
        System.out.println(invoicee);
    }
    // faktury na których jest produkt2
    List<Invoice> invoicesByNumber = findInvoicesByNumber( invoiceNumber: 1,session);
    for(Invoice invoicee:invoicesByNumber) {
        System.out.println(invoicee);
    }



    session.close();
}

no usages
public static List<Invoice> findInvoicesWithProduct(int productId,Session session) {
    String hql = "SELECT i FROM Invoice i JOIN i.productInInvoiceSet p WHERE p.id = :productId";
    Query<Invoice> query = session.createQuery(hql, Invoice.class);
    query.setParameter( s: "productId", productId);
    return query.list();
}


2 usages
public static List<Invoice> findInvoicesByNumber(int invoiceNumber, Session session) {
    String hql = "FROM Invoice i WHERE i.invoiceNumber = :invoiceNumber";
    Query<Invoice> query = session.createQuery(hql, Invoice.class);
    query.setParameter( s: "invoiceNumber", invoiceNumber);
    return query.list();
}
```

```java
package org.example;

import jakarta.persistence.*;

import java.util.HashSet;
import java.util.Set;

20 usages
@Entity
public class Invoice {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int invoiceID;

    1 usage
    private int invoiceNumber;

    1 usage
    private int quantity;

    public Invoice() {
    }

    2 usages
    public Invoice(int invoiceNumber) { this.invoiceNumber = invoiceNumber; }

    1 usage
    @ManyToMany
    private Set<Product> productInInvoiceSet = new HashSet<>();

    1 usage
    public void addToInvoice(Product product, int numberOfNewProducts){
        productInInvoiceSet.add(product);
        quantity += numberOfNewProducts;
    }
}
```


CATEGORY
NAME varchar(255)
CATEGORYID integer
SUPPLIER
CITY varchar(255)
COMPANYNAME varchar(255)
STREET varchar(255)
SUPPLIERID integer
SUPPLIER_FK:SUPPLIERID
CATEGORY_CATEGORYID:CATEGORYID
PRODUCT
SUPPLIER_FK integer
CATEGORY_CATEGORYID integer
UNITSONSTOCK integer
PRODUCTNAME varchar(255)
PRODUCTID integer
INVOICE
INVOICENUMBER integer
QUANTITY integer
INVOICEID integer
PRODUCTININVOICESET_PRODUCTID:PRODUCTID
OICEID:INVOICEID
INVOICE_PRODUCT
INVOICEINPRODUCTSET_INVOICEID integer
PRODUCTININVOICESET_PRODUCTID integer

INVOICE_PRODUCT [MyLab6Database@127.0.0.1]

5 rows

WHERE

	INVOICEINP...	PRODUCTINI...
1	1	1
2	1	3
3	1	4
4	2	2
5	2	3

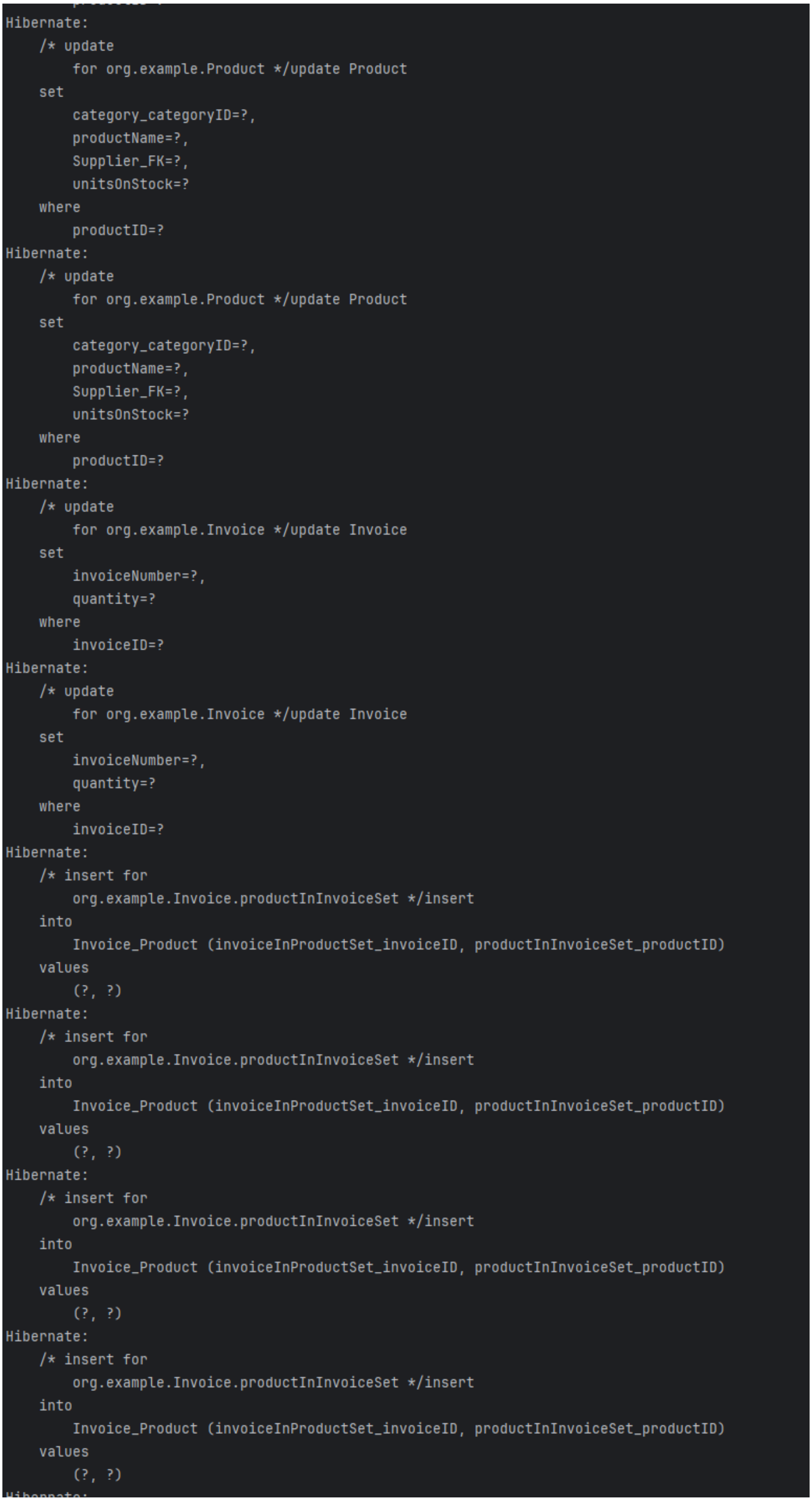
INVOICE_PRODUCT [MyLab6Database@127.0.0.1] INVOICE [MyLab6Database@127.0.0.1]

2 rows Tx: Auto DDL

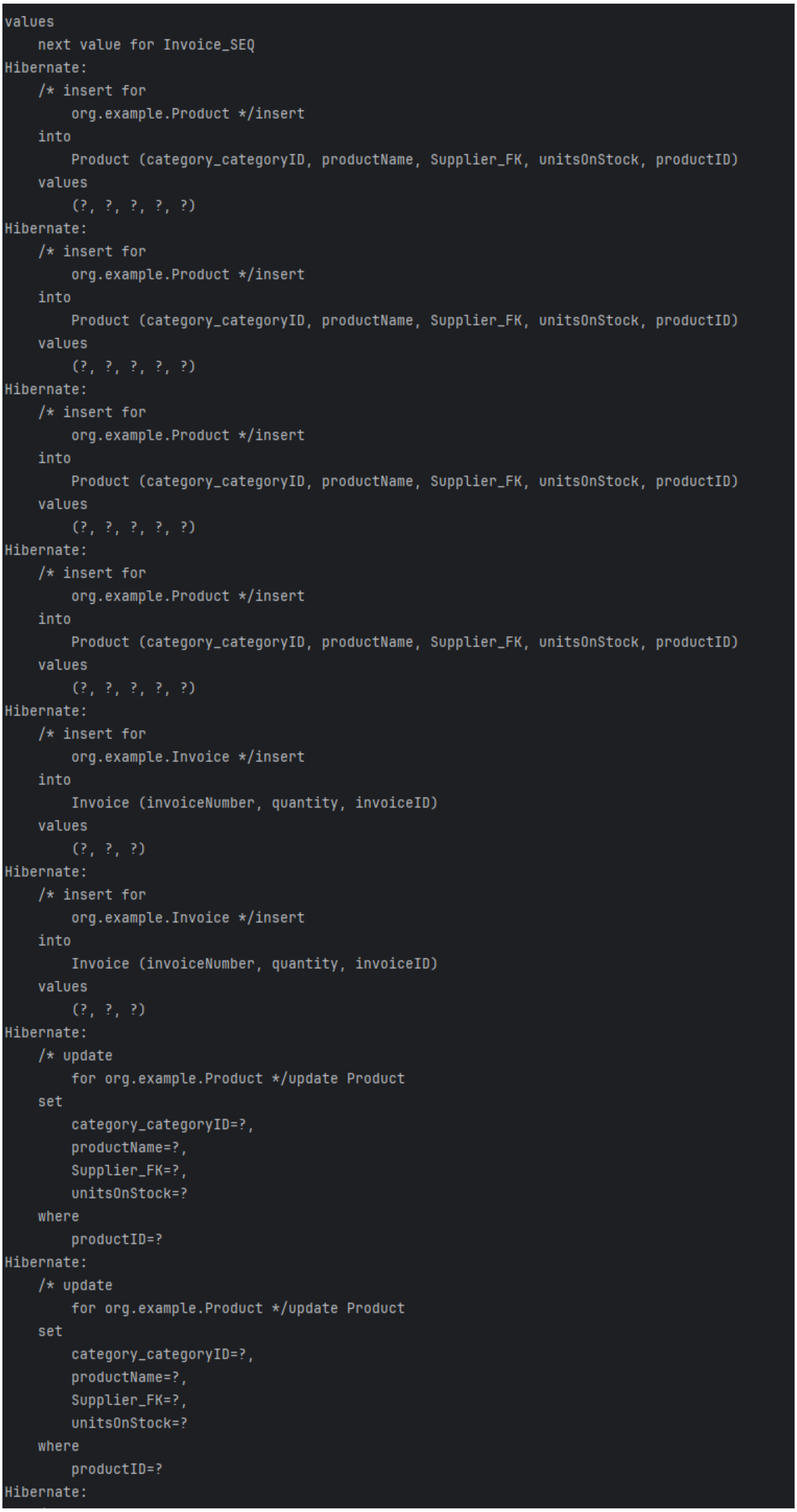
WHERE ORDER BY

	INVOICEID	INVOICENUMBER	QUANTITY
1	1	1	14
2	2	2	11

```
        (?, ?)
Hibernate:
    /*
FROM
    Invoice i
WHERE
    i.invoiceNumber = :invoiceNumber */ select
        i1_0.invoiceID,
        i1_0.invoiceNumber,
        i1_0.quantity
    from
        Invoice i1_0
    where
        i1_0.invoiceNumber=?
org.example.Invoice@59303963
Hibernate:
    /*
FROM
    Invoice i
WHERE
    i.invoiceNumber = :invoiceNumber */ select
        i1_0.invoiceID,
        i1_0.invoiceNumber,
        i1_0.quantity
    from
        Invoice i1_0
    where
        i1_0.invoiceNumber=?
org.example.Invoice@59303963
```



```
Hibernate: 
    /* update
        for org.example.Product */update Product 
    set
        category_categoryID=?,
        productName=?,
        Supplier_FK=?,
        unitsOnStock=? 
    where
        productID=?
Hibernate: 
    /* update
        for org.example.Product */update Product 
    set
        category_categoryID=?,
        productName=?,
        Supplier_FK=?,
        unitsOnStock=? 
    where
        productID=?
Hibernate: 
    /* update
        for org.example.Invoice */update Invoice 
    set
        invoiceNumber=?,
        quantity=? 
    where
        invoiceID=?
Hibernate: 
    /* update
        for org.example.Invoice */update Invoice 
    set
        invoiceNumber=?,
        quantity=? 
    where
        invoiceID=?
Hibernate: 
    /* insert for
        org.example.Invoice.productInInvoiceSet */insert 
    into
        Invoice_Product (invoiceInProductSet_invoiceID, productInInvoiceSet_productID) 
    values
        (?, ?)
Hibernate: 
    /* insert for
        org.example.Invoice.productInInvoiceSet */insert 
    into
        Invoice_Product (invoiceInProductSet_invoiceID, productInInvoiceSet_productID) 
    values
        (?, ?)
Hibernate: 
    /* insert for
        org.example.Invoice.productInInvoiceSet */insert 
    into
        Invoice_Product (invoiceInProductSet_invoiceID, productInInvoiceSet_productID) 
    values
        (?, ?)
Hibernate: 
    /* insert for
        org.example.Invoice.productInInvoiceSet */insert 
    into
        Invoice_Product (invoiceInProductSet_invoiceID, productInInvoiceSet_productID) 
    values
        (?, ?)
```

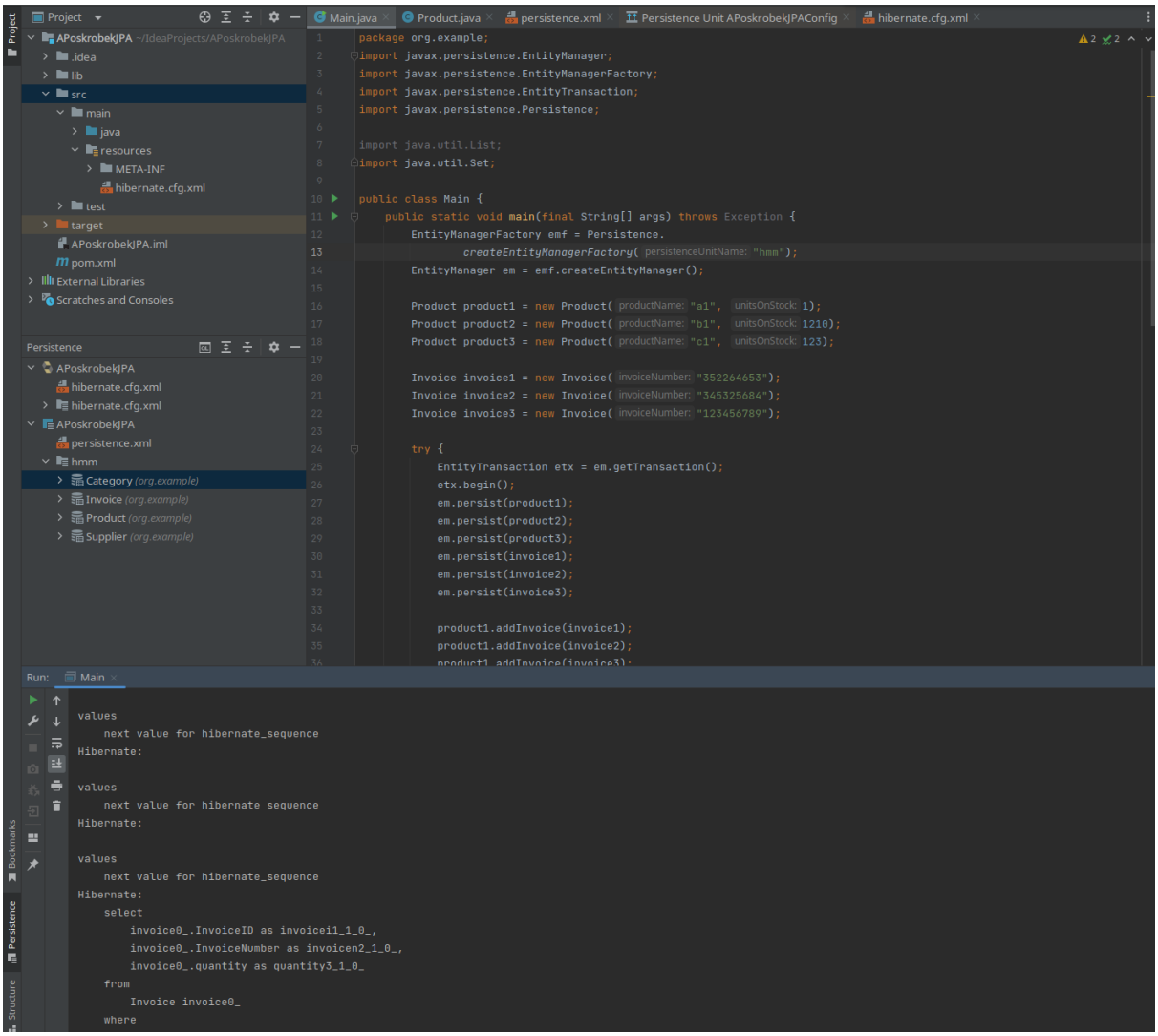


```
values
    next value for Invoice_SEQ
Hibernate:
    /* insert for
        org.example.Product */insert
    into
        Product (category_categoryID, productName, Supplier_FK, unitsOnStock, productID)
    values
        (?, ?, ?, ?, ?)
Hibernate:
    /* insert for
        org.example.Product */insert
    into
        Product (category_categoryID, productName, Supplier_FK, unitsOnStock, productID)
    values
        (?, ?, ?, ?, ?)
Hibernate:
    /* insert for
        org.example.Product */insert
    into
        Product (category_categoryID, productName, Supplier_FK, unitsOnStock, productID)
    values
        (?, ?, ?, ?, ?)
Hibernate:
    /* insert for
        org.example.Product */insert
    into
        Product (category_categoryID, productName, Supplier_FK, unitsOnStock, productID)
    values
        (?, ?, ?, ?, ?)
Hibernate:
    /* insert for
        org.example.Invoice */insert
    into
        Invoice (invoiceNumber, quantity, invoiceID)
    values
        (?, ?, ?)
Hibernate:
    /* insert for
        org.example.Invoice */insert
    into
        Invoice (invoiceNumber, quantity, invoiceID)
    values
        (?, ?, ?)
Hibernate:
    /* update
        for org.example.Product */update Product
    set
        category_categoryID=?,
        productName=?,
        Supplier_FK=?,
        unitsOnStock=?
    where
        productID=?
Hibernate:
    /* update
        for org.example.Product */update Product
    set
        category_categoryID=?,
        productName=?,
        Supplier_FK=?,
        unitsOnStock=?
    where
        productID=?
Hibernate:
```

```
Hibernate:
    create table Invoice (
        invoiceID integer not null,
        invoiceNumber integer not null,
        quantity integer not null,
        primary key (invoiceID)
    )
Hibernate:
    create table Invoice_Product (
        invoiceInProductSet_invoiceID integer not null,
        productInInvoiceSet_productID integer not null,
        primary key (invoiceInProductSet_invoiceID, productInInvoiceSet_productID)
    )
Hibernate:
    create table Product (
        Supplier_FK integer,
        category_categoryID integer,
        productID integer not null,
        unitsOnStock integer not null,
        productName varchar(255),
        primary key (productID)
    )
Hibernate:
    create table Supplier (
        supplierID integer not null,
        city varchar(255),
        companyName varchar(255),
        street varchar(255),
        primary key (supplierID)
    )
Hibernate:
    alter table Invoice_Product
       add constraint FKevshobpm07fskoqngeo3xgnh0
       foreign key (productInInvoiceSet_productID)
       references Product
Hibernate:
    alter table Invoice_Product
       add constraint FK89bhrkfde7au2em9d3hj7rj4y
       foreign key (invoiceInProductSet_invoiceID)
       references Invoice
Hibernate:
    alter table Product
       add constraint FK987q0koesbyk7oqky7lg431xr
       foreign key (category_categoryID)
       references Category
Hibernate:
    alter table Product
       add constraint FKve96qacvsr1a50rgwl94enru
       foreign key (Supplier_FK)
       references Supplier
Hibernate:

values
    next value for Product_SEQ
Hibernate:

values
    next value for Product_SEQ
Hibernate:

values
    next value for Invoice_SEQ
Hibernate:
```

```
Hibernate:
    alter table Invoice_Product
       drop constraint FK89bhrkfde7au2em9d3hj7rj4y
Hibernate:
    alter table Product
       drop constraint FK987q0koesbyk7oqky7lg431xr
Hibernate:
    alter table Product
       drop constraint FKve96qacvsr1a50rgwl94enru
Hibernate:
    drop table Category
Hibernate:
    drop table Invoice
Hibernate:
    drop table Invoice_Product
Hibernate:
    drop table Product
Hibernate:
    drop table Supplier
Hibernate:
    drop sequence Category_SEQ restrict
Hibernate:
    drop sequence Invoice_SEQ restrict
Hibernate:
    drop sequence Product_SEQ restrict
Hibernate:
    drop sequence Supplier_SEQ restrict
Hibernate:
    create sequence Category_SEQ start with 1 increment by 50
Hibernate:
    create sequence Invoice_SEQ start with 1 increment by 50
Hibernate:
    create sequence Product_SEQ start with 1 increment by 50
Hibernate:
    create sequence Supplier_SEQ start with 1 increment by 50
Hibernate:
    create table Category (
        categoryID integer not null,
        name varchar(255),
        primary key (categoryID)
    )
```

6.JPA


Project
APoskrobekJPA ~/IdeaProjects/APoskrobekJPA
.idea
lib
src
main
java
resources
META-INF
hibernate.cfg.xml
test
target
APoskrobekJPA.iml
pom.xml
External Libraries
Scratches and Consoles
Persistence
APoskrobekJPA
hibernate.cfg.xml
hibernate.cfg.xml
APoskrobekJPA
persistence.xml
hmm
Category (org.example)
Invoice (org.example)
Product (org.example)
Supplier (org.example)
Main.java
Product.java
persistence.xml
Persistence Unit APoskrobekJPAConfig
hibernate.cfg.xml
package org.example;
import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.EntityTransaction;
import javax.persistence.Persistence;
import java.util.List;
import java.util.Set;
public class Main {
public static void main(final String[] args) throws Exception {
EntityManagerFactory emf = Persistence.
createEntityManagerFactory(persistenceUnitName: "hmm");
EntityManager em = emf.createEntityManager();
Product product1 = new Product(productName: "a1", unitsOnStock: 1);
Product product2 = new Product(productName: "b1", unitsOnStock: 1210);
Product product3 = new Product(productName: "c1", unitsOnStock: 123);
Invoice invoice1 = new Invoice(invoiceNumber: "352264653");
Invoice invoice2 = new Invoice(invoiceNumber: "345325684");
Invoice invoice3 = new Invoice(invoiceNumber: "123456789");
try {
EntityTransaction etx = em.getTransaction();
etx.begin();
em.persist(product1);
em.persist(product2);
em.persist(product3);
em.persist(invoice1);
em.persist(invoice2);
em.persist(invoice3);
product1.addInvoice(invoice1);
product1.addInvoice(invoice2);
Run: Main
values
next value for hibernate_sequence
Hibernate:
values
next value for hibernate_sequence
Hibernate:
values
next value for hibernate_sequence
Hibernate:
select
invoice0_.InvoiceID as invoicei1_1_0_,
invoice0_.InvoiceNumber as invoicen2_1_0_,
invoice0_.quantity as quantity3_1_0_
from
Invoice invoice0_
where

Main.java × Product.java × persistence.xml × Persistence Unit APoskrobekJPAConfig × hibernate.cfg.xml ×

```xml
<?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="hmm"
                    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://127.0.0.1/APoskrobekJPA"/>
      <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>
```

Main.java × Product.java × persistence.xml × Persistence Unit APoskrobekJPAConfig × hibernate.cfg.xml ×

```xml
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
        "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
    <session-factory>
        <property
                name="connection.driver_class">org.apache.derby.jdbc.ClientDriver</property>
        <property name="connection.url">jdbc:derby://127.0.0.1/APoskrobekJPA</property>
        <property name="show_sql">true</property>
        <property name="hbm2ddl.auto">create</property>
        <mapping class="org.example.Product"></mapping>
        <mapping class="org.example.Category"></mapping>
        <mapping class="org.example.Supplier"></mapping>
        <mapping class="org.example.Invoice"></mapping>
    </session-factory>
</hibernate-configuration>
```

7. Kaskady:

Main.java | PRODUCT | INVOICE | PRODUCT_INVOICE | Product.java | Invoice

3 rows | Tx: Auto | DDL | CSV

WHERE | ORDER BY

	DBID	PRODUCTNAME	UNITSONSTOCK	CATEGORY_FK	SUPPLIER_FK
1	1	kaskada1	1	<null>	<null>
2	2	kaskada2	1210	<null>	<null>
3	3	kaskada3	123	<null>	<null>

Main.java | PRODUCT | INVOICE | PRODUCT_INVOICE | Product.java | Invoice

4 rows | Tx: Auto | DDL | CSV

WHERE | ORDER BY

	PRODUCTSSOLD_DBID	INVOICES_INVOICEID
1	1	4
2	1	5
3	1	6
4	2	4

```
3 usages
@ManyToMany(cascade = CascadeType.PERSIST)
private Set<Invoice> invoices;
3 usages
public Product(String productName, int unitsOnStock) {
    this.productName = productName;
    this.unitsOnStock = units
    invoices = new HashSet<In
}
```

String productName
APoskrobekJPA

```
3 usages
@ManyToMany(mappedBy = "invoices", cascade = CascadeType.PERSIST)
private Set<Product> productsSold;

3 usages
public Invoice(String invoiceNumber) {
    InvoiceNumber = invoiceNumber;
    this.quantity = 0;
    productsSold = new HashSet<Product>();
```

Project

- APoskrobekJPA ~/IdeaProjects/APoskrobekJPA
  - .idea
  - lib
  - src
    - main
      - java
        - org.example
          - Category
          - Invoice
          - Main
          - Product
          - Supplier
      - resources
        - META-INF
        - hibernate.cfg.xml
    - test
  - target

Persistence

- APoskrobekJPA
  - hibernate.cfg.xml
  - hibernate.cfg.xml
- APoskrobekJPA
  - persistence.xml
  - hmm
    - Category (org.example)
    - Invoice (org.example)
    - Product (org.example)
    - Supplier (org.example)

Main.java | Product.java | Invoice.java

```java
public class Main {
    public static void main(final String[] args) throws Exception {
        EntityManagerFactory emf = Persistence.
                createEntityManagerFactory( persistenceUnitName: "hmm");
        EntityManager em = emf.createEntityManager();

        Product product1 = new Product( productName: "kaskada1",  unitsOnStock: 1);
        Product product2 = new Product( productName: "kaskada2",  unitsOnStock: 1210);
        Product product3 = new Product( productName: "kaskada3",  unitsOnStock: 123);

        Invoice invoice1 = new Invoice( invoiceNumber: "kaskadaFaktura1");
        Invoice invoice2 = new Invoice( invoiceNumber: "kaskadaFaktura2");
        Invoice invoice3 = new Invoice( invoiceNumber: "kaskadaFaktura3");

        try {
            EntityTransaction etx = em.getTransaction();
            etx.begin();
            em.persist(product1);
            em.persist(product2);
            em.persist(product3);
//            em.persist(invoice1);
//            em.persist(invoice2);
//            em.persist(invoice3);

            product1.addInvoice(invoice1);
            product1.addInvoice(invoice2);
            product1.addInvoice(invoice3);
            invoice1.addProduct(product1);
            invoice2.addProduct(product1);
            invoice3.addProduct(product1);

            product2.addInvoice(invoice2);
            invoice2.addProduct(product2);

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

Run: Main

```
Hibernate:
    insert
    into
        Product_Invoice
        (productsSold_dbID, invoices_InvoiceID)
    values
        (?, ?)
Hibernate:
    insert
    into
        Product_Invoice
        (productsSold_dbID, invoices_InvoiceID)
    values
        (?, ?)

Process finished with exit code 0
```

Project

- APoskrobekJPA ~/IdeaProjects/APoskrobekJPA
  - .idea
  - lib
  - src
    - main
      - java
        - org.example
          - Category
          - Invoice
          - Main
          - Product
          - Supplier
      - resources
        - META-INF
        - hibernate.cfg.xml
    - test
  - target

Persistence

- APoskrobekJPA
  - hibernate.cfg.xml
  - hibernate.cfg.xml
- APoskrobekJPA
  - persistence.xml
  - hmm
    - Category (org.example)
    - Invoice (org.example)
    - Product (org.example)
    - Supplier (org.example)

Main.java | Product.java | Invoice.java

```java
public class Main {
    public static void main(final String[] args) throws Exception {
        EntityManagerFactory emf = Persistence.
                createEntityManagerFactory( persistenceUnitName: "hmm");
        EntityManager em = emf.createEntityManager();

        Product product1 = new Product( productName: "kaskada1",  unitsOnStock: 1);
        Product product2 = new Product( productName: "kaskada2",  unitsOnStock: 1210);
        Product product3 = new Product( productName: "kaskada3",  unitsOnStock: 123);

        Invoice invoice1 = new Invoice( invoiceNumber: "kaskadaFaktura1");
        Invoice invoice2 = new Invoice( invoiceNumber: "kaskadaFaktura2");
        Invoice invoice3 = new Invoice( invoiceNumber: "kaskadaFaktura3");

        try {
            EntityTransaction etx = em.getTransaction();
            etx.begin();
//            em.persist(product1);
//            em.persist(product2);
//            em.persist(product3);
            em.persist(invoice1);
            em.persist(invoice2);
            em.persist(invoice3);

            product1.addInvoice(invoice1);
            product1.addInvoice(invoice2);
            product1.addInvoice(invoice3);
            invoice1.addProduct(product1);
            invoice2.addProduct(product1);
            invoice3.addProduct(product1);

            product2.addInvoice(invoice2);
            invoice2.addProduct(product2);

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

Run: Main

```
Hibernate:
    insert
    into
        Product_Invoice
        (productsSold_dbID, invoices_InvoiceID)
    values
        (?, ?)
Hibernate:
    insert
    into
        Product_Invoice
        (productsSold_dbID, invoices_InvoiceID)
    values
        (?, ?)

Process finished with exit code 0
```

8.Embeded

8.1 Dodanie do modelu klasy adres i wbudowanie jej do tabeli Dostawców

Adress.java | Main.java | SUPPLIER | Supplier.java

1 row | Tx: Auto | DDL | CSV

WHERE | ORDER BY

	DBID	CITY	STREET	ZIPCODE	COMPANYNAME
1	1	bbb	aaaa	ccc	tralala

Adress.java | Main.java | SUPPLIER | Supplier.java

```java
7 usages
@Entity
public class Supplier {
    @Id
    @GeneratedValue(
            strategy = GenerationType.AUTO)
    private int dbID;
    1 usage
    private String companyName;
    2 usages
    @OneToMany(mappedBy = "supplier")
    private Set<Product> productsGroup;

    1 usage
    @Embedded
    private Adress address;

    1 usage
    public Supplier(String companyName) {
        this.companyName = companyName;
        this.productsGroup = new HashSet<>();
    }

    public Supplier(){

    }

    public void setProduct(Product product) { productsGroup.add(product); }

    1 usage
    public void setAddress(Adress address){
        this.address = address;
    }
}
```

Adress.java | Main.java | SUPPLIER | Supplier.java

```java
package org.example;

import javax.persistence.Embeddable;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

4 usages
@Embeddable
public class Adress {
    1 usage
    private String street;
    1 usage
    private String city;
    1 usage
    private String zipCode;

    1 usage
    public Adress(String street, String city, String zipCode) {
        this.street = street;
        this.city = city;
        this.zipCode = zipCode;
    }

    public Adress() {
    }
}
```

Adress.java | Main.java | SUPPLIER | Supplier.java

```java
package org.example;
import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.EntityTransaction;
import javax.persistence.Persistence;

import java.util.List;
import java.util.Set;

public class Main {
    public static void main(final String[] args) throws Exception {
        EntityManagerFactory emf = Persistence.
                createEntityManagerFactory( persistenceUnitName: "hmm");
        EntityManager em = emf.createEntityManager();

        Supplier supplier = new Supplier( companyName: "tralala");
        Adress address = new Adress( street: "aaaa",  city: "bbb",  zipCode: "ccc");

        try {
            EntityTransaction etx = em.getTransaction();
            etx.begin();
            supplier.setAddress(address);
            em.persist(supplier);

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

8.2. Zmodyfikowanie modelu w taki sposób, że dane adresowe znajdują się w klasie dostawców. Zmapowane do dwóch osobnych tabel

Supplier.java | Main.java | ADDRESS_TBL | SUPPLIER | Adre

1 row | Tx: Auto | DDL

WHERE | ORDER BY

	CITY	STREET	ZIPCODE	DBID
1	bbb	aaaa	ccc	1

Supplier.java | Main.java | ADDRESS_TBL | SUPPLIER | Adre

1 row | Tx: Auto | DDL

WHERE | ORDER BY

	DBID	COMPANYNAME
1	1	parampampam

Supplier.java | Main.java | Adress.java

```java
7 usages
@Entity
@SecondaryTable(name="ADDRESS_TBL")
public class Supplier {
    @Id
    @GeneratedValue(
            strategy = GenerationType.AUTO)
    private int dbID;
    1 usage
    private String companyName;
    1 usage
    @Column(table = "ADDRESS_TBL")
    private String street;
    1 usage
    @Column(table = "ADDRESS_TBL")
    private String city;
    1 usage
    @Column(table = "ADDRESS_TBL")
    private String zipCode;
    1 usage
    @OneToMany(mappedBy = "supplier")
    private Set<Product> productsGroup;

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

    public Supplier(){

    }

    public void setProduct(Product product) { productsGroup.add(product); }

}
```

Supplier.java | Main.java | Adress.java

```java
package org.example;
import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.EntityTransaction;
import javax.persistence.Persistence;

import java.util.List;
import java.util.Set;

public class Main {
    public static void main(final String[] args) throws Exception {
        EntityManagerFactory emf = Persistence.
                createEntityManagerFactory( persistenceUnitName: "hmm");
        EntityManager em = emf.createEntityManager();

        Supplier supplier = new Supplier( companyName: "parampampam",  street: "aaaa",  city: "bbb",  zipC

        try {
            EntityTransaction etx = em.getTransaction();
            etx.begin();
            em.persist(supplier);

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

9 Dziedziczenie

9.1 Strategia Single-table

Supplier.java | Customer.java | Company.java | Main.java | COMPANY

2 rows | Tx: Auto | DDL | CSV

WHERE | ORDER BY

	COMPANY_TYPE	DBID	COMPANYNAME	BANKACCOUNT	DISCOUNT
1	supplier	1	Dziedziczenie	12421423423534534	<null>
2	customer	2	Dziedziczenie	<null>	10

```java
public class Main {
    public static void main(final String[] args) throws Exception {
        EntityManagerFactory emf = Persistence.
                createEntityManagerFactory( persistenceUnitName: "hmm");
        EntityManager em = emf.createEntityManager();

        Supplier supplier = new Supplier( bankAccount: "12421423423534534", companyName: "Dziedziczenie",  street: "aaaa",  city: "bbb
        Customer customer = new Customer( discount: 10, companyName: "Dziedziczenie",  street: "aaaa",  city: "bbb",  zipCode: "ccc");

        try {
            EntityTransaction etx = em.getTransaction();
            etx.begin();
            em.persist(supplier);
            em.persist(customer);

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

```
2 usages   2 inheritors
@Entity(name="company")
@Inheritance(strategy = InheritanceType.SINGLE_TABLE)
@DiscriminatorColumn(name="company_type")
@SecondaryTable(name="ADDRESS_TBL")
public abstract class Company {
    @Id
    @GeneratedValue(
            strategy = GenerationType.AUTO)
    private int dbID;
    1 usage
    private String companyName;
    1 usage
    @Column(table = "ADDRESS_TBL")
    private String street;
    1 usage
    @Column(table = "ADDRESS_TBL")
    private String city;
    1 usage
    @Column(table = "ADDRESS_TBL")
    private String zipCode;

    1 usage
    @OneToMany(mappedBy = "supplier")
    private Set<Product> productsGroup;

    2 usages
    public Company(String companyName, String street, String city, String zipCode) {
        this.companyName = companyName;
        this.street = street;
        this.city = city;
        this.zipCode = zipCode;
    }

    2 usages
    public Company() {
    }

    public void setProduct(Product product) {
        productsGroup.add(product);
    }

}
```

```
import javax.persistence.DiscriminatorValue;
import javax.persistence.Entity;

2 usages
@Entity
@DiscriminatorValue(value="customer")
public class Customer extends Company{
    1 usage
    private int discount;

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

    }
}
```

```java
import ...

7 usages
@Entity
@DiscriminatorValue(value="supplier")
public class Supplier extends Company{
    1 usage
    private String bankAccount;

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

    }
}
```

9.2 Strategia Joined Subclass

Company.java | Main.java | SUPPLIER

1 row

WHERE | ORDER BY

	BANKACCOUNT	SUPPLIER
1	12421423423534534	1

CUSTOMER

1 row

WHERE | ORDER BY

	DISCOUNT	CUSTOMER
1	10	2

COMPANY

2 rows

WHERE | ORDER BY

	DBID	COMPANYNAME
1	1	Dziedziczenie
2	2	Dziedziczenie

Supplier.java | Customer.java | Company.java | Main.java

```java
package org.example;

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

2 usages  2 inheritors
@Entity(name="company")
@Inheritance(strategy = InheritanceType.JOINED)
@SecondaryTable(name="ADDRESS_TBL")
public abstract class Company {
    @Id
    @GeneratedValue(
            strategy = GenerationType.AUTO)
    private int dbID;
    1 usage
    private String companyName;
    1 usage
    @Column(table = "ADDRESS_TBL")
    private String street;
    1 usage
    @Column(table = "ADDRESS_TBL")
    private String city;
    1 usage
    @Column(table = "ADDRESS_TBL")
    private String zipCode;

    1 usage
    @OneToMany(mappedBy = "supplier")
    private Set<Product> productsGroup;

    2 usages
    public Company(String companyName, String street, String city, String zipCode) {
```

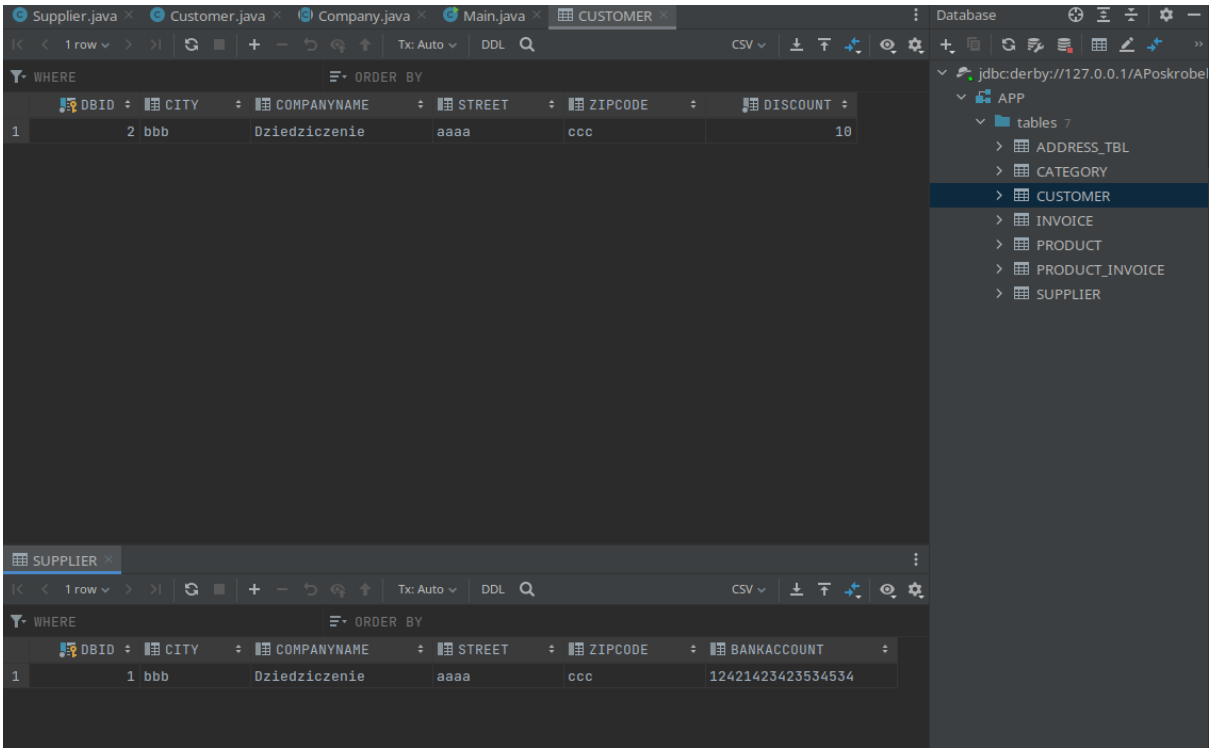
Supplier.java | Customer.java | Company.java | Main.java

```java
package org.example;

import javax.persistence.DiscriminatorValue;
import javax.persistence.Entity;
import javax.persistence.PrimaryKeyJoinColumn;

2 usages
@Entity
@PrimaryKeyJoinColumn(name="customer")
public class Customer extends Company{
    1 usage
    private int discount;

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

    }
}
```

Supplier.java | Customer.java | Company.java | Main.java

```java
package org.example;
import ...

7 usages
@Entity
@PrimaryKeyJoinColumn(name="supplier")
public class Supplier extends Company{
    1 usage
    private String bankAccount;

    1 usage
    public Supplier(String bankAccount, String companyName, String street, String city
        super(companyName, street, city, zipCode);
        this.bankAccount = bankAccount;
    }
    public Supplier(){
    }
}
```

9.3 Strategia Table per class



Supplier.java | Customer.java | Company.java | Main.java | CUSTOMER

```java
package org.example;

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

2 usages  2 inheritors
@Entity(name="company")
@Inheritance(strategy = InheritanceType.TABLE_PER_CLASS)
public abstract class Company {
    @Id
    @GeneratedValue(
            strategy = GenerationType.AUTO)
    private int dbID;
    1 usage
    private String companyName;
    1 usage
    private String street;
    1 usage
    private String city;
    1 usage
    private String zipCode;

    1 usage
    @OneToMany(mappedBy = "supplier")
    private Set<Product> productsGroup;
```

Supplier.java | Customer.java | Company.java | Main.java | CUSTOMER

```java
package org.example;

import javax.persistence.DiscriminatorValue;
import javax.persistence.Entity;
import javax.persistence.PrimaryKeyJoinColumn;

2 usages
@Entity
public class Customer extends Company{
    1 usage
    private int discount;

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

    }
}
```

Supplier.java | Customer.java | Company.java | Main.java | CUSTOMER

```java
package org.example;
import ...

7 usages
@Entity
public class Supplier extends Company{
    1 usage
    private String bankAccount;

    1 usage
    public Supplier(String bankAccount, String companyName, String street, String city, String zipCo
        super(companyName, street, city, zipCode);
        this.bankAccount = bankAccount;
    }
    public Supplier(){
    }
}
```