Licenciatura en Sistemas - Orientación a Objetos II – 2024

Prof. Titular: Mg. María Alejandra Vranić

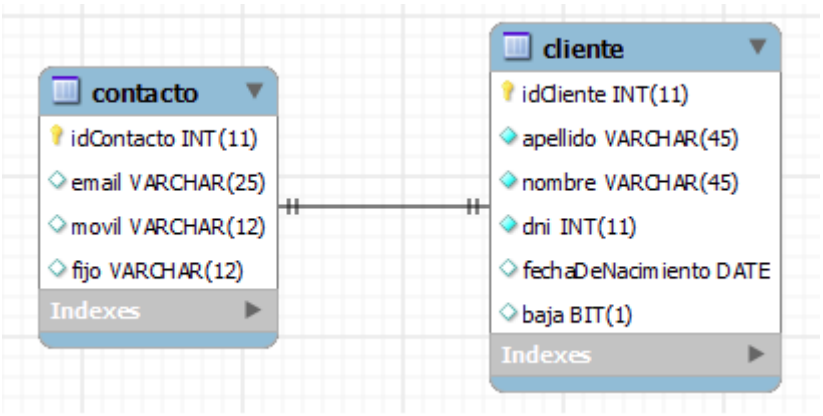
Prof. Ayudantes: Esp. Lic. Gustavo Siciliano
Lic. Ezequiel Scordamaglia
Lic. Oscar Ruina

Índice

1) Descripción de la DB y diagrama de clases

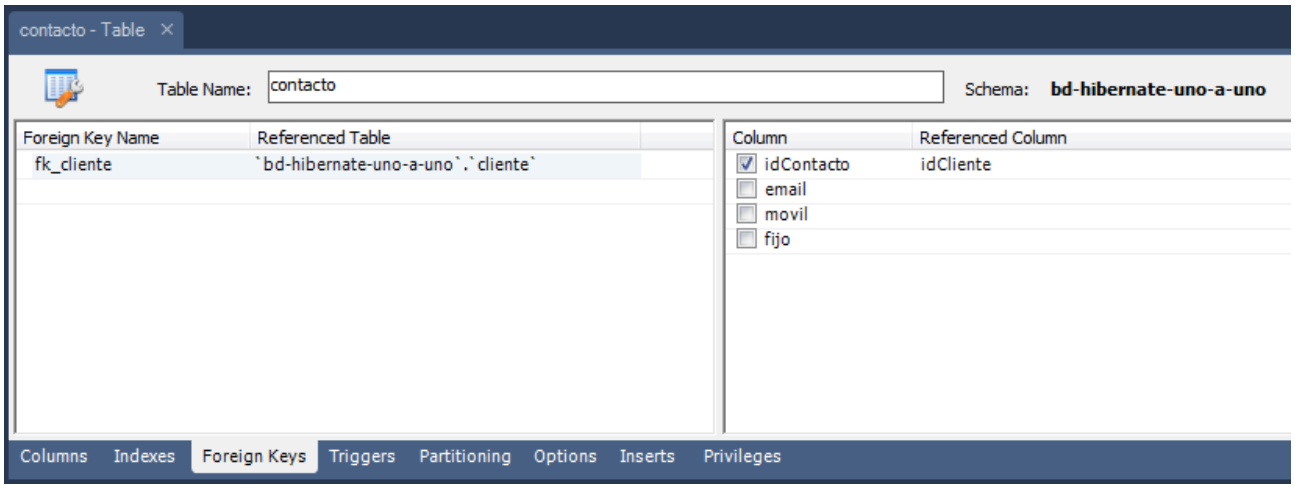
La clave primaria de la entidad contacto va depender de la tabla cliente y para cada cliente existe a lo sumo un contacto.

Diagrama Entidad Relación:



Relaciones entre las tablas

Cuando se agrega un registro cliente, en el caso de agregar el contacto el idContacto será el mismo número de idCliente



Detalle de FKs

Ahora el Contacto va ser atributo de Cliente, ya que cuando traemos un objeto Cliente también nos interesa saber los datos de contacto.

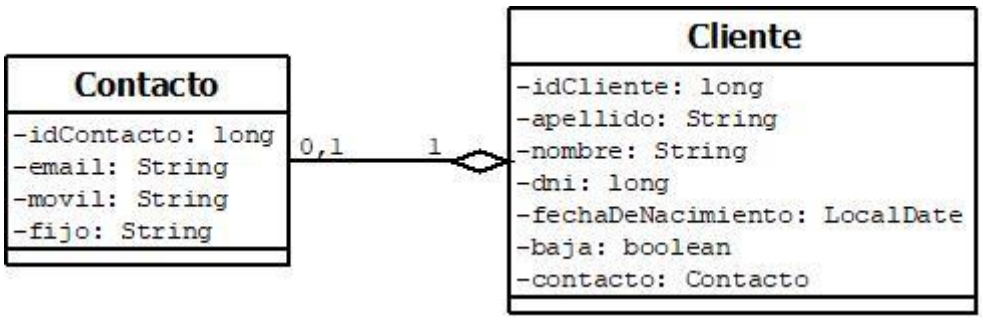


Diagrama de clases

2) Clase de datos Contacto y relación con Cliente

Contacto.java

```
package datos;

public class Contacto {
	private long idContacto;
	private String email;
	private String movil;
	private String fijo;
	private Cliente cliente;

	public Contacto(){}

	public Contacto(String email, String movil, String fijo, Cliente cliente) {
		this.email = email;
		this.movil = movil;
		this.fijo = fijo;
		this.cliente=cliente;
	}

	public long getIdContacto() {
		return idContacto;
	}
	protected void setIdContacto(long idContacto) {
		this.idContacto = idContacto;
	}
	public String getEmail() {
		return email;
	}
	public void setEmail(String email) {
		this.email = email;
	}
	public String getMovil() {
		return movil;
	}
	public void setMovil(String movil) {
		this.movil = movil;
	}
	public String getFijo() {
		return fijo;
	}
	public void setFijo(String fijo) {
		this.fijo = fijo;
	}
	public Cliente getCliente() {
		return cliente;
	}
	public void setCliente(Cliente cliente) {
		this.cliente = cliente;
	}
	@Override
	public String toString() {
		return "Contacto [idContacto=" + idContacto + ", email=" + email + ", movil=" + movil + ",
fijo=" + fijo + "]";
	}
}
```

Cambios en Cliente.java

Se agregan a la clase del proyecto anterior los siguientes cambios:

1) Un atributo de tipo Contacto para establecer la relación uno a uno con la clase.

```
private Contacto contacto;
```

2) Se incluye dentro del constructor que recibe parámetros:

```
this.contacto=contacto;
```

3) Se escriben los métodos de set y get como para cualquier atributo:

```
public Contacto getContacto() {
	return contacto;
}

public void setContacto(Contacto contacto) {
	this.contacto = contacto;
}
```

4) Finalmente, se modifica el toString para poder imprimirlo:

```
@Override
public String toString() {
	return "Cliente [idCliente=" + idCliente + ", apellido=" + apellido + ", nombre=" + nombre + ", dni=" +
dni + ", fechaDeNacimiento=" + fechaDeNacimiento + ", baja=" + baja + ", contacto=" + contacto + "]";
}
```

Cambios en el mapeo de Cliente sobre Cliente.hbm.xml

Únicamente se debe agregar la línea de la relación entre las clases, puede ubicarse después del último tag de property:

```
<one-to-one name="contacto" class="datos.Contacto" ></one-to-one>
```

3) Archivo de mapeo de Contacto

La línea más importante es la del idContacto, ya que se especifica que es una FK en la BD.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
   <class name="datos.Contacto" table="contacto" >
        <id name="idContacto" column="idContacto">
           <generator class="foreign">
                    <param name="property">cliente</param>
             </generator>
       </id>
       <property name="email" type="string" column="email" />
       <property name="movil" type="string" column="movil" />
       <property name="fijo" type="string" column="fijo" />
       <one-to-one name="cliente"  class="datos.Cliente"></one-to-one>
   </class>
</hibernate-mapping>
```

4) Clase de acceso a datos para el objeto Contacto (ContactoDao) y cambios sobre ClienteDao

ContactoDao

```
package dao;
import org.hibernate.HibernateException;
import org.hibernate.Session;
import org.hibernate.Transaction;
import datos.Contacto;
public class ContactoDao {
   private static Session session;
   private Transaction tx;
   private void iniciaOperacion() throws HibernateException {
       session = HibernateUtil.getSessionFactory().openSession();
       tx = session.beginTransaction();
   }
   private void manejaExcepcion(HibernateException he) throws HibernateException {
       tx.rollback();
       throw new HibernateException("ERROR en la capa de acceso a datos", he);
   }
   public int agregar(Contacto objeto) {
       int id = 0;
       try {
           iniciaOperacion();
           id = Integer.parseInt(session.save(objeto).toString());
           tx.commit();
       } catch (HibernateException he) {
           manejaExcepcion(he);
           throw he;
       } finally {
           session.close();
       }
       return id;
   }
   public Contacto traer(long idContacto) {
       Contacto objeto = null;
       try {
              iniciaOperacion();
              objeto = (Contacto) session.get(Contacto.class, idContacto);
       } finally {
              session.close();
       }
       return objeto;
  }
  public void actualizar(Contacto objeto) {
       try {
              iniciaOperacion();
              session.update(objeto);
              tx.commit();
       } catch (HibernateException he) {
              manejaExcepcion(he);
              throw he;
       } finally {
              session.close();
       }
  }
```

```
    public void eliminar(Contacto objeto) {
        try {
                iniciaOperacion();
                session.delete(objeto);
                tx.commit();
        } catch (HibernateException he) {
                manejaExcepcion(he);
                throw he;
        } finally {
                session.close();
        }
    }
}
```

Cambios sobre ClienteDao

Se agrega el siguiente método:

```
public Cliente traerClienteYContacto(long idCliente) throws HibernateException {
        Cliente objeto = null;
        try {
                iniciaOperacion();
                String hql = "from Cliente c inner join fetch c.contacto where c.idCliente = :idCliente";
                objeto = (Cliente) session.createQuery(hql).setParameter("idCliente",
idCliente).uniqueResult();
        } finally {
                session.close();
        }
        return objeto;
}
```

5) Clase de lógica de negocio del objeto Contacto (ContactoABM) y cambios sobre ClienteABM

ContactoABM

```
package negocio;
import dao.ContactoDao;
import datos.Cliente;
import datos.Contacto;
public class ContactoABM {
	ContactoDao dao = new ContactoDao();
	public Contacto traer(long idContacto) {
		Contacto c = dao.traer(idContacto);
		return c;
	}

	public int agregar(String email, String movil, String fijo, Cliente cliente) {
		// Lanzar excepción si el cliente ya tiene un contacto
		Contacto c = new Contacto(email, movil, fijo, cliente);
		return dao.agregar(c);
	}
	public void modificar(Contacto c) {
		dao.actualizar(c);
	}
	public void eliminar(long idContacto) {
		Contacto c = dao.traer(idContacto);
		dao.eliminar(c);
	}
}
```

Cambios sobre ClienteABM

Se realizan los siguiente cambios:

1) Se incluye el objeto contacto en el método agregar:

```
public int agregar(String apellido, String nombre, int dni, LocalDate fechaDeNacimiento,
Contacto contacto) {
	Cliente c = new Cliente(apellido, nombre, dni, fechaDeNacimiento, contacto);
	return dao.agregar(c);
}
```

2) Se agrega el método “traerClienteYContacto”:

```
public Cliente traerClienteYContacto(long idCliente) {
	return dao.traerClienteYContacto(idCliente);
}
```

6) Testeo

```
package test;
import datos.Cliente;
import negocio.ClienteABM;

public class TestTraerClienteYContacto {

	public static void main(String[] args) {
		ClienteABM abmCliente = new ClienteABM();
		long idCliente=1;
		Cliente c=abmCliente.traerClienteYContacto(idCliente);
		System.out.printf("\nCliente y contacto: %s", c);
	}
}
```

```
package test;
import datos.Cliente;
import negocio.ClienteABM;
import negocio.ContactoABM;

public class TestAgregarContacto {

	public static void main(String[] args) {

		ClienteABM abmCliente =new ClienteABM();
		Cliente cliente=abmCliente.traer(1L);
		System.out.println(cliente);

		ContactoABM abmContacto = new ContactoABM();
		abmContacto.agregar("ajaramillo@unla.edu.ar","11-1111-1111","011-1111-111
	1", cliente);
	 }
}
```

7) Tarea

Completar el proyecto y generar los métodos de negocio como se realizó con la práctica anterior.