Hibernate, JPA - laboratorium

Aleksandra Mazur

II. Basics

a, b, c, d, e, f)

```
C:\WINDOWS\system32\cmd.exe
ij> connect 'jdbc:derby://127.0.0.1/AMazurJPA;create=true';
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
ij>
```

Konfiguracja środowiska powiodła się.

g, h) Klasa Product

Stworzono klasę Product z polami ProductName, UnitsOnStock i uzupełniono w klasie elementy potrzebne do zmapowania klasy do bazy danych.

Product.java

```java
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

@Entity
public class Product {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    public int ProductId;
    public String ProductName;
    public int UnitsOnStock;

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

    public Product() {
        // for Hibernate
    }
}
```

i, j) Hibernate config

W pliku *hibernate.cfg.xml* uzupełniono potrzebne property.

hibernate.cfg.xml

```xml
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD//EN"
        "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
    <session-factory>
        <property name="connection.url">jdbc:derby://127.0.0.1/AMazurJPA</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>
        <property name="hibernate.hbm2ddl.auto">update</property>
        <mapping class="Product"></mapping>
    </session-factory>
</hibernate-configuration>
```

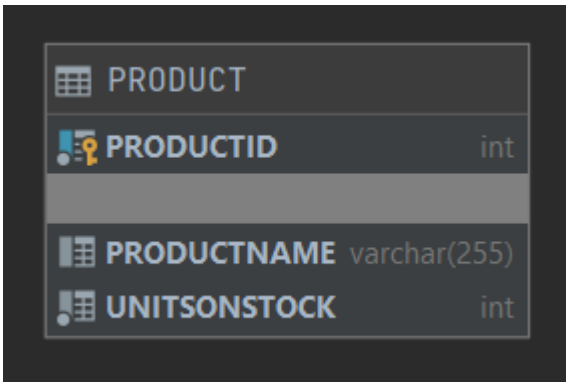
III. Przykładowy produkt

Klasa Main:

```java
Main.java
import java.util.Map;

public class Main {
    private static final SessionFactory ourSessionFactory;

    static {
        try {
            Configuration configuration = new Configuration();
            configuration.configure();

            ourSessionFactory = configuration.buildSessionFactory();
        } catch (Throwable ex) {
            throw new ExceptionInInitializerError(ex);
        }
    }

    public static Session getSession() throws HibernateException {
        return ourSessionFactory.openSession();
    }

    public static void main(final String[] args) throws Exception {
        Product product = new Product( productName: "Book",  unitsOnStock: 5);
        final Session session = getSession();
        Transaction transaction = session.beginTransaction();
        session.save(product);
        transaction.commit();
        try {
            System.out.println("querying all the managed entities...");
            final Metamodel metamodel = session.getSessionFactory().getMetamodel();
            for (EntityType<?> entityType : metamodel.getEntities()) {
                final String entityName = entityType.getName();
                final Query query = session.createQuery( s: "from " + entityName);
                System.out.println("executing: " + query.getQueryString());
                for (Object o : query.list()) {
                    System.out.println("  " + o);
                }
            }
        } finally {
            session.close();
        }
    }
}
```

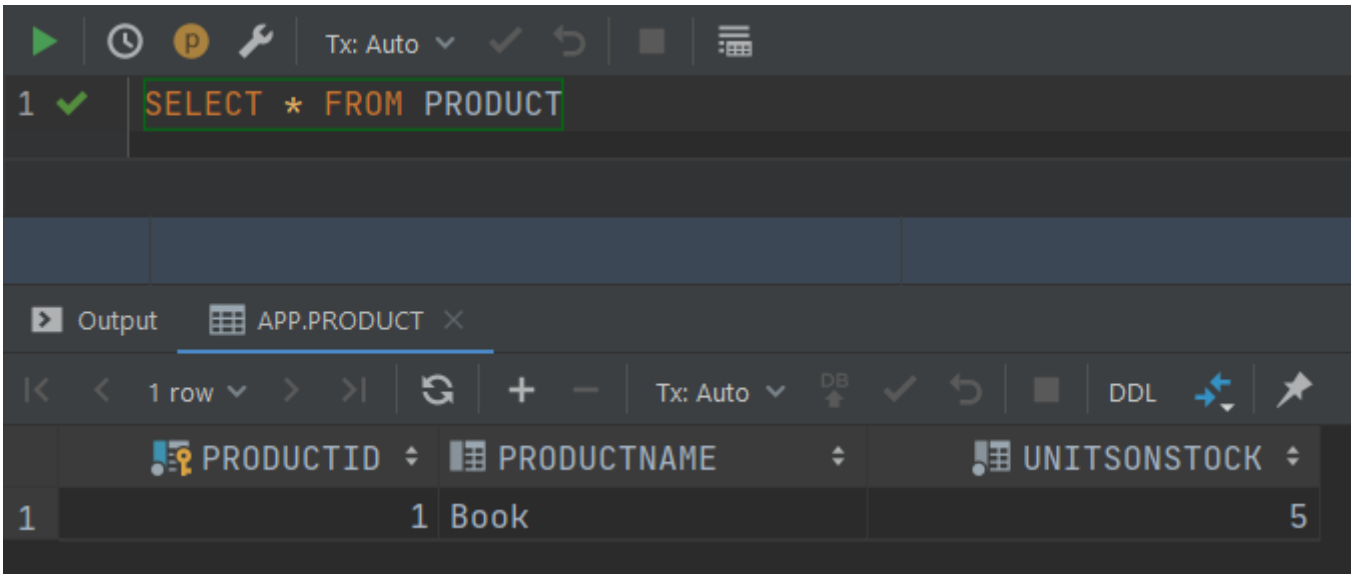
```
Main
kwi 28, 2020 8:56:12 AM org.hibernate.engine.transacti
INFO: HHH000490: Using JtaPlatform implementation: [or
Hibernate:

values
    next value for hibernate_sequence
Hibernate:
    /* insert Product
        */ insert
        into
            Product
            (ProductName, UnitsOnStock, ProductId)
        values
            (?, ?, ?)
querying all the managed entities...
executing: from Product
Hibernate:
    /*
from
    Product */ select
        product0_.ProductId as producti1_0_,
        product0_.ProductName as productn2_0_,
        product0_.UnitsOnStock as unitsons3_0_
    from
        Product product0_
  Product@151bf776

Process finished with exit code 0
```

Schemat w bazie danych:




```
create table PRODUCT
(
    PRODUCTID    INTEGER not null
        constraint "SQL0000000081-120fc126-0171-bf75-6e31-ffffc71c8945"
            primary key,
    PRODUCTNAME  VARCHAR(255),
    UNITSONSTOCK INTEGER not null
);
```

Jak widać produkt dodał się poprawnie:



IV. Klasa Supplier

Main.java Supplier.java

```java
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

@Entity
public class Supplier {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    public int SupplierId;
    public String CompanyName;
    public String Street;
    public String City;

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

    public Supplier() {
    }
}
```

Zmodyfikowano klasę Product, dodając do niej pole Supplier.

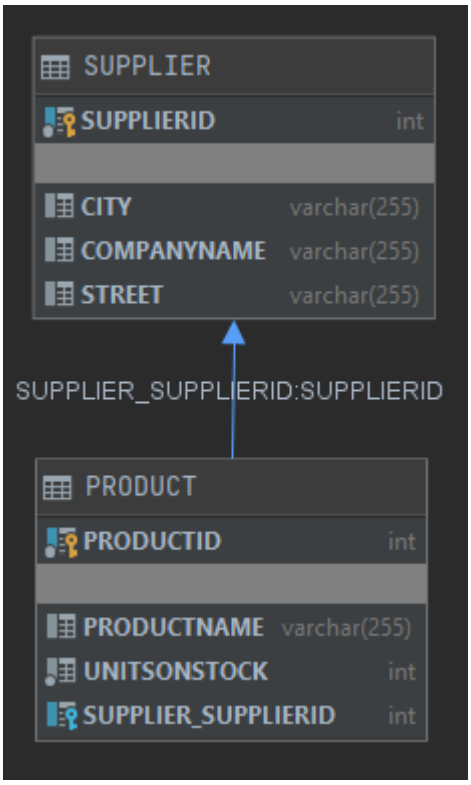
Main.java × Supplier.java × Product.java ×

```java
public class Product {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    public int ProductId;
    public String ProductName;
    public int UnitsOnStock;

    @ManyToOne
    public Supplier supplier;

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

    public Product() {
        // for Hibernate
    }

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

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

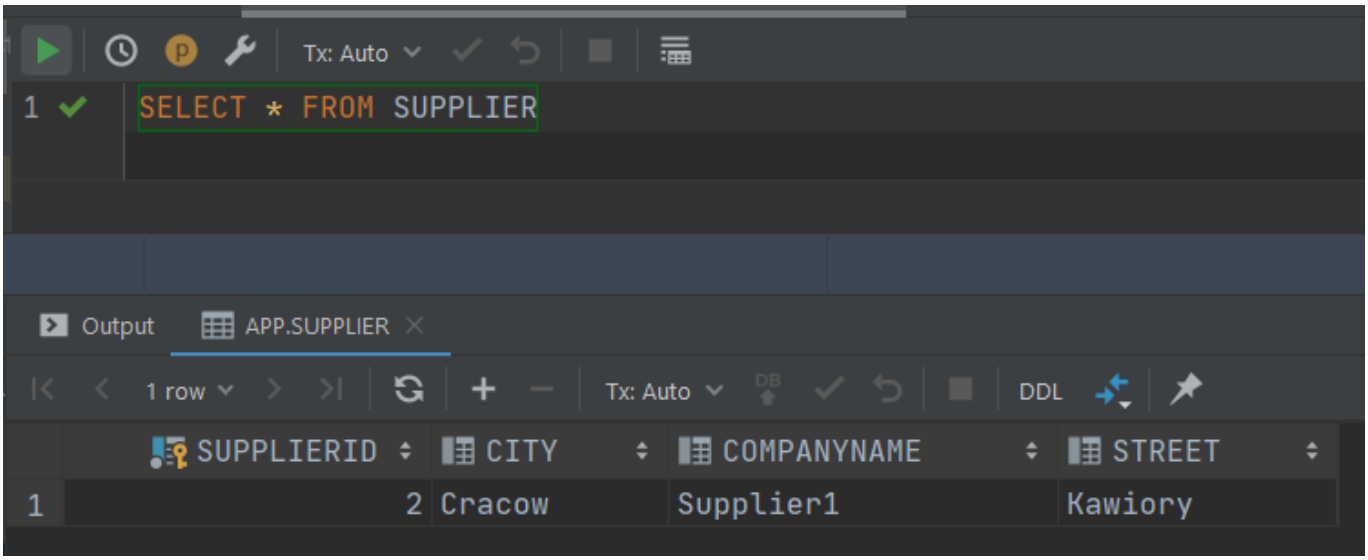
Utworzono nowego dostawcę i przypisano go do wcześniej utworzonego produktu.

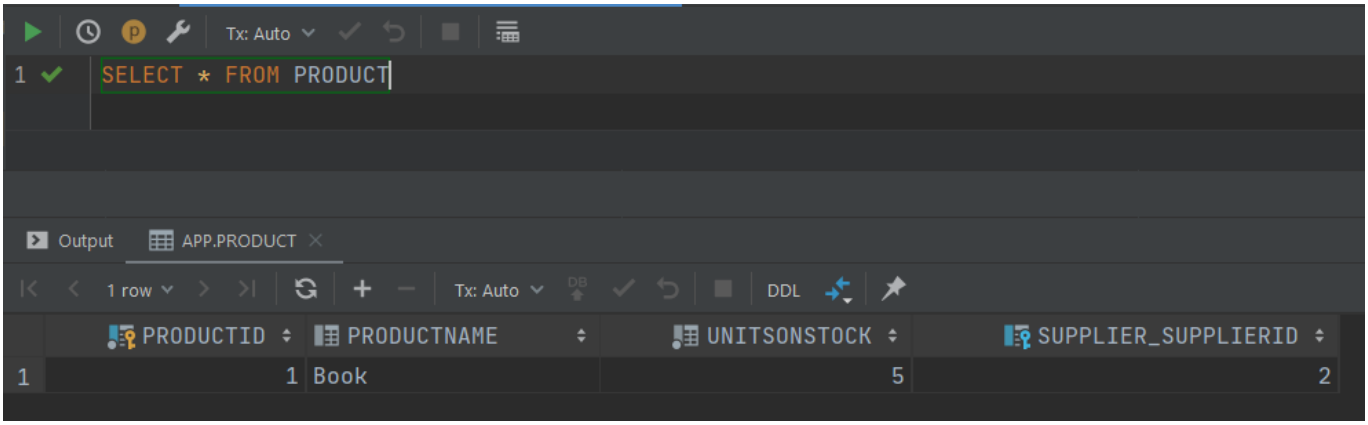
```java
    public static void main(final String[] args) throws Exception {
        final Session session = getSession();
        Transaction transaction = session.beginTransaction();
        Supplier supplier = new Supplier( companyName: "Supplier1",  street: "Kawiory",  city: "Cracow");
        Product foundProduct = session.get(Product.class,  serializable: 1);
        foundProduct.setSupplier(supplier);
        session.save(supplier);
        session.save(foundProduct);
        transaction.commit();

        try {
            System.out.println("querying all the managed entities...");
            final Metamodel metamodel = session.getSessionFactory().getMetamodel();
            for (EntityType<?> entityType : metamodel.getEntities()) {
                final String entityName = entityType.getName();
                final Query query = session.createQuery( s: "from " + entityName);
                System.out.println("executing: " + query.getQueryString());
                for (Object o : query.list()) {
                    System.out.println("  " + o);
                }
            }
        } finally {
            session.close();
        }
    }
}
```

Schemat bazy danych:



Jak widać dane dodały się poprawnie.





V. Odwrócona relacja Supplier - Product

a) Wariant z tabelą łącznikową

Usunięto z klasy Product pole Supplier.

```java
import javax.persistence.*;

@Entity
public class Product {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    public int ProductId;
    public String ProductName;
    public int UnitsOnStock;


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

    public Product() {
        // for Hibernate
    }

}
```

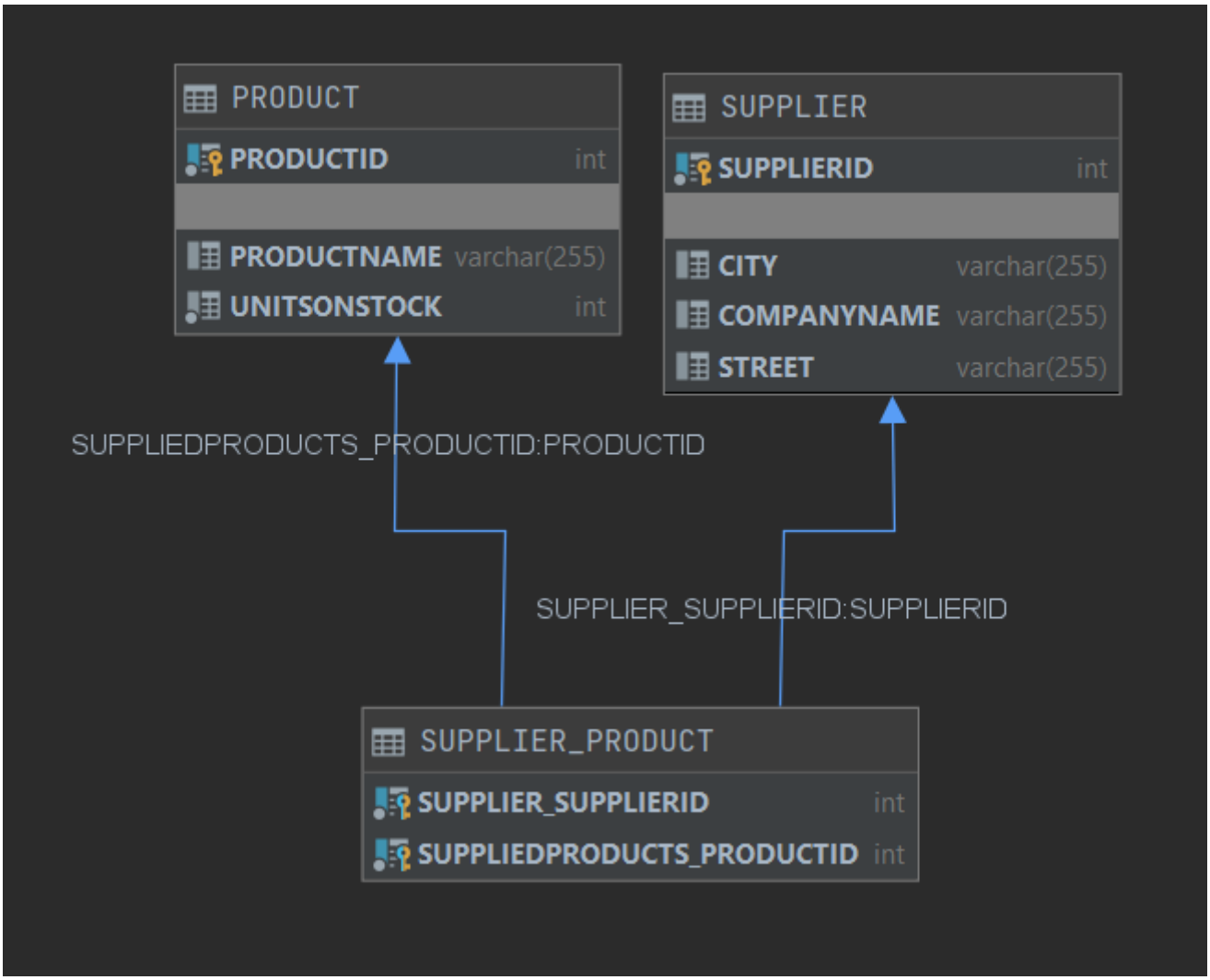
Do klasy Supplier dodano zbiór produktów.

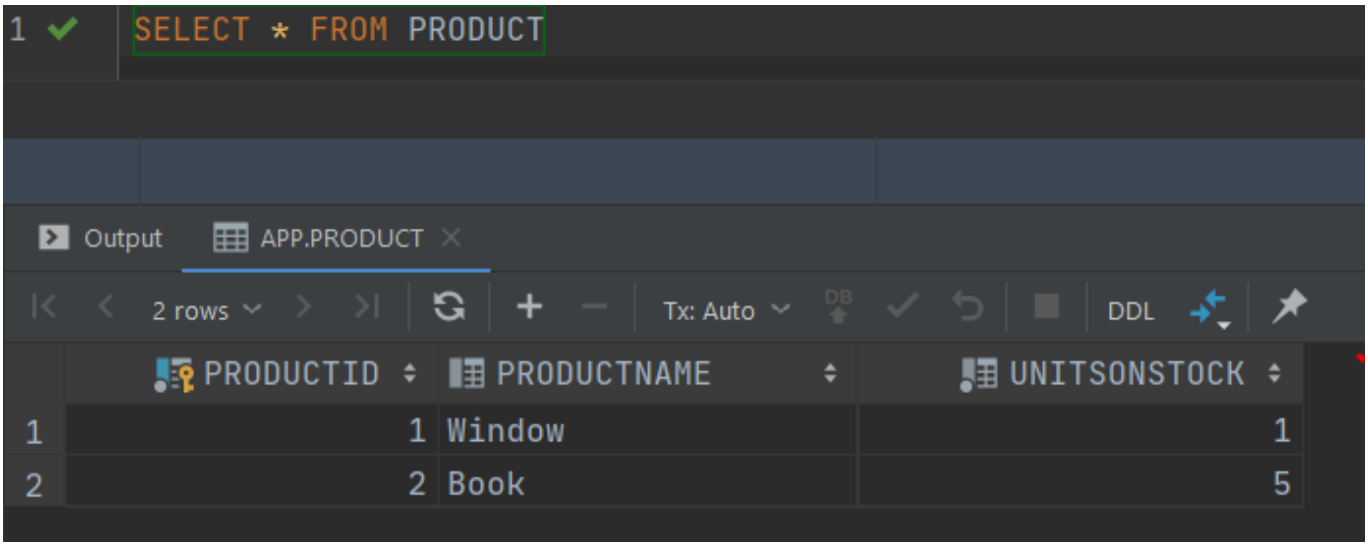
```java
import javax.persistence.*;
import java.util.HashSet;
import java.util.Set;

@Entity
public class Supplier {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    public int SupplierId;
    public String CompanyName;
    public String Street;
    public String City;

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


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

    public Supplier() {
    }

    public Supplier(String companyName, String street, String city, Product product) {
        CompanyName = companyName;
        Street = street;
        City = city;
        this.suppliedProducts.add(product);
    }

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

Dodano kilka produktów i dostawcę.

```java
public static void main(final String[] args) throws Exception {
    final Session session = getSession();
    Transaction transaction = session.beginTransaction();
    Supplier supplier = new Supplier( companyName: "Supplier1",  street: "Aleje",  city: "Warsaw");
    Product product = new Product( productName: "Window",  unitsOnStock: 1);
    Product product2 = new Product( productName: "Book",  unitsOnStock: 5);
    supplier.addProduct(product);
    supplier.addProduct(product2);
    session.save(product);
    session.save(product2);
    session.save(supplier);
    transaction.commit();

    try {
        System.out.println("querying all the managed entities...");
        final Metamodel metamodel = session.getSessionFactory().getMetamodel();
        for (EntityType<?> entityType : metamodel.getEntities()) {
            final String entityName = entityType.getName();
            final Query query = session.createQuery( s: "from " + entityName);
            System.out.println("executing: " + query.getQueryString());
            for (Object o : query.list()) {
                System.out.println("  " + o);
            }
        }
    } finally {
        session.close();
    }
}
```

Schemat bazy danych:


PRODUCT
PRODUCTID int
PRODUCTNAME varchar(255)
UNITSONSTOCK int
SUPPLIER
SUPPLIERID int
CITY varchar(255)
COMPANYNAME varchar(255)
STREET varchar(255)
SUPPLIEDPRODUCTS_PRODUCTID:PRODUCTID
SUPPLIER_SUPPLIERID:SUPPLIERID
SUPPLIER_PRODUCT
SUPPLIER_SUPPLIERID int
SUPPLIEDPRODUCTS_PRODUCTID int


1 SELECT * FROM PRODUCT
Output
APP.PRODUCT
2 rows
Tx: Auto
DDL
PRODUCTID PRODUCTNAME UNITSONSTOCK
1 1 Window 1
2 2 Book 5

```
SELECT * FROM SUPPLIER
```

Output APP.SUPPLIER

1 row

	SUPPLIERID	CITY	COMPANYNAME	STREET
1	3	Warsaw	Supplier1	Aleje

```
SELECT * FROM SUPPLIER_PRODUCT
```

Output APP.SUPPLIER_PRODUCT

2 rows

	SUPPLIER_SUPPLIERID	SUPPLIEDPRODUCTS_PRODUCTID
1	3	1
2	3	2

Dane dodały się poprawnie.