Miquel Esplà Gomis
Bases de Datos (38210)
Máster Universitario en Desarrollo de Aplicaciones y Servicios Web
Departamento de Lenguajes y Sistemas Informáticos
Universitat d'Alacant
Universidad de Alicante

Relaciones en Hibernate

Uno de los problemas que nos podemos encontrar al mapear objetos con bases de datos relacionales es que la forma de establecer relaciones en ambos modelos es muy diferente. Podemos considerar tres tipos de relaciones: relaciones 1:1, relaciones 1:N y relaciones N:M. En el caso de las bases de datos, las relaciones se establecen a través de:

1. restricciones sobre los valores que una columna puede tomar en una o más tablas, y
2. actualizaciones en cascada.

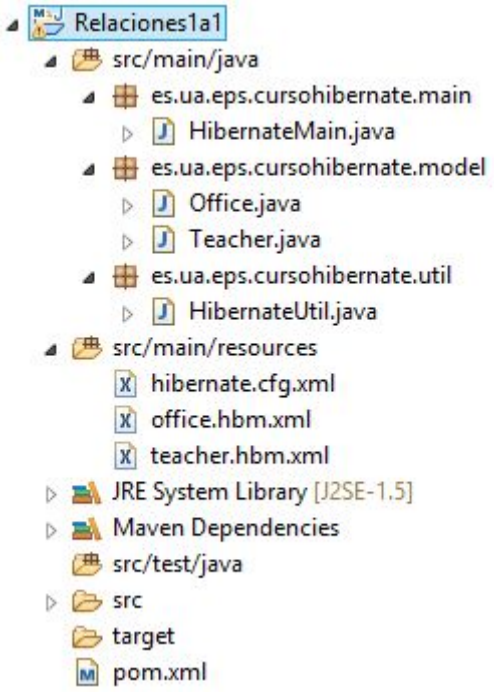
En el caso de la programación orientada a objetos, las relaciones se establecen con referencias entre objetos.

Por ejemplo, en una relación 1:1 entre dos objetos `CreditCard` y `Customer`, podríamos hacer que `CreditCard` contuviese un campo `Costumer` y, a su vez, `Customer` contuviese un campo `CreditCard`. De esta forma, dos objetos de estos tipos se "apuntarán" mutuamente, estableciendo una relación 1:1. En el caso de una base de datos relacional, las dos tablas tendrán un identificador (clave primaria), y la tabla `CreditCard` contendría un campo identificador de `Costumer` (clave ajena), que definiría la relación entre los elementos de ambas tablas.

El resto del documento muestra ejemplos de los tres tipos de relación con anotaciones en XML.

Relaciones 1:1

En el primer ejemplo, planteamos la relación entre profesores (Teacher) y despachos (Office) en una universidad. Asumimos que cada profesor sólo tiene un despacho y cada despacho tan solo puede dar cabida a un profesor. Además, cuando añadimos un despacho a la base de datos, ésta debe tener desde un principio un profesor asignado. El proyecto de Eclipse tendrá el siguiente aspecto:



El fichero `pom.xml`

El fichero `pom.xml` que se muestra a continuación es el mismo que el del documento que compara el mapeo con XML y anotaciones:

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>es.ua.eps.cursohibernate</groupId>
  <artifactId>AnotacionVsXML</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>jar</packaging>

  <name>AnotacionVsXML</name>
  <url>http://maven.apache.org</url>

  <properties>
       <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  </properties>

  <dependencies>
       <!-- SQL Server Connector -->
     <dependency>
       <groupId>net.sourceforge.jtds</groupId>
              <artifactId>jtds</artifactId>
       <version>1.3.1</version>
     </dependency>
     <!-- Hibernate 5.2.9 Final -->
     <dependency>
       <groupId>org.hibernate</groupId>
       <artifactId>hibernate-core</artifactId>
       <version>5.2.10.Final</version>
     </dependency>
   </dependencies>
</project>
```

Se puede observar que se han añadido dos dependencias. Una para descargar el driver de la base de datos y la otra para descargar las librerías necesarias de Hibernate. El driver utilizado es JTDS y nos permite la conexión con una base de datos MS SQL Server.

Tablas para almacenar los objetos

Como puede observarse, los despachos contienen un identificador del profesor asignado, un nombre de edificio y la planta en el edificio.

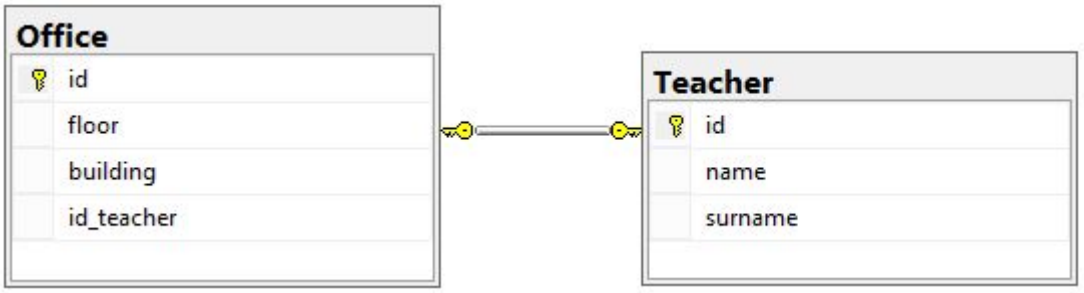


Tabla Office:

```
CREATE TABLE [dbo].[Office](
	[id] [int] IDENTITY(1,1) NOT NULL,
	[floor] [int] NOT NULL,
	[building] [nchar](10) NOT NULL,
	[id_teacher] [int] NULL,
 CONSTRAINT [PK_Office] PRIMARY KEY CLUSTERED
(
	[id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Office]  WITH CHECK ADD  CONSTRAINT [FK_Office_Teacher] FOREIGN
KEY([id_teacher])
REFERENCES [dbo].[Teacher] ([id])
GO
ALTER TABLE [dbo].[Office] CHECK CONSTRAINT [FK_Office_Teacher]
```

Para los profesores guardamos un identificador, un nombre, y un apellido.

Tabla Teacher:

```
CREATE TABLE [dbo].[Teacher](
	[id] [int] IDENTITY(1,1) NOT NULL,
	[name] [varchar](50) NOT NULL,
	[surname] [varchar](50) NOT NULL,
 CONSTRAINT [PK_Teacher] PRIMARY KEY CLUSTERED
(
	[id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
```

POJOs

Teacher (`Teacher.java`)

```java
package es.ua.eps.cursohibernate.model;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.OneToOne;
import javax.persistence.Table;


@Entity
@Table(name="TEACHER")
public class Teacher {
	@Id
	@GeneratedValue(strategy=GenerationType.IDENTITY)
	@Column(name="id")
	private int id;

	@Column(name="name")
	private String name;

	@Column(name="surname")
	private String surname;

	@OneToOne(mappedBy="teacher")
	private Office office;

	public int getId() {
		return id;
	}
	public void setId(int id) {
		this.id = id;
	}
	public String getName() {
		return name;
	}
	public void setName(String name) {
		this.name = name;
	}

	public String getSurname() {
		return surname;
	}
	public void setSurname(String surname) {
		this.surname = surname;
	}
	public Office getOffice() {
		return office;
	}
	public void setOffice(Office office) {
		this.office = office;
	}
}
```

Office (`Office.java`)

```
package es.ua.eps.cursohibernate.model;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.OneToOne;
import javax.persistence.Table;

@Entity
@Table(name="OFFICE")
public class Office {

	@Id
	@GeneratedValue(strategy=GenerationType.IDENTITY)
	@Column(name="id")
	private int id;

	@Column(name="building")
	private String building;

	@Column(name="floor")
	private int floor;

	@OneToOne
	@JoinColumn(name="id_teacher", nullable=false)
	private Teacher teacher;

	public int getId() {
		return id;
	}
	public void setId(int id) {
		this.id = id;
	}
	public String getBuilding() {
		return building;
	}
	public void setBuilding(String building) {
		this.building = building;
	}
	public int getFloor() {
		return floor;
	}
	public void setFloor(int floor) {
		this.floor = floor;
	}
	public Teacher getTeacher() {
		return teacher;
	}
	public void setTeacher(Teacher teacher) {
		this.teacher = teacher;
	}
}
```

Fichero de Configuración de Hibernate (`hibernate.cfg.xml`)

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
```

```
        "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
        "http://hibernate.org/dtd/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
       <session-factory>
        <!-- Datos de conexión con la base de datos- Driver, URL, user, password -->
        <property
name="hibernate.connection.driver_class">net.sourceforge.jtds.jdbc.Driver</property>
        <property
name="hibernate.connection.url">jdbc:jtds:sqlserver://localhost/university</property>
        <property name="hibernate.connection.username">mespla</property>
        <property name="hibernate.connection.password">mespla1</property>
        <!-- Tamaño de la Connection Pool-->
        <property name="hibernate.connection.pool_size">1</property>

        <!-- Las sesiones con la base de datos se asocian al hilo de ejecución actual -->
        <property name="hibernate.current_session_context_class">thread</property>

        <!-- Opción para depuración: muestra las SQL queries generadas -->
        <property name="hibernate.show_sql">true</property>

        <!-- Dialecto de la base de datos específica: SQL Server -->
        <property name="hibernate.dialect">org.hibernate.dialect.SQLServerDialect</property>

        <!-- Clases que contienen las anotaciones de mapeo -->
       <mapping class="es.ua.eps.cursohibernate.model.Teacher" />
       <mapping class="es.ua.eps.cursohibernate.model.Office" />

     </session-factory>
</hibernate-configuration>
```

La gestión de sesiones

La gestión de sesiones se realiza casi siempre igual y, de hecho, todos los ejemplos de este documento utilizan esta misma clase para hacerlo:

```
package es.ua.eps.cursohibernate.util;

import org.hibernate.SessionFactory;
import org.hibernate.boot.registry.StandardServiceRegistryBuilder;
import org.hibernate.cfg.Configuration;
import org.hibernate.service.ServiceRegistry;

public class HibernateUtil {

    //SessionFactory para mapeo XML
    private static SessionFactory sessionFactory;

    private static SessionFactory buildSessionFactory() {
       try {
              // Cargamos la configuración de hibernate.cfg.xml
               Configuration configuration = new Configuration();
               configuration.configure("hibernate.cfg.xml");
               System.out.println("Hibernate Configuration loaded");

               //El ServiceReistry controla las solicitudes de acceso a la base de datos
               ServiceRegistry serviceRegistry = new
                     StandardServiceRegistryBuilder().configure("hibernate.cfg.xml").build();
               System.out.println("Hibernate serviceRegistry created");

               //La SessionFactory estiona la conexión a la base de datos
               SessionFactory sessionFactory =
                      configuration.buildSessionFactory(serviceRegistry);

              return sessionFactory;
```

```
        }
        catch (Throwable ex) {
                System.err.println("Initial SessionFactory creation failed." + ex);
                throw new ExceptionInInitializerError(ex);
        }
    }

    public static SessionFactory getSessionFactory() {
        if(sessionFactory == null)
             sessionFactory = buildSessionFactory();
        else if (sessionFactory.isClosed())
                 sessionFactory = buildSessionFactory();
        return sessionFactory;
    }
}
```

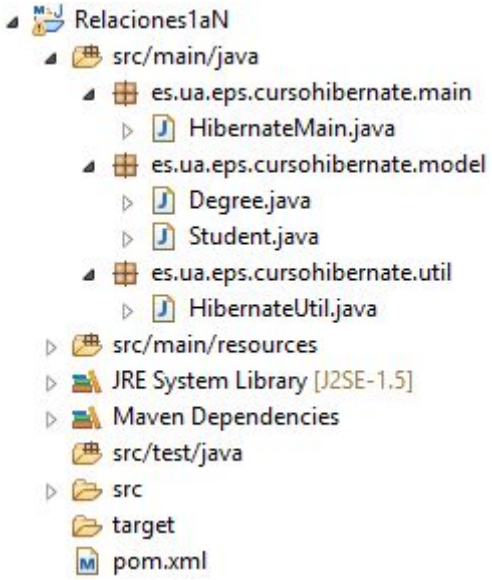
Main de ejemplo (`HibernateMain.java`)

```
package es.ua.eps.cursohibernate.main;

import org.hibernate.Session;

import es.ua.eps.cursohibernate.model.Office;
import es.ua.eps.cursohibernate.model.Teacher;
import es.ua.eps.cursohibernate.util.HibernateUtil;

public class HibernateMain {

        public static void main(String[] args) {
                Teacher teacher = new Teacher();
                teacher.setName("Pedro");
                teacher.setSurname("Esquerdo");

                Office office = new Office();
                office.setBuilding("EPS IV");
                office.setFloor(1);
                office.setTeacher(teacher);

                //Get Session
                Session session = HibernateUtil.getSessionFactory().getCurrentSession();
                System.out.println("Session factory creada");
                //start transaction
                session.beginTransaction();
                //Save the teacher and office objects
                session.save(teacher);
                session.save(office);
                //Commit transaction
                session.getTransaction().commit();
                System.out.println("Teacher ID="+teacher.getId());
                System.out.println("Office ID="+office.getId());

                //terminate session factory, otherwise program won't end
                HibernateUtil.getSessionFactory().close();
        }
}
```

Relaciones 1:N

En el siguiente ejemplo planteamos la relación entre estudiantes (Student) y carreras (Degree). En este caso asumimos que cada estudiante tan solo se puede matricular de una sola carrera y que

cada carrera tiene un número N de estudiantes matriculados. El proyecto en Eclipse tendrá el siguiente aspecto:



Tanto el fichero `pom.xml` como la clase `HibernateUtil.java` (gestión de la sesión de la base de datos) serán los mismos que en el ejemplo anterior y por tanto no se volverán a definir en este ejemplo.

Tablas para almacenar los objetos

Como puede observarse, los despachos contienen un identificador del profesor asignado, un nombre de edificio y la planta en el edificio.

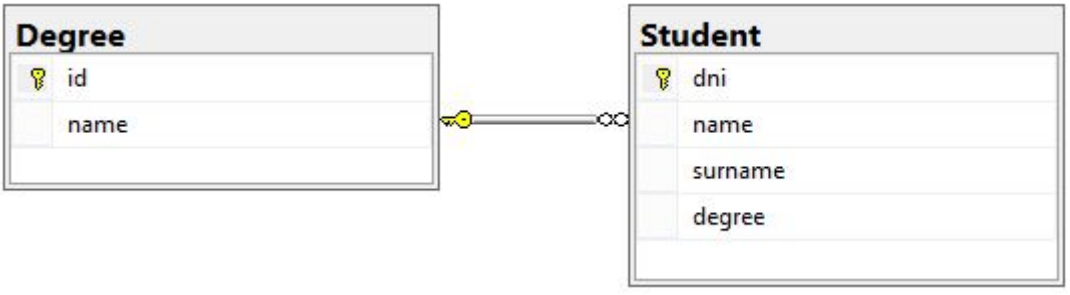


Tabla Degree:

```
CREATE TABLE [dbo].[Degree](
	[id] [int] IDENTITY(1,1) NOT NULL,
	[name] [varchar](50) NOT NULL,
 CONSTRAINT [PK_Degree] PRIMARY KEY CLUSTERED
(
	[id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
```

Para los profesores guardamos un identificador, un nombre, y un apellido.

Tabla Student:

```
CREATE TABLE [dbo].[Student](
	[dni] [nchar](10) NOT NULL,
	[name] [varchar](50) NOT NULL,
	[surname] [varchar](50) NOT NULL,
	[degree] [int] NOT NULL,
 CONSTRAINT [PK_Student] PRIMARY KEY CLUSTERED
(
	[dni] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Student]  WITH CHECK ADD  CONSTRAINT [FK_Student_Degree] FOREIGN KEY([degree])
REFERENCES [dbo].[Degree] ([id])
GO
ALTER TABLE [dbo].[Student] CHECK CONSTRAINT [FK_Student_Degree]
```

POJOs

Degree (Degree.java)

```
package es.ua.eps.cursohibernate.model;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.OneToMany;
import javax.persistence.Table;

import java.util.Set;

@Entity
@Table(name="DEGREE")
public class Degree {

	@Id
	@GeneratedValue(strategy=GenerationType.IDENTITY)
	@Column(name="id")
	private int id;

	@Column(name="name")
	private String name;

	@OneToMany(mappedBy="degree")
	private Set<Student> studentsset;

	public Set<Student> getStudentsset() {
		return studentsset;
	}
	public void setStudentsset(Set<Student> studentsset) {
		this.studentsset = studentsset;
	}
	public int getId() {
		return id;
	}
	public void setId(int id) {
```

```
		this.id = id;
	}
	public String getName() {
		return name;
	}
	public void setName(String name) {
		this.name = name;
	}
}
```

Student (`Student.java`)

```
package es.ua.eps.cursohibernate.model;

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

@Entity
@Table(name="STUDENT")
public class Student {

	@Id
	@Column(name="dni")
	private String dni;

	@Column(name="name")
	private String name;

	@Column(name="surname")
	private String surname;

	@ManyToOne
	@JoinColumn(name="degree", nullable=false)
	private Degree degree;

	public Student() {}

	public String getDni() {
		return dni;
	}
	public void setDni(String dni) {
		this.dni = dni;
	}
	public String getName() {
		return name;
	}
	public void setName(String name) {
		this.name = name;
	}
	public String getSurname() {
		return surname;
	}
	public void setSurname(String surname) {
		this.surname = surname;
	}
	public Degree getDegree() {
		return degree;
	}
	public void setDegree(Degree degree) {
		this.degree = degree;
	}
}
```

Fichero de Configuración de Hibernate (`hibernate.cfg.xml`)

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
        "http://hibernate.org/dtd/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
       <session-factory>
        <!-- Datos de conexión con la base de datos- Driver, URL, user, password -->
        <property
name="hibernate.connection.driver_class">net.sourceforge.jtds.jdbc.Driver</property>
        <property
name="hibernate.connection.url">jdbc:jtds:sqlserver://localhost/university</property>
        <property name="hibernate.connection.username">mespla</property>
        <property name="hibernate.connection.password">mespla1</property>
        <!-- Tamaño de la Connection Pool-->
        <property name="hibernate.connection.pool_size">1</property>

        <!-- Las sesiones con la base de datos se asocian al hilo de ejecución actual -->
        <property name="hibernate.current_session_context_class">thread</property>

        <!-- Opción para depuración: muestra las SQL queries generadas -->
        <property name="hibernate.show_sql">true</property>

        <!-- Dialecto de la base de datos específica: SQL Server -->
        <property name="hibernate.dialect">org.hibernate.dialect.SQLServerDialect</property>

        <!-- Clases que contienen las anotaciones de mapeo -->

       <mapping class="es.ua.eps.cursohibernate.model.Degree" />
       <mapping class="es.ua.eps.cursohibernate.model.Student" />

    </session-factory>
</hibernate-configuration>
```

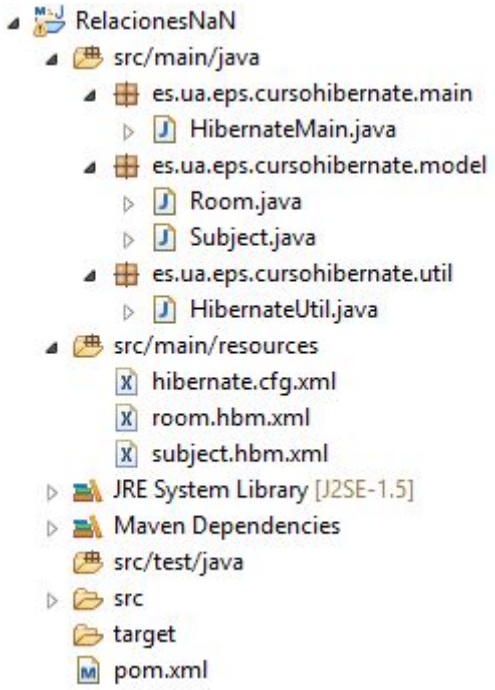
Main de ejemplo (`HibernateMain.java`)

```java
package es.ua.eps.cursohibernate.main;

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

import org.hibernate.Session;

import es.ua.eps.cursohibernate.model.Degree;
import es.ua.eps.cursohibernate.model.Student;
import es.ua.eps.cursohibernate.util.HibernateUtil;

public class HibernateMain {

       public static void main(String[] args) {

              Degree degree = new Degree();
              degree.setName("Ingeniería Informática");

              Student student1 = new Student();
              student1.setDni("21231567");
              student1.setName("Jaume");
              student1.setSurname("Esteve");
              student1.setDegree(degree);
```

```
            Student student2 = new Student();
            student2.setDni("27390111");
            student2.setName("Rosa");
            student2.setSurname("Cerveró");
            student2.setDegree(degree);

            //Get Session
            Session session = HibernateUtil.getSessionFactory().getCurrentSession();
            System.out.println("Session factory creada");
            //start transaction
            session.beginTransaction();
            //Save the Model object
            session.save(degree);
            //Important change: students have to be added as well in the same transaction
            session.save(student1);
            session.save(student2);
            //Commit transaction
            session.getTransaction().commit();
            System.out.print("Degree ID=");
            System.out.println(degree.getId());

            //terminate session factory, otherwise program won't end
            HibernateUtil.getSessionFactory().close();
        }
}
```

Relaciones M:N

En el siguiente ejemplo planteamos la relación entre asignaturas (Subject) y aulas (Room). En este caso asumimos que cada asignatura se imparte en *M* aulas, y que cada aula acoge clases de *N* asignaturas diferentes. El proyecto en Eclipse tendrá el siguiente aspecto:



Tanto el fichero pom.xml como la clase HibernateUtil.java (gestión de la sesión de la base de datos) serán los mismos que en el ejemplo anterior y por tanto no se volverán a definir en este ejemplo.

Tablas para almacenar los objetos

Como puede observarse, la relación M:N se implementa a través de una tabla adicional () que tiene por clave primaria la combinación de las claves primarias de Subject y Room.

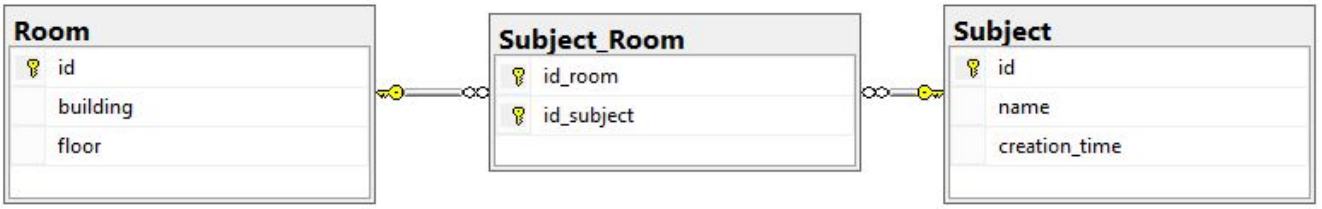


Tabla Subject:

```
CREATE TABLE [dbo].[Subject](
	[id] [int] IDENTITY(1,1) NOT NULL,
	[name] [varchar](30) NULL,
	[creation_time] [datetime] NULL,
 CONSTRAINT [PK_Subject] PRIMARY KEY CLUSTERED
(
	[id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Subject] ADD  CONSTRAINT [DF_Subject_name]  DEFAULT (NULL) FOR [name]
GO
ALTER TABLE [dbo].[Subject] ADD  CONSTRAINT [DF_Subject_creation_time]  DEFAULT (NULL) FOR
[creation_time]
```

Para los profesores guardamos un identificador, un nombre, y un apellido.

Tabla Room:

```
CREATE TABLE [dbo].[Room](
	[id] [int] IDENTITY(1,1) NOT NULL,
	[building] [varchar](50) NOT NULL,
	[floor] [int] NOT NULL,
 CONSTRAINT [PK_Room] PRIMARY KEY CLUSTERED
(
	[id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
```

Tabla Subject_Roms:

```
CREATE TABLE [dbo].[Subject_Room](
	[id_room] [int] NOT NULL,
	[id_subject] [int] NOT NULL,
 CONSTRAINT [PK_Subject_Room] PRIMARY KEY CLUSTERED
(
	[id_room] ASC,
```

```
	[id_subject] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Subject_Room]  WITH CHECK ADD  CONSTRAINT [FK_Subject_Room_Room] FOREIGN
KEY([id_room])
REFERENCES [dbo].[Room] ([id])
GO
ALTER TABLE [dbo].[Subject_Room] CHECK CONSTRAINT [FK_Subject_Room_Room]
GO
ALTER TABLE [dbo].[Subject_Room]  WITH CHECK ADD  CONSTRAINT [FK_Subject_Room_Subject] FOREIGN
KEY([id_subject])
REFERENCES [dbo].[Subject] ([id])
GO
ALTER TABLE [dbo].[Subject_Room] CHECK CONSTRAINT [FK_Subject_Room_Subject]
GO
```

POJOs

Room (Room.java)

```
package es.ua.eps.cursohibernate.model;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.ManyToMany;
import javax.persistence.Table;

import java.util.Set;

@Entity
@Table(name="ROOM")
public class Room {

	@Id
	@Column(name="id")
	@GeneratedValue(strategy=GenerationType.IDENTITY)
	private int id;

	@Column(name="building")
	private String building;

	@Column(name="floor")
	private int floor;

	@ManyToMany(mappedBy = "rooms")
	private Set<Subject> subjects;

	public Set<Subject> getSubjects() {
		return subjects;
	}
	public void setSubjects(Set<Subject> subjects) {
		this.subjects = subjects;
	}
	public int getId() {
		return id;
	}
```

```
	public void setId(int id) {
		this.id = id;
	}
	public String getBuilding() {
		return building;
	}
	public void setBuilding(String building) {
		this.building = building;
	}
	public int getFloor() {
		return floor;
	}
	public void setFloor(int floor) {
		this.floor = floor;
	}
}
```

Subject (`Subject.java`)

```
package es.ua.eps.cursohibernate.model;

import java.util.Date;
import java.util.Set;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinTable;
import javax.persistence.ManyToMany;
import javax.persistence.Table;

import javax.persistence.JoinColumn;

@Entity
@Table(name="SUBJECT")
public class Subject {

	@Id
	@Column(name="id")
	@GeneratedValue(strategy=GenerationType.IDENTITY)
	private int id;

	@Column(name="name")
	private String name;

	@Column(name="creation_time")
	private Date creationTime;

	@ManyToMany
	@JoinTable(name = "Subject_Room",
				joinColumns = { @JoinColumn(name = "id_subject") },
				inverseJoinColumns = { @JoinColumn(name = "id_room") })
	private Set<Room> rooms;

	public Set<Room> getRooms() {
		return rooms;
	}
	public void setRooms(Set<Room> rooms) {
		this.rooms = rooms;
	}
	public int getId() {
		return id;
	}
	public void setId(int id) {
		this.id = id;
	}
```

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

	public Date getCreationTime() {
		return creationTime;
	}
	public void setCreationTime(Date insertTime) {
		this.creationTime = insertTime;
	}
}
```

Fichero de Configuración de Hibernate (`hibernate.cfg.xml`)

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
        "http://hibernate.org/dtd/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
       <session-factory>
        <!-- Datos de conexión con la base de datos- Driver, URL, user, password -->
        <property
name="hibernate.connection.driver_class">net.sourceforge.jtds.jdbc.Driver</property>
        <property
name="hibernate.connection.url">jdbc:jtds:sqlserver://localhost/university</property>
        <property name="hibernate.connection.username">mespla</property>
        <property name="hibernate.connection.password">mespla1</property>
        <!-- Tamaño de la Connection Pool-->
        <property name="hibernate.connection.pool_size">1</property>

        <!-- Las sesiones con la base de datos se asocian al hilo de ejecución actual -->
        <property name="hibernate.current_session_context_class">thread</property>

        <!-- Opción para depuración: muestra las SQL queries generadas -->
        <property name="hibernate.show_sql">true</property>

        <!-- Dialecto de la base de datos específica: SQL Server -->
        <property name="hibernate.dialect">org.hibernate.dialect.SQLServerDialect</property>

        <!-- Clases que contienen las anotaciones de mapeo -->
       <mapping class="es.ua.eps.cursohibernate.model.Room" />
       <mapping class="es.ua.eps.cursohibernate.model.Subject" />
    </session-factory>

</hibernate-configuration>
```

Main de ejemplo (`HibernateMain.java`)

```
package es.ua.eps.cursohibernate.main;

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

import org.hibernate.Session;

import es.ua.eps.cursohibernate.model.Room;
```

```java
import es.ua.eps.cursohibernate.model.Subject;
import es.ua.eps.cursohibernate.util.HibernateUtil;

public class HibernateMain {

	public static void main(String[] args) {
		Room room1 = new Room();
		room1.setBuilding("EPS IV");
		room1.setFloor(2);

		Room room2 = new Room();
		room2.setBuilding("EPS I");
		room2.setFloor(0);

		Subject subject1 = new Subject();
		subject1.setName("Compiladores");
		subject1.setCreationTime(new Date());

		Subject subject2 = new Subject();
		subject2.setName("Física");
		subject2.setCreationTime(new Date());

		Subject subject3 = new Subject();
		subject3.setName("Aprendizaje");
		subject3.setCreationTime(new Date());

		Set<Subject> subjects1=new HashSet<Subject>();
		subjects1.add(subject1);
		subjects1.add(subject2);
		Set<Subject> subjects2=new HashSet<Subject>();
		subjects2.add(subject3);

		Set<Room> rooms1=new HashSet<Room>();
		rooms1.add(room1);
		rooms1.add(room2);
		Set<Room> rooms2=new HashSet<Room>();
		rooms2.add(room1);
		Set<Room> rooms3=new HashSet<Room>();
		rooms3.add(room2);

		subject1.setRooms(rooms1);
		subject2.setRooms(rooms2);
		subject3.setRooms(rooms3);

		room1.setSubjects(subjects1);
		room2.setSubjects(subjects2);
		//Get Session
		Session session = HibernateUtil.getSessionFactory().getCurrentSession();
		System.out.println("Session factory creada");
		//start transaction
		session.beginTransaction();

		//Important change: students have to be added as well in the same transaction
		//session.save(room1);
		session.save(room1);
		session.save(room2);
		session.save(subject1);
		session.save(subject2);
		session.save(subject3);
		//Commit transaction
		session.getTransaction().commit();
		System.out.print("Degree ID=");
		System.out.println(room1.getId());
		System.out.print("Degree ID=");
		System.out.println(room2.getId());

		//terminate session factory, otherwise program won't end
		HibernateUtil.getSessionFactory().close();
	}
```

```
}
```