Report 1.

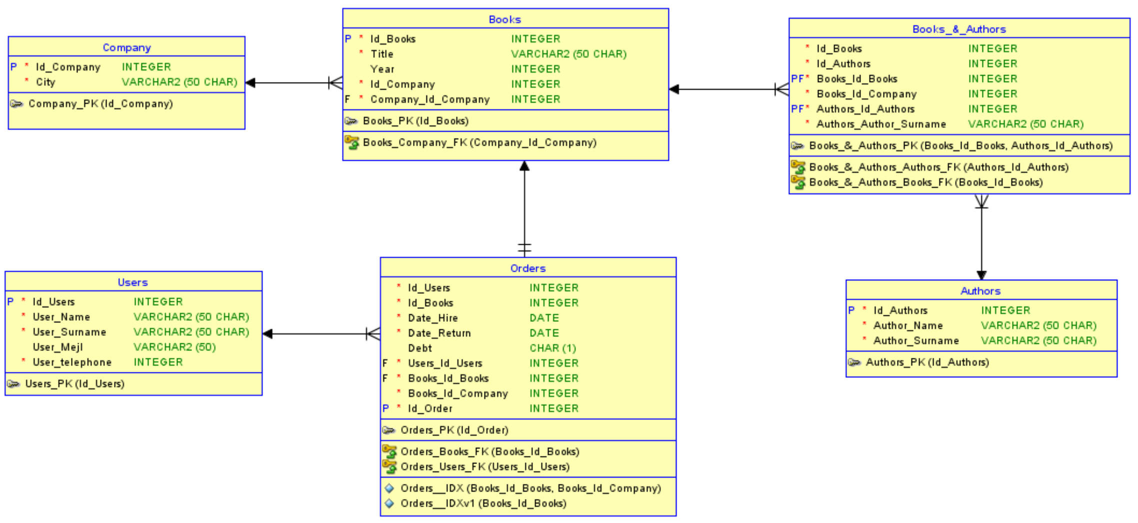
Magdalena Thomas nr 155998

Schema of database

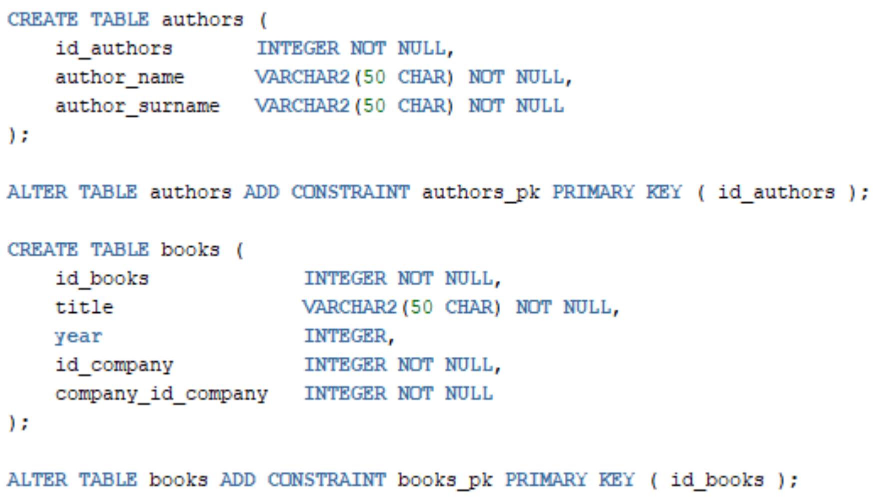
The database which I have planned can be used for library. It consists 5 tabels and 2 extra tables made of many to many relation. At least, the database has two 1:n relations (Company & Books, Users & Orders) and two n:m relations (Orders & Books and Books & Authors).

The ER diagram was prepared using SQL Developer Data Modeler, were the code was generated. Picture 1. shows diagram made with SQL tool, wheras Picture 2. presents a fragment of generated code.

Picture 1.


Company
P * Id_Company INTEGER
* City VARCHAR2 (50 CHAR)
Company_PK (Id_Company)
Books
P * Id_Books INTEGER
* Title VARCHAR2 (50 CHAR)
Year INTEGER
* Id_Company INTEGER
F * Company_Id_Company INTEGER
Books_PK (Id_Books)
Books_Company_FK (Company_Id_Company)
Books_&_Authors
* Id_Books INTEGER
* Id_Authors INTEGER
PF* Books_Id_Books INTEGER
* Books_Id_Company INTEGER
PF* Authors_Id_Authors INTEGER
* Authors_Author_Surname VARCHAR2 (50 CHAR)
Books_&_Authors_PK (Books_Id_Books, Authors_Id_Authors)
Books_&_Authors_Authors_FK (Authors_Id_Authors)
Books_&_Authors_Books_FK (Books_Id_Books)
Users
P * Id_Users INTEGER
* User_Name VARCHAR2 (50 CHAR)
* User_Surname VARCHAR2 (50 CHAR)
User_Mejl VARCHAR2 (50)
* User_telephone INTEGER
Users_PK (Id_Users)
Orders
* Id_Users INTEGER
* Id_Books INTEGER
* Date_Hire DATE
* Date_Return DATE
Debt CHAR (1)
F * Users_Id_Users INTEGER
F * Books_Id_Books INTEGER
* Books_Id_Company INTEGER
P * Id_Order INTEGER
Orders_PK (Id_Order)
Orders_Books_FK (Books_Id_Books)
Orders_Users_FK (Users_Id_Users)
Orders__IDX (Books_Id_Books, Books_Id_Company)
Orders__IDXv1 (Books_Id_Books)
Authors
P * Id_Authors INTEGER
* Author_Name VARCHAR2 (50 CHAR)
* Author_Surname VARCHAR2 (50 CHAR)
Authors_PK (Id_Authors)

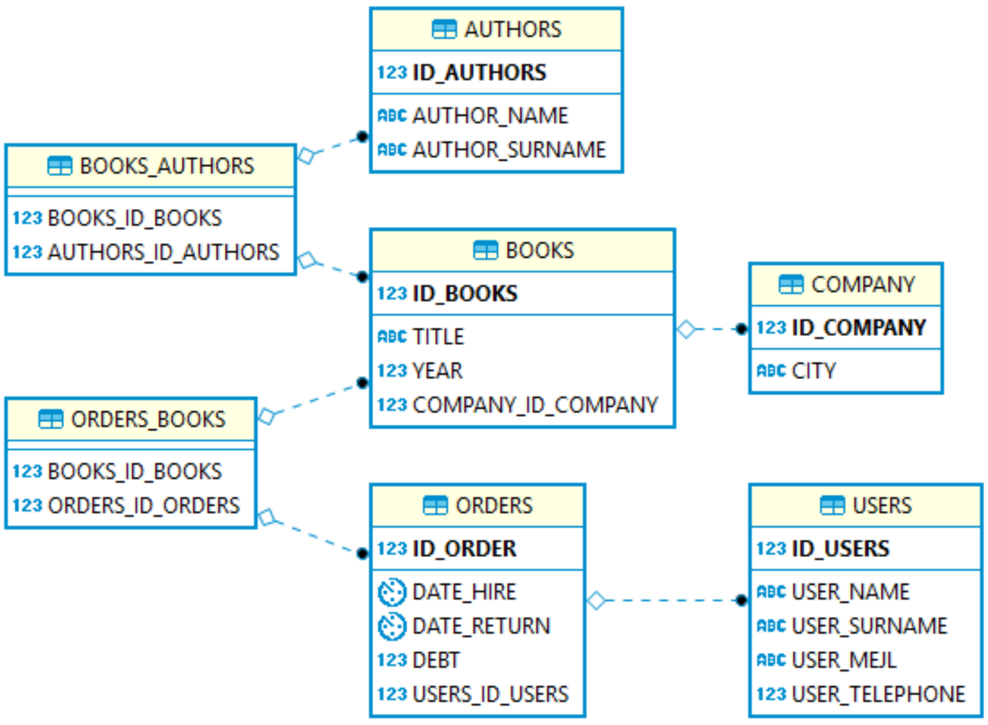
Picture 2.



```
CREATE TABLE authors (
    id_authors        INTEGER NOT NULL,
    author_name       VARCHAR2(50 CHAR) NOT NULL,
    author_surname    VARCHAR2(50 CHAR) NOT NULL
);

ALTER TABLE authors ADD CONSTRAINT authors_pk PRIMARY KEY ( id_authors );

CREATE TABLE books (
    id_books              INTEGER NOT NULL,
    title                 VARCHAR2(50 CHAR) NOT NULL,
    year                  INTEGER,
    id_company            INTEGER NOT NULL,
    company_id_company    INTEGER NOT NULL
);

ALTER TABLE books ADD CONSTRAINT books_pk PRIMARY KEY ( id_books );
```

During work, the structure of database has been changed. At least, the ER diagram made with Eclipse program looks like Picture 3. shows.

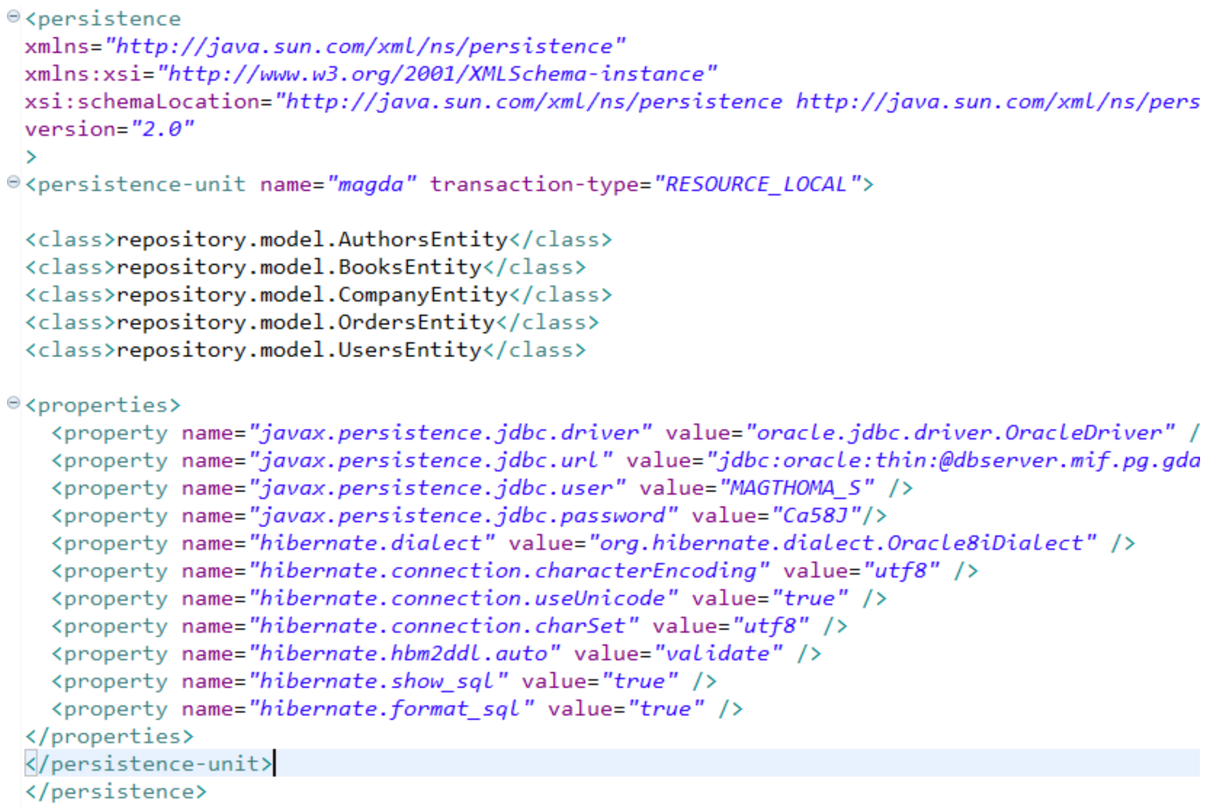
Picture 3. The finall structure of database



Description how the data was generated

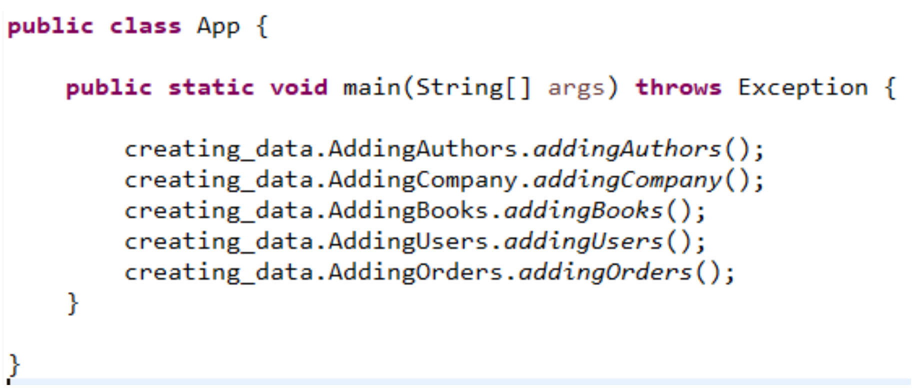
To generate data the JAVA script was written and Data Factory library were used. Pictures below present fragments of code including connection with database, persistence file ect. Every table has got two classes – item and entity which is a counterpart of item in database.
To fullfill database with data the „adding…” functions where used (Picture 5.). To save the data five methods „save” was prepared (Picture 8.).

Picture 4. The persistence file



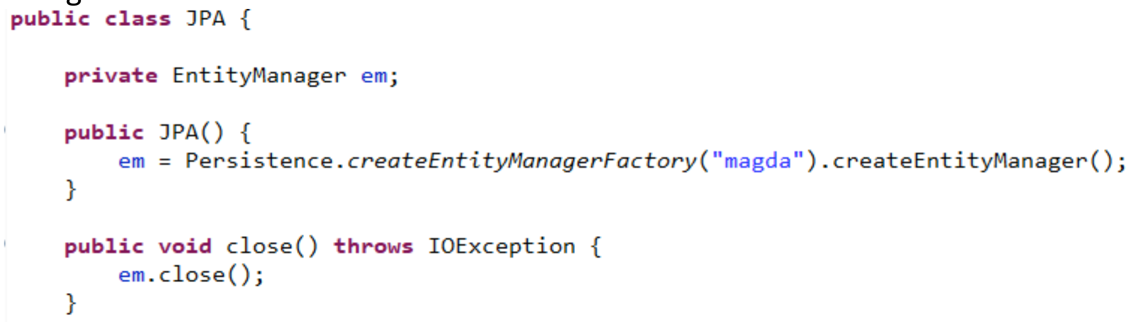
```xml
<persistence
xmlns="http://java.sun.com/xml/ns/persistence"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/persistence http://java.sun.com/xml/ns/pers
version="2.0"
>
<persistence-unit name="magda" transaction-type="RESOURCE_LOCAL">

<class>repository.model.AuthorsEntity</class>
<class>repository.model.BooksEntity</class>
<class>repository.model.CompanyEntity</class>
<class>repository.model.OrdersEntity</class>
<class>repository.model.UsersEntity</class>

<properties>
  <property name="javax.persistence.jdbc.driver" value="oracle.jdbc.driver.OracleDriver" /
  <property name="javax.persistence.jdbc.url" value="jdbc:oracle:thin:@dbserver.mif.pg.gda
  <property name="javax.persistence.jdbc.user" value="MAGTHOMA_S" />
  <property name="javax.persistence.jdbc.password" value="Ca58J"/>
  <property name="hibernate.dialect" value="org.hibernate.dialect.Oracle8iDialect" />
  <property name="hibernate.connection.characterEncoding" value="utf8" />
  <property name="hibernate.connection.useUnicode" value="true" />
  <property name="hibernate.connection.charSet" value="utf8" />
  <property name="hibernate.hbm2ddl.auto" value="validate" />
  <property name="hibernate.show_sql" value="true" />
  <property name="hibernate.format_sql" value="true" />
</properties>
</persistence-unit>
</persistence>
```

Picture 5.



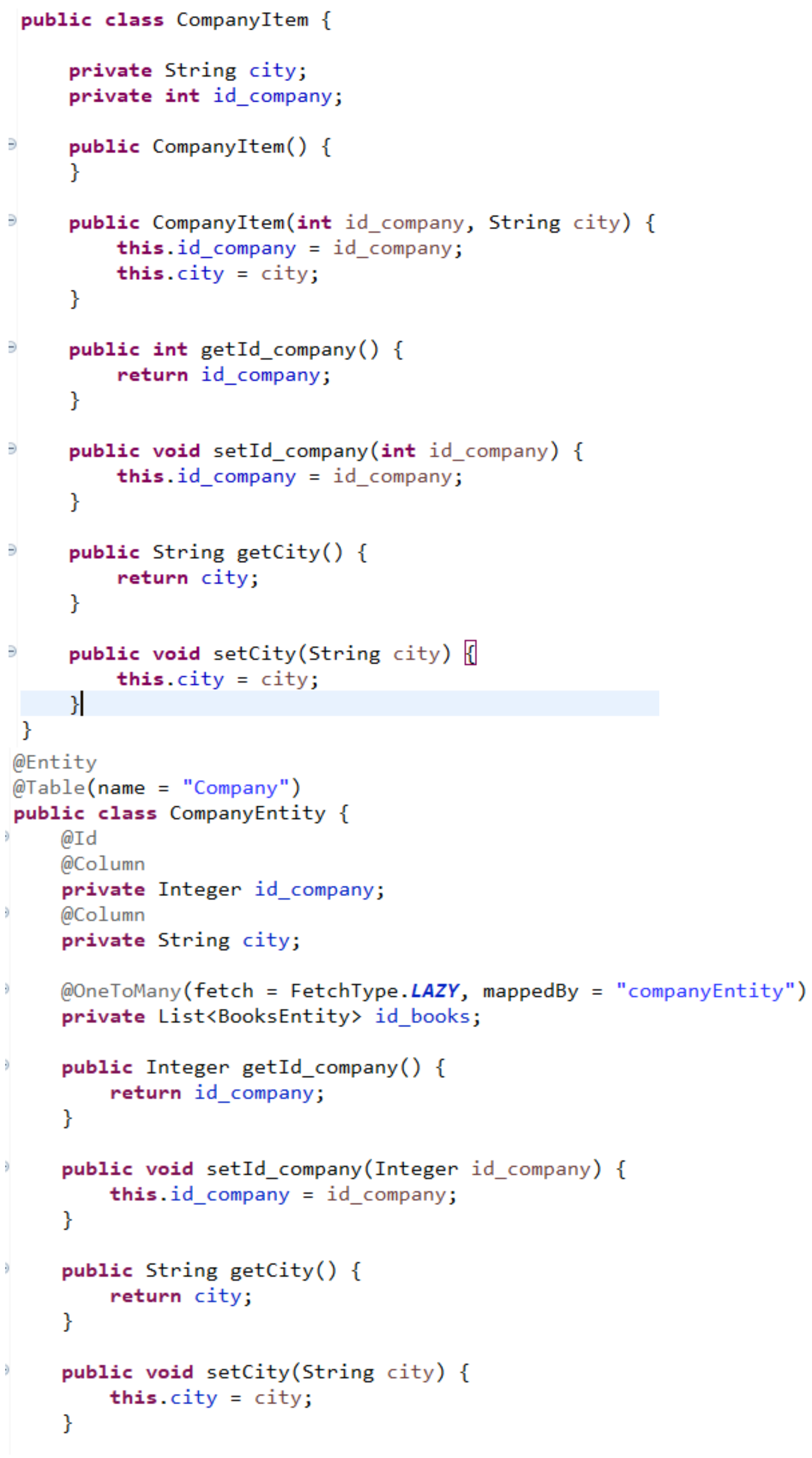
```java
public class App {

    public static void main(String[] args) throws Exception {

        creating_data.AddingAuthors.addingAuthors();
        creating_data.AddingCompany.addingCompany();
        creating_data.AddingBooks.addingBooks();
        creating_data.AddingUsers.addingUsers();
        creating_data.AddingOrders.addingOrders();
    }

}
```

Picture 6. Fragment of code in which, the JPA connection of database named ‘ ‘’magda’’ has been initiated.



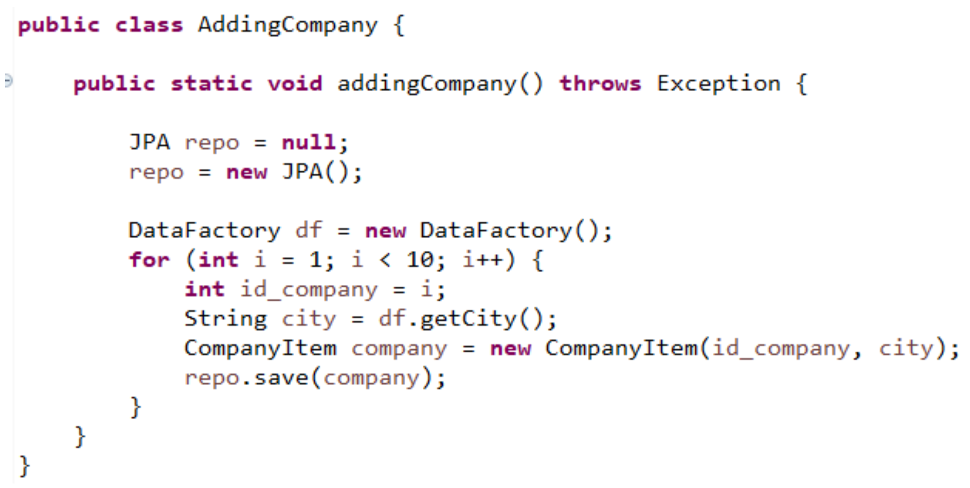
```java
public class JPA {

    private EntityManager em;

    public JPA() {
        em = Persistence.createEntityManagerFactory("magda").createEntityManager();
    }

    public void close() throws IOException {
        em.close();
    }
```

Picture 7. The example of item and entity class have been presented.

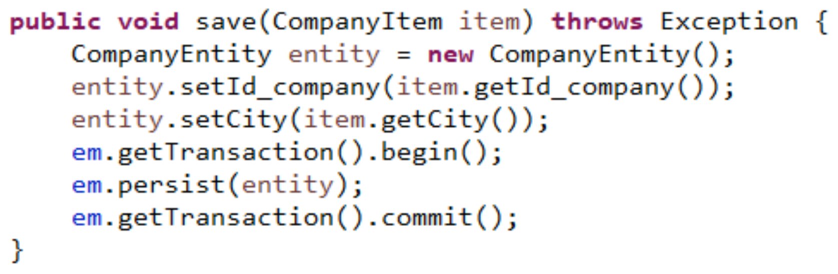


```java
public class CompanyItem {

    private String city;
    private int id_company;

    public CompanyItem() {
    }

    public CompanyItem(int id_company, String city) {
        this.id_company = id_company;
        this.city = city;
    }

    public int getId_company() {
        return id_company;
    }

    public void setId_company(int id_company) {
        this.id_company = id_company;
    }

    public String getCity() {
        return city;
    }

    public void setCity(String city) {
        this.city = city;
    }
}
@Entity
@Table(name = "Company")
public class CompanyEntity {
    @Id
    @Column
    private Integer id_company;
    @Column
    private String city;

    @OneToMany(fetch = FetchType.LAZY, mappedBy = "companyEntity")
    private List<BooksEntity> id_books;

    public Integer getId_company() {
        return id_company;
    }

    public void setId_company(Integer id_company) {
        this.id_company = id_company;
    }

    public String getCity() {
        return city;
    }

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

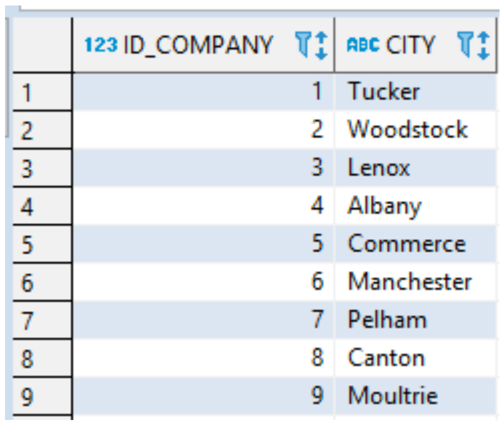
Picture 8. Fragments of code which ilustarate how the data where added to database.



```
public class AddingCompany {

    public static void addingCompany() throws Exception {

        JPA repo = null;
        repo = new JPA();

        DataFactory df = new DataFactory();
        for (int i = 1; i < 10; i++) {
            int id_company = i;
            String city = df.getCity();
            CompanyItem company = new CompanyItem(id_company, city);
            repo.save(company);
        }
    }
}
```



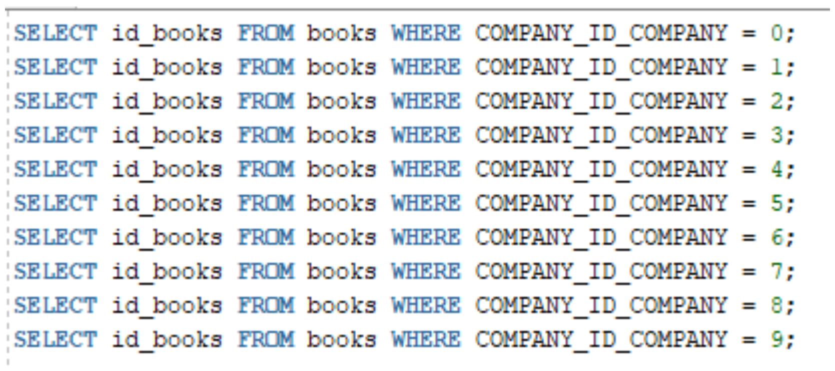
```
public void save(CompanyItem item) throws Exception {
    CompanyEntity entity = new CompanyEntity();
    entity.setId_company(item.getId_company());
    entity.setCity(item.getCity());
    em.getTransaction().begin();
    em.persist(entity);
    em.getTransaction().commit();
}
```



	ID_COMPANY	CITY
1	1	Tucker
2	2	Woodstock
3	3	Lenox
4	4	Albany
5	5	Commerce
6	6	Manchester
7	7	Pelham
8	8	Canton
9	9	Moultrie

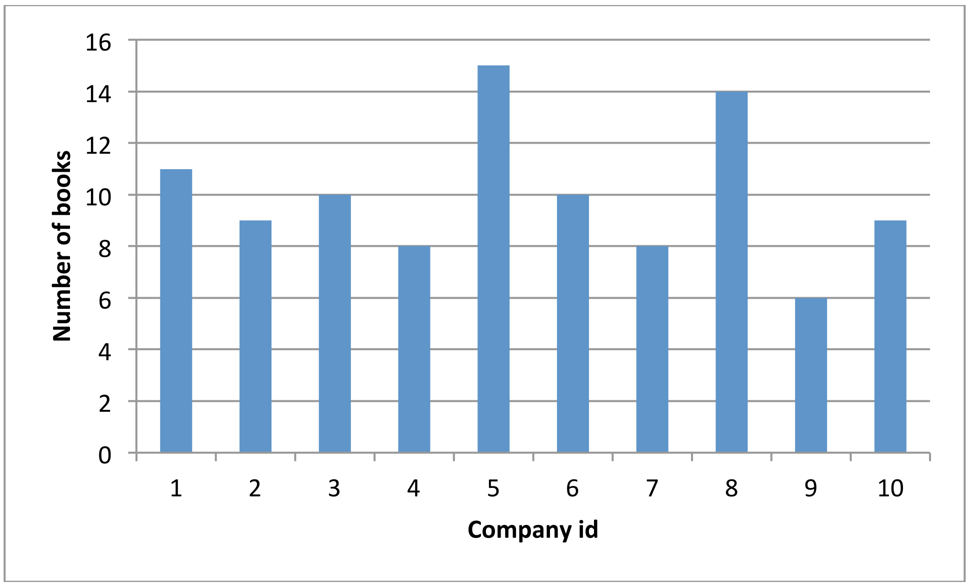
Description of statistic of the data

Picture 9. presents one of the queries statements which where used to preparted data for statics. All of the obtained data has been shown on a charts below.

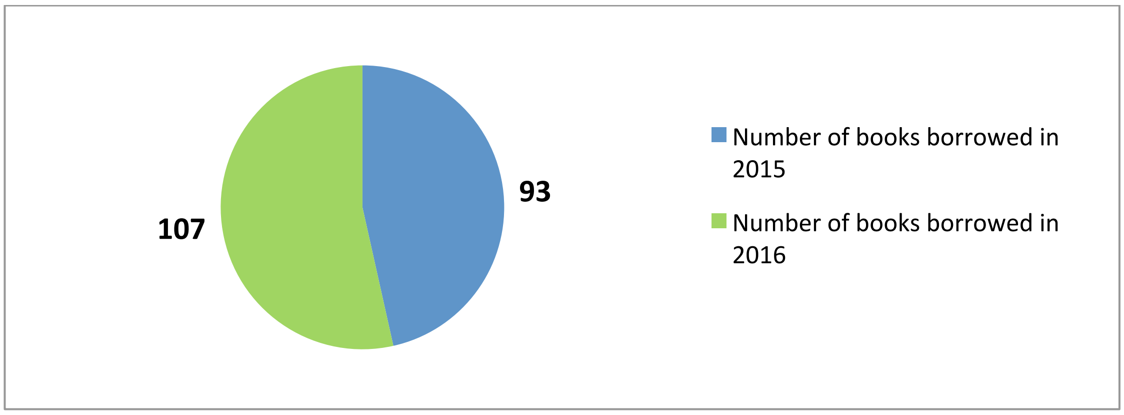


```
SELECT id_books FROM books WHERE COMPANY_ID_COMPANY = 0;
SELECT id_books FROM books WHERE COMPANY_ID_COMPANY = 1;
SELECT id_books FROM books WHERE COMPANY_ID_COMPANY = 2;
SELECT id_books FROM books WHERE COMPANY_ID_COMPANY = 3;
SELECT id_books FROM books WHERE COMPANY_ID_COMPANY = 4;
SELECT id_books FROM books WHERE COMPANY_ID_COMPANY = 5;
SELECT id_books FROM books WHERE COMPANY_ID_COMPANY = 6;
SELECT id_books FROM books WHERE COMPANY_ID_COMPANY = 7;
SELECT id_books FROM books WHERE COMPANY_ID_COMPANY = 8;
SELECT id_books FROM books WHERE COMPANY_ID_COMPANY = 9;
```

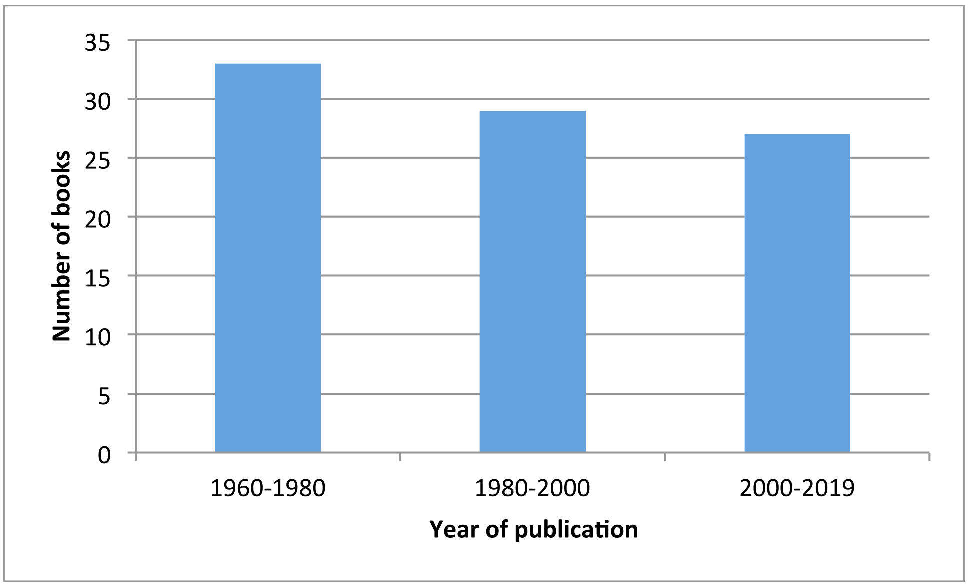
Picture 10. Chart illustrated the number of books which have the same company.



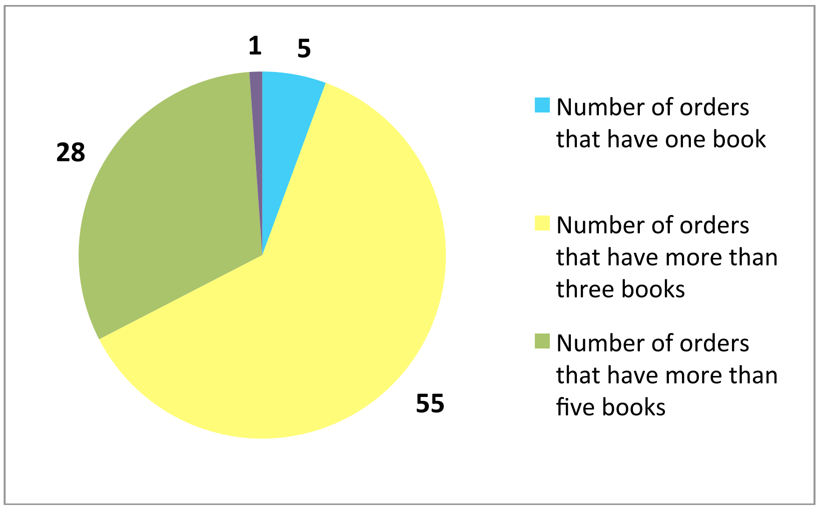
Picture 11. Chart of the number of books borrowed in 2015 and 2016.



Picture 12. Chart of the number of books published in presented years.



Picture 13. Chart of the number of orders with diffrent numbers of books.


Number of orders that have one book
Number of orders that have more than three books
Number of orders that have more than five books