DECSAI
Departamento de Ciencias de la Computación e I.A.
Universidad de Granada

Acceso a los datos

© Fernando Berzal, berzal@acm.org

Acceso a los datos

- Bases de datos relacionales: SQL
- O/R Mapping
- Bases de datos distribuidas
- Bases de datos NoSQL
- Bases de datos multidimensionales: Data Warehousing

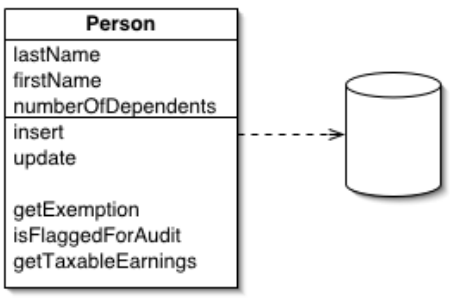
Acceso a los datos

Distintas formas de acceder a los datos almacenados en una base de datos desde una aplicación:

- **Registros activos**
  (encapsulan las estructuras de datos externas).
- **Gateways**
  (clases auxiliares con operaciones CRUD).
- **O/R Mapping**
  [object-relational mapping]

Acceso a los datos

Registros activos
Objetos que encapsulan directamente las estructuras de datos externas (p.ej. tuplas de una base de datos relacional) e incorporan la lógica del dominio que les corresponda, aparte de las operaciones necesarias para obtener y guardar objetos en la base de datos.



Acceso a los datos

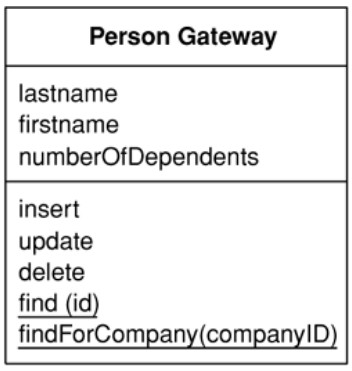
Gateways

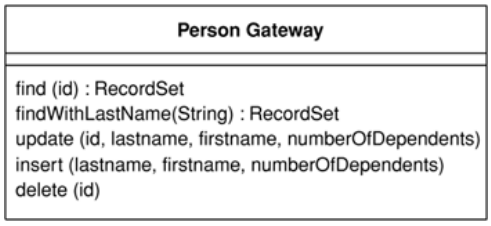
Clases auxiliares que se corresponden con las entidades presentes en la base de datos e implementan las operaciones necesarias para manipular la base de datos [CRUD: Create, Retrieve, Update & Delete].

- El uso de gateways nos permite no mezclar la lógica de la aplicación con el acceso a los datos externos.

Acceso a los datos

Gateways





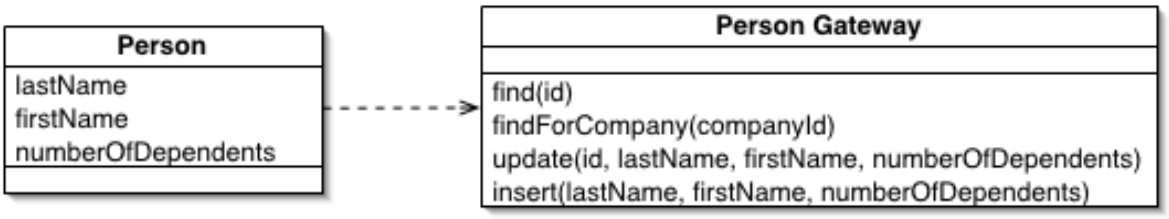
Row Data Gateway
(1 gateway/tupla)

Table Data Gateway
(1 gateway/tabla)

Acceso a los datos

Gateways

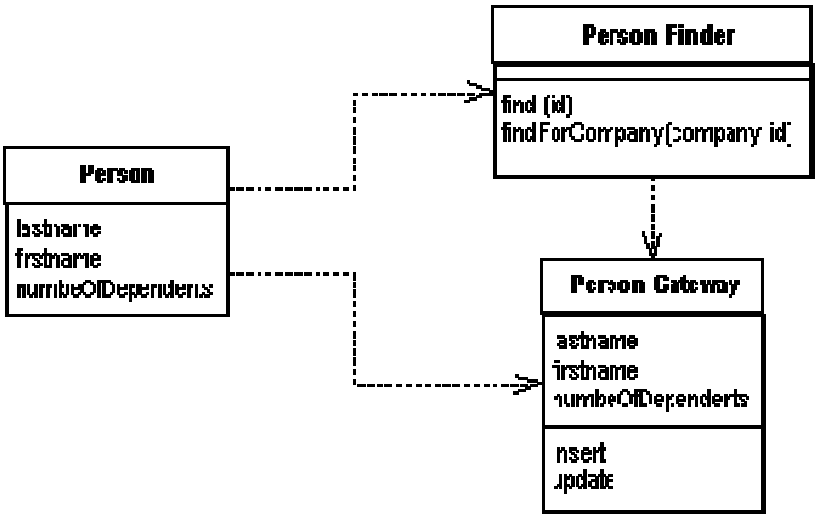
Table Data Gateway



Acceso a los datos

Gateways

Row Data Gateway



Acceso a los datos

O/R Mapping.

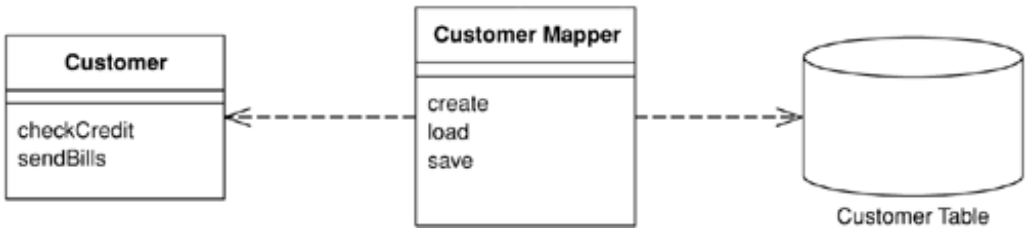
Se establece una correspondencia entre el modelo orientado a objetos del dominio y la representación de los distintos objetos en una base de datos relacional.

- En las dos alternativas anteriores, los objetos de la aplicación han de ser conscientes de cómo se representan en la base de datos.
- En el caso del O/R mapping, los objetos pueden ignorar la estructura de la base de datos y cómo se realiza la comunicación con la base de datos.

Acceso a los datos

O/R Mapping.

Se establece una correspondencia entre el modelo orientado a objetos del dominio y la representación de los distintos objetos en una base de datos relacional.



Acceso a los datos

O/R Mapping.

Se establece una correspondencia entre el modelo orientado a objetos del dominio y la representación de los distintos objetos en una base de datos relacional.

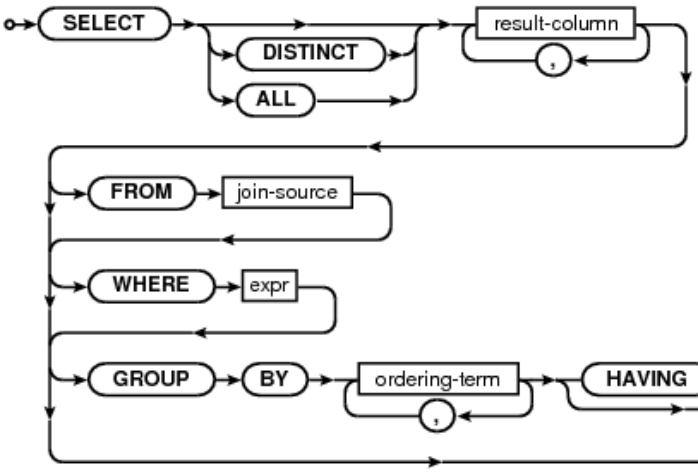
- La inversión de control característica de esta opción independiza el modelo orientado a objetos del dominio de la capa de acceso a los datos: se puede cambiar la base de datos sin tener que tocar el modelo orientado a objetos del dominio y viceversa.

- Solución más flexible (facilita el desarrollo, la depuración y la evolución de las aplicaciones).

Acceso a bases de datos

SQL

Structured Query Language



SQL Call-Level Interface [CLI]

ISO SQL/CLI (estándar SQL-92)

API [Application Programming Interface] para acceder a una base de datos relacional utilizando sentencias SQL desde el código de una aplicación:

- ODBC [Open DataBase Connectivity] de Microsoft.
- JDBC [Java DataBase Connectivity] en Java.
- ADO.NET [ActiveX Data Objects] para .NET.
- DB-API en Python.

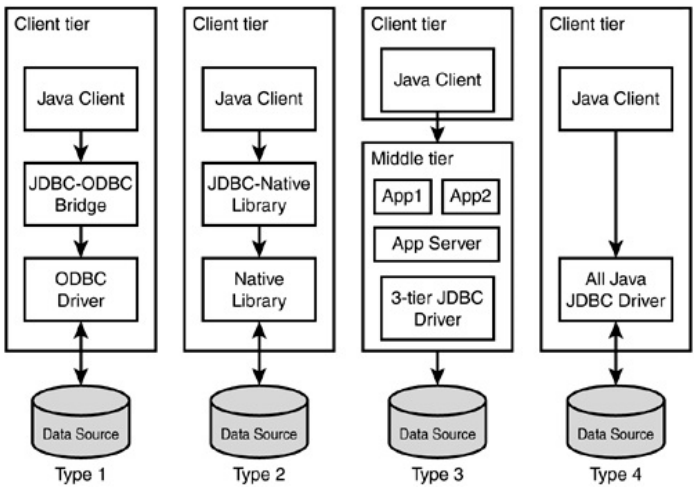
...

SQL Call-Level Interface [CLI]

JDBC [Java DataBase Connectivity]

Drivers JDBC

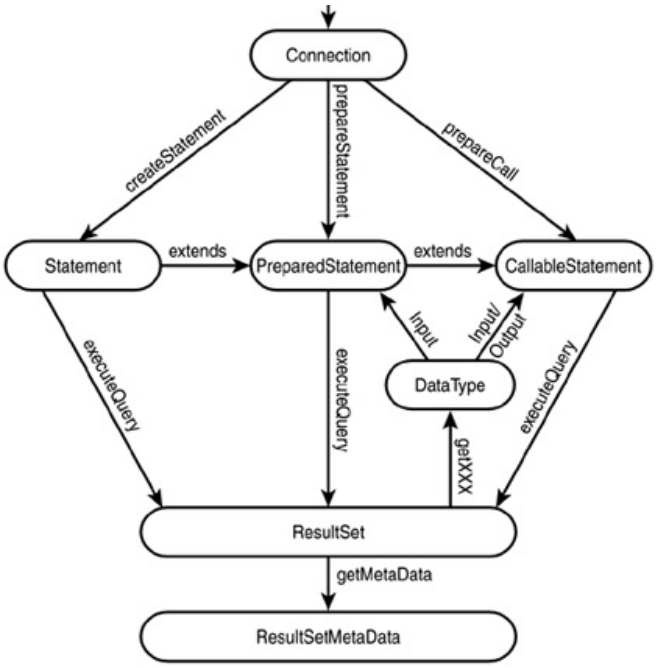
- JDBC-ODBC bridge
- Native API driver
- Middleware driver
- Pure Java driver



SQL Call-Level Interface [CLI]

JDBC [Java DataBase Connectivity]

Principales clases
(paquete java.sql)



SQL Call-Level Interface [CLI]

JDBC [Java DataBase Connectivity]

Establecimiento de conexiones

```
try {
  // 1. Cargamos el driver JDBC de nuestro DBMS (p.ej. Oracle)
  Class.forName("oracle.jdbc.driver.OracleDriver");
  // 2. Establecemos una conexión con la BD
  Connection connection = DriverManager.getConnection(
    "jdbc:oracle:thin:@localhost:1521:SID",
    "usuario", "password");
  …
} catch (ClassNotFoundException driverError) {
    // Driver no encontrado
} catch (SQLException sqlError) {
    // Error SQL (p.ej. usuario/clave incorrectas)
}
```

SQL Call-Level Interface [CLI]

JDBC [Java DataBase Connectivity]

Ejecución de sentencias SQL

```
Statement statement = connection.createStatement();
ResultSet set = statement.executeQuery("SELECT * FROM clients");

// Resultado de la consulta
while (set.next()) {
    … set.getString("name"); …
    … set.getString("address"); …
    … set.getDate("birthdate"); …
    … set.getBigDecimal("balance"); …
}
```


SQL Call-Level Interface [CLI]

JDBC [Java DataBase Connectivity]

Tipos de datos SQL

Tipo de dato SQL	Método JDBC
CHAR/VARCHAR	String getString()
DECIMAL/NUMERIC	java.math.BigDecimal getBigDecimal()
FLOAT/DOUBLE	double getDouble()
INTEGER	int getInt()
DATE	java.sql.Date getDate()
TIME	java.sql.Time getTime()
TIMESTAMP	java.sql.Timestamp getTimestamp()
BINARY	byte[] getBytes()
BLOB	java.io.InputStream getBinaryStream() java.sql.Blob getBlob()

SQL Call-Level Interface [CLI]

JDBC [Java DataBase Connectivity]

Peligro: Inyección de código SQL

```
String sql = "select * from user where username='" + username
           + "' and password='" + password + "'";
stmt = conn.createStatement();
rs = stmt.executeQuery(sql);
if (rs.next()) {
   out.println("Successfully logged in");
} else {
   out.println("Invalid username and/or password");
}
```

Entrada del usuario: username = admin' OR '1'='1

```
select * from user
where username='admin' OR '1'='1' and password=' '
```


SQL Call-Level Interface [CLI]

JDBC [Java DataBase Connectivity]

Para evitar ataques por inyección de código SQL…

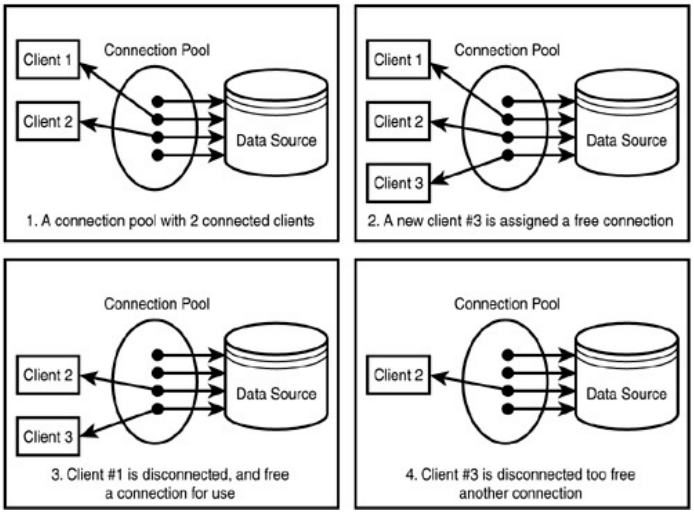
PreparedStatement

```
PreparedStatement statement = connection.prepareStatement (
  "UPDATE clients SET address = ? WHERE ID = ?");

statement.setString (1, "Nueva dirección");
statement.setInt (2, 123456 );
statement.execute();

// Resultado
… statement.getUpdateCount() …  // getResultSet() para consultas
```

SQL Call-Level Interface [CLI]

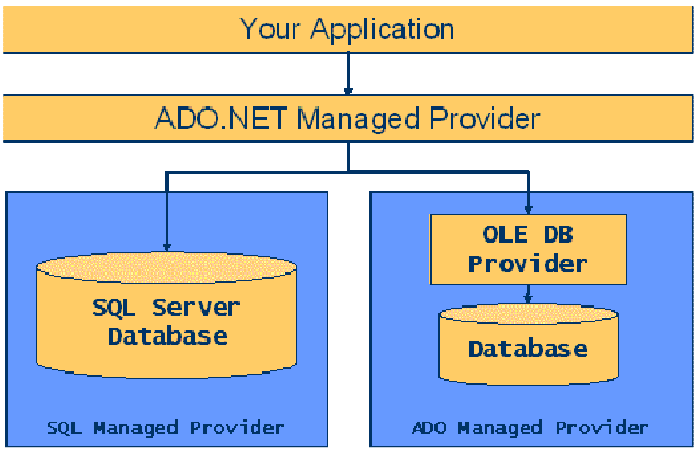
JDBC [Java DataBase Connectivity]

Pool de conexiones



SQL Call-Level Interface [CLI]

ADO.NET (plataforma .NET)



SQL Call-Level Interface [CLI]

ADO.NET (plataforma .NET)

Establecimiento de conexiones

```
string connectionString = "User ID=sa;Initial Catalog=MYDB;"
                        + "Data Source=MYSERVER";
SqlConnection connection = new SqlConnection(connectionString);
```

Ejecución de consultas (usando DataSet)

```
SqlDataAdapter adapter = new SqlDataAdapter();
DataSet dataset = new DataSet();

string sqlQuery = "SELECT * FROM Customers";
adapter.SelectCommand = new SqlCommand(sqlQuery, connection);

connection.Open();
adapter.Fill(dataset);
connection.Close();
```


SQL Call-Level Interface [CLI]

ADO.NET (plataforma .NET)

Ejecución de consultas (usando DataReader)

```
string sqlQuery = "SELECT Name FROM Users";
SqlCommand sqlCommand = new SqlCommand(sqlQuery, connection);

connection.Open();
SqlDataReader reader = sqlCommand.ExecuteReader();

while (reader.Read()) {
   … reader.GetString(0) …
}

myReader.Close();
connection.Close();
```

SQL Call-Level Interface [CLI]

ADO.NET (plataforma .NET)
Ejecución de sentencias SQL

```
string sqlInsert = "INSERT INTO Clients(Name) VALUES (@Name)";
SqlCommand sqlCommand = new SqlCommand(sqlInsert, connection);

SqlParameter  param = sqlCommand.Parameters.Add (
    new SqlParameter("@Name", SqlDbType.VarChar, 100));

param.Value = ...

connection.Open();
sqlCommand.ExecuteNonQuery();
connection.Close();
```


SQL Call-Level Interface [CLI]

SQLite (Android)
Esquema (local)

```
public final class FeedReaderContract {
  public FeedReaderContract() {}

  /* Inner class that defines the table contents */
  public static abstract class FeedEntry implements BaseColumns {
    public static final String TABLE_NAME = "entry";
    public static final String COLUMN_NAME_ENTRY_ID = "entryid";
    public static final String COLUMN_NAME_TITLE = "title";
    public static final String COLUMN_NAME_SUBTITLE = "subtitle";
    ...
  }
}
```

SQL Call-Level Interface [CLI]

SQLite (Android)

Base de datos (local): Creación

```
public class FeedReaderDbHelper extends SQLiteOpenHelper {
    // If you change the database schema,
    // you must increment the database version.
    public static final int DATABASE_VERSION = 1;
    public static final String DATABASE_NAME = "FeedReader.db";

    public FeedReaderDbHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }
    public void onCreate(SQLiteDatabase db) {
        db.execSQL(SQL_CREATE_TABLE);
    }
```


SQL Call-Level Interface [CLI]

SQLite (Android)

Base de datos (local): Actualizaciones

```
    public void onUpgrade
      (SQLiteDatabase db, int oldVersion, int newVersion) {
        // Data cache, just discard the data and start over
        db.execSQL(SQL_DROP_TABLE);
        onCreate(db);
    }
    public void onDowngrade
      (SQLiteDatabase db, int oldVersion, int newVersion) {
        onUpgrade(db, oldVersion, newVersion);
    }
}
```

SQL Call-Level Interface [CLI]

SQLite (Android)

Almacenamiento de datos

```
FeedReaderDbHelper mDbHelper
  = new FeedReaderDbHelper(getContext());
SQLiteDatabase db
  = mDbHelper.getWritableDatabase();

// Map of values, where column names are the keys
ContentValues values = new ContentValues();
values.put(FeedEntry.COLUMN_NAME_ENTRY_ID, id);
values.put(FeedEntry.COLUMN_NAME_TITLE, title);
values.put(FeedEntry.COLUMN_NAME_CONTENT, content);

// Insert the new row
long newRowId = db.insert(FeedEntry.TABLE_NAME,
         FeedEntry.COLUMN_NAME_NULLABLE, values);
```

SQL Call-Level Interface [CLI]

SQLite (Android)

Consulta de datos

```
SQLiteDatabase db = mDbHelper.getReadableDatabase();

Cursor c = db.query(       // .. or db.rawQuery(sqlStatement)
    FeedEntry.TABLE_NAME,  // The table to query
    projection,            // The columns to return
    selection,             // The columns for the WHERE clause
    selectionArgs,         // The values for the WHERE clause
    null,                  // don't group the rows
    null,                  // don't filter by row groups
    sortOrder );           // The sort order

cursor.moveToFirst();
long itemId = cursor.getLong(
    cursor.getColumnIndexOrThrow(FeedEntry._ID));
```

SQL Call-Level Interface [CLI]

SQLite (Android)

Borrado y actualizaciones

```
// 'where' clause
String selection = FeedEntry.COLUMN_NAME_ENTRY_ID + " LIKE ?";
String[] selectionArgs = { String.valueOf(rowId) };

// SQL statement
db.delete(table_name, selection, selectionArgs);

// New values
ContentValues values = new ContentValues();
values.put(FeedEntry.COLUMN_NAME_TITLE, title);
// SQL statement
int count = db.update(FeedReaderDbHelper.FeedEntry.TABLE_NAME,
    values, selection, selectionArgs);
```


SQL Call-Level Interface [CLI]

SQLite (Android)

Utilizando SQL directamente

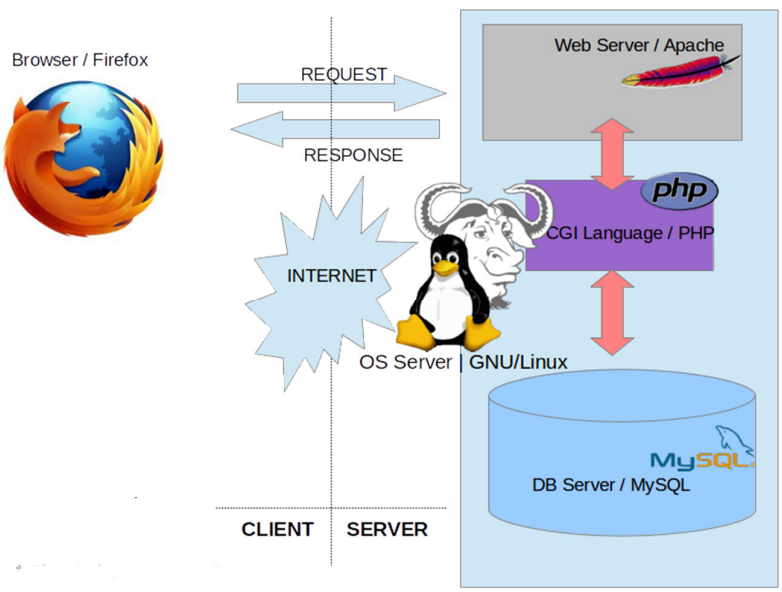
```
// SQL query
Cursor cursor = db.rawQuery (
                  "SELECT id, name FROM people WHERE id = ?",
                  new String[] {"1234"} );

// SQL statement
db.execSQL ( string );

   // NOTA: Google recomienda utilizar
   // - db.insert(String, String, ContentValues)
   // - db.update(String, ContentValues, String, String[])
   // - db.delete(String, String, String[])
```

SQL

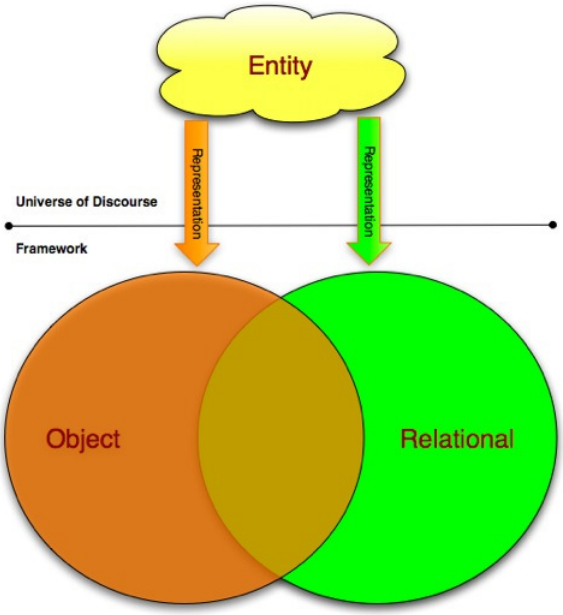
LAMP stack

SQL

LAMP stack

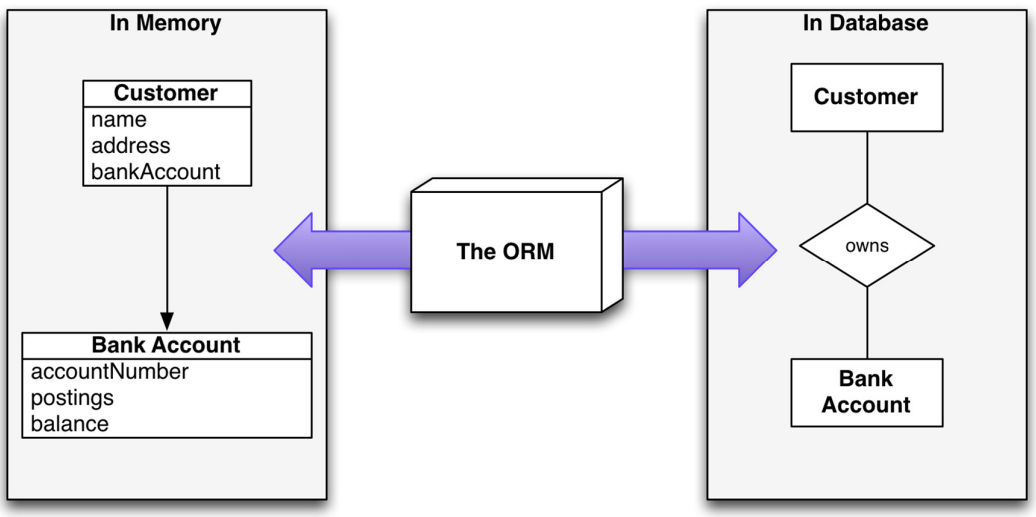


O/R Mapping



The object-relational impedance mismatch
http://impedancemismatch.com/

O/R Mapping



O/R Mapping

Ejemplo: C#

Usando ADO.NET (CLI estándar para .NET):

```
String sql = "SELECT ... FROM clientes WHERE id = 10";
DbCommand cmd = new DbCommand(connection, sql);
Result res = cmd.Execute();
String name = res[0]["FIRST_NAME"];
```

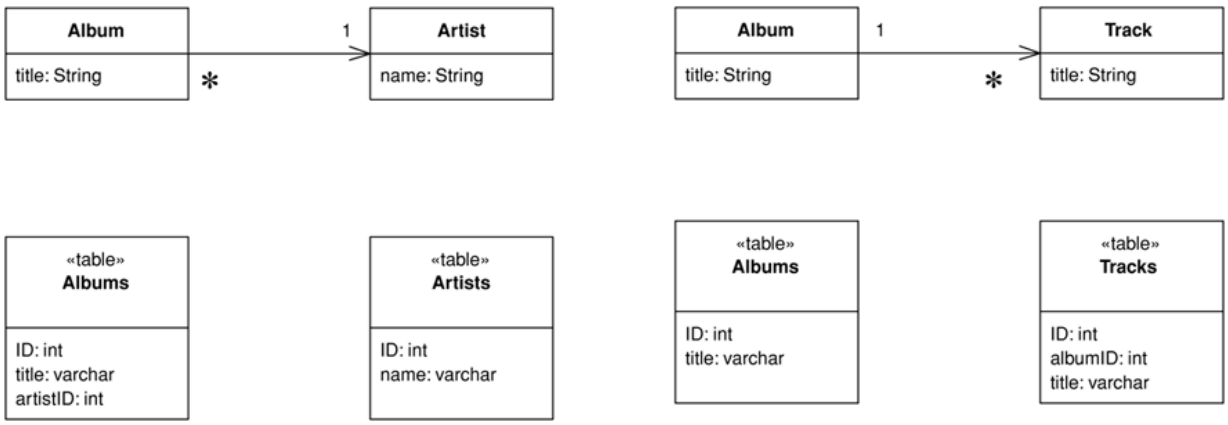
Usando ORM:

```
Client client = repository.GetClient(10);
String name = client.getFirstName();
```


O/R Mapping

Foreign-key mapping

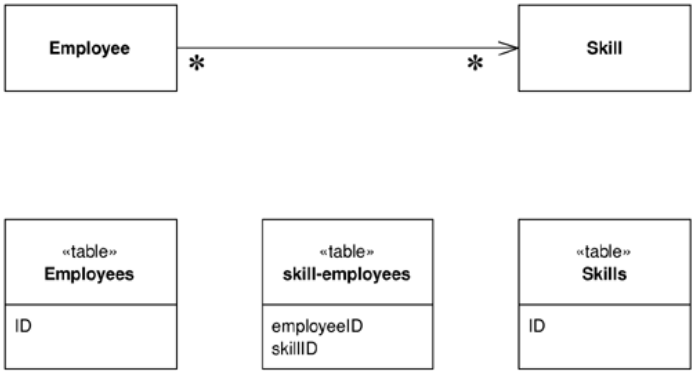
Relaciones muchos a uno y uno a muchos



O/R Mapping

Association table mapping

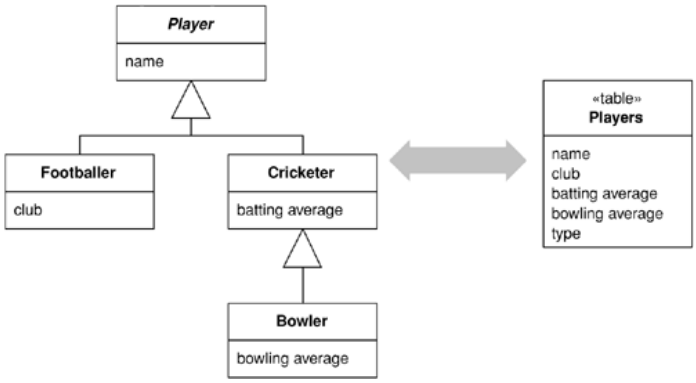
Relaciones muchos a muchos



O/R Mapping

Single table inheritance

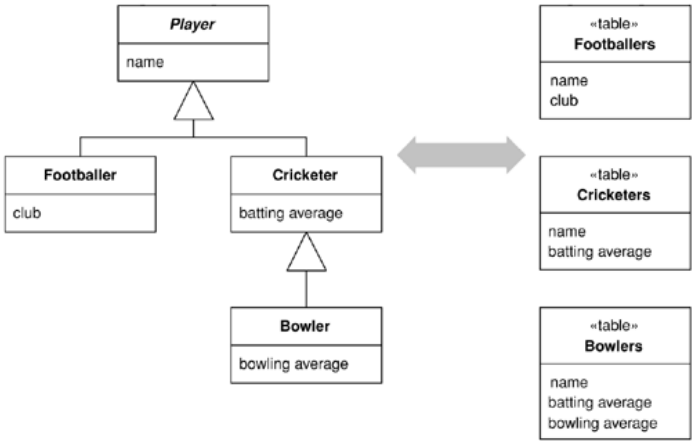
Herencia & relaciones "es-un" (1/3)



O/R Mapping

Concrete table inheritance

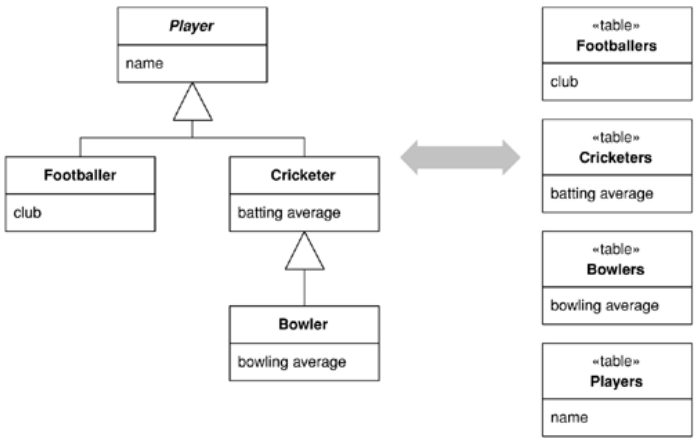
Herencia & relaciones "es-un" (2/3)



O/R Mapping

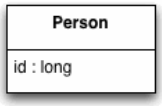
Class table inheritance

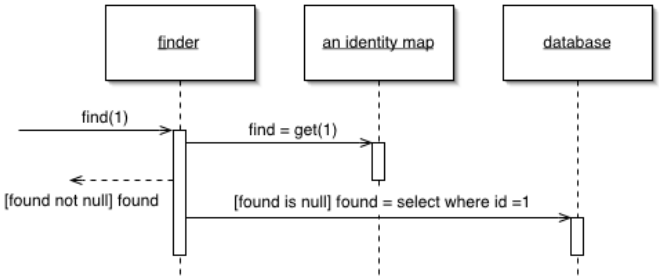
Herencia & relaciones "es-un" (3/3)



O/R Mapping

Gestión de identidades

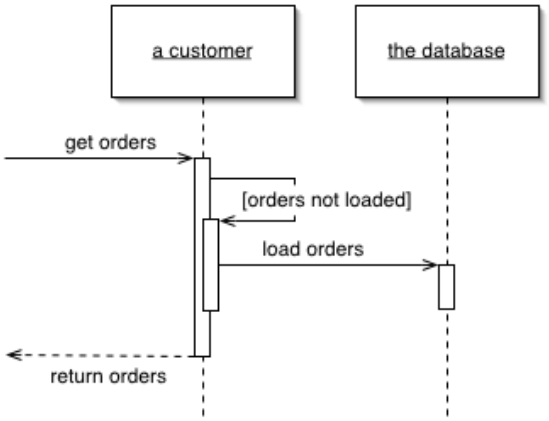




O/R Mapping

Lazy Load

Carga de los datos



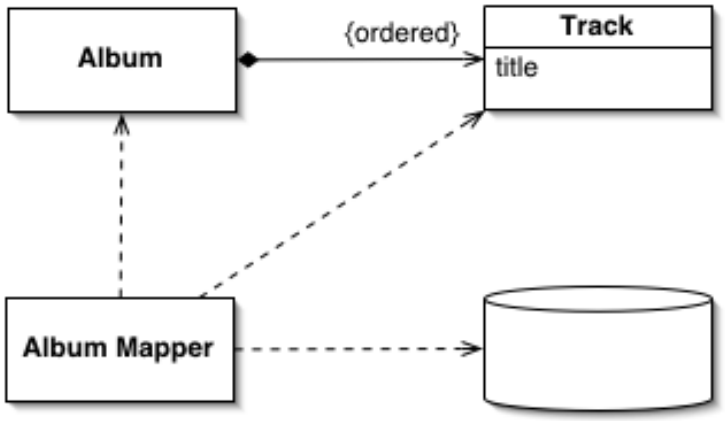
Alternativas de implementación:

- Inicialización (miembros null hasta que se accede a ellos).
- Proxy (se carga el objeto real la primera vez que se llama).
- Value holder (método getValue para acceder al objeto).
- Ghost (objeto sin datos, se rellena de golpe).

O/R Mapping

Dependent mapping

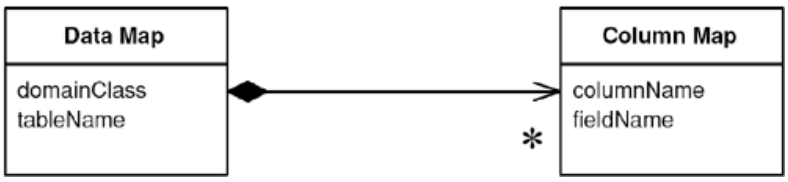
Carga de los datos para entidades débiles



O/R Mapping

Herramientas de O/R Mapping

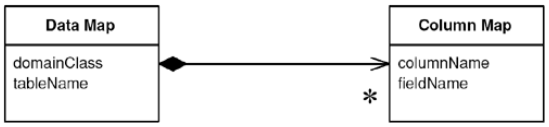
En vez de programar manualmente la correspondencia entre objetos y tablas, se pueden utilizar metadatos para especificar la correspondencia y automatizar el proceso.

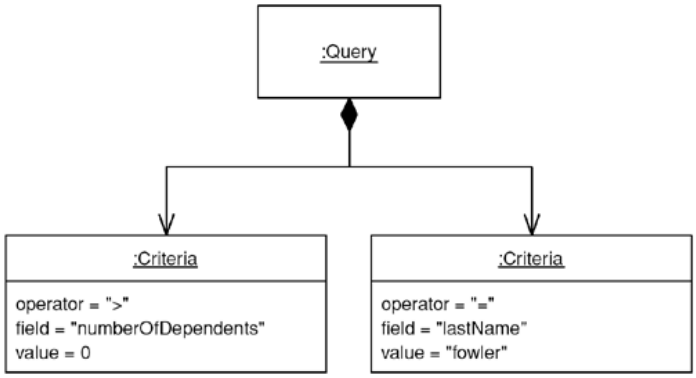


O/R Mapping

Herramientas de O/R Mapping

Realización de consultas:
"Query objects"

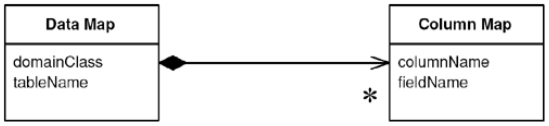


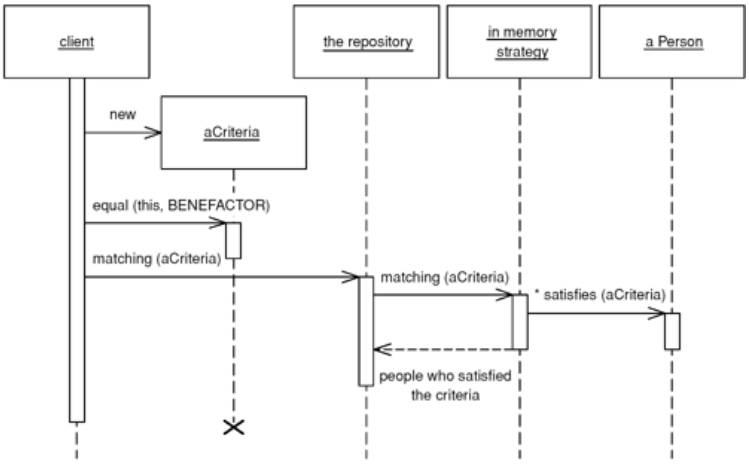


O/R Mapping

Herramientas de O/R Mapping

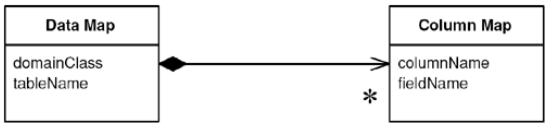
Almacenamiento de datos:
"Repositories"





O/R Mapping

Herramientas de O/R Mapping



- JDO [Java Data Objects]
- JPA [Java Persistence API]: Lenguaje de consulta JPQL
- DataNucleus (JDO & JPA), e.g. Google App Engine
- Hibernate (Java, JPA) & Nhibernate (.NET)
- iBATIS (Java, .NET, Ruby) → MyBATIS (Java)

Herramientas de O/R Mapping

DataNucleus (JDO & JPA)

DataNucleus

1. Clases en Java

```
public class Product
{
    String name;
    String description;
    double price;
    …
}
```

```
public class Book extends Product
{
    String author;
    String isbn;
    String publisher;
    …
}
```

Herramientas de O/R Mapping

DataNucleus (JDO & JPA)

2a. Persistencia (anotaciones)

```
@PersistenceCapable
public class Product
{
    String name;
    String description;
    double price;
    …
}
```

```
@PersistenceCapable
public class Book extends Product
{
    String author;
    String isbn;
    String publisher;
    …
}
```


Herramientas de O/R Mapping

DataNucleus (JDO & JPA)

2b. Persistencia (XML)

```
<?xml version="1.0"?>
<!DOCTYPE orm PUBLIC "-//Sun Microsystems, Inc.//DTD Java Data Objects Metadata 2.0//EN"
    "http://java.sun.com/dtd/orm_2_0.dtd">
<orm>
  <package name="org.datanucleus.samples.jdo.tutorial">
    <class name="Product" identity-type="datastore" table="JDO_PRODUCTS">
      <inheritance strategy="new-table"/>
      <field name="name">
         <column name="PRODUCT_NAME" length="100" jdbc-type="VARCHAR"/></field>
      <field name="description">
         <column length="255" jdbc-type="VARCHAR"/></field></class>
    <class name="Book" identity-type="datastore" table="JDO_BOOKS">
      <inheritance strategy="new-table"/>
      <field name="isbn">
          <column length="20" jdbc-type="VARCHAR"/></field>
      <field name="author">
          <column length="40" jdbc-type="VARCHAR"/></field>
      <field name="publisher">
          <column length="40" jdbc-type="VARCHAR"/></field></class>
    </package>
</orm>
```

Herramientas de O/R Mapping

DataNucleus (JDO & JPA)

3. Instrumentación de las clases: JDO "Enhancers"

Usando Ant

```
ant enhance
```

Usando Maven

```
mvn datanucleus:enhance
```

Manualmente

```
java –cp … org.datanucleus.enhancer.DataNucleusEnhancer *.java
```


Herramientas de O/R Mapping

DataNucleus (JDO & JPA)

4. Generación automática del esquema de la base de datos

Fichero de configuración (datanucleus.properties)

```
javax.jdo.PersistenceManagerFactoryClass=org.datanucleus.jdo.JDOPersistenceManagerFactory
javax.jdo.option.ConnectionDriverName=org.hsqldb.jdbcDriver
javax.jdo.option.ConnectionURL=jdbc:hsqldb:mem:nucleus1 javax.jdo.option.ConnectionUserName=sa
javax.jdo.option.ConnectionPassword= javax.jdo.option.Mapping=hsql
datanucleus.autoCreateSchema=true datanucleus.validateTables=false
datanucleus.validateConstraints=false
```

Usando Ant

```
ant createschema
```

Usando Maven

```
mvn datanucleus:schema-create
```

Manualmente

```
java –cp … org.datanucleus.store.rdbms.SchemaTool
     -props datanucleus.properties -create *.java
```

Herramientas de O/R Mapping

DataNucleus (JDO & JPA)

5. Uso desde una aplicación: CREATE

```
PersistenceManagerFactory pmf =
  JDOHelper.getPersistenceManagerFactory("datanucleus.properties");
PersistenceManager pm = pmf.getPersistenceManager();
Transaction tx=pm.currentTransaction();
try {
    tx.begin();
    Product product = new Product(“iPad", “Apple tablet", 649.99);
    pm.makePersistent(product);
    tx.commit();
} finally {
    if (tx.isActive()) tx.rollback();
    pm.close();
}
```


Herramientas de O/R Mapping

DataNucleus (JDO & JPA)

5. Uso desde una aplicación: READ

```
Transaction tx=pm.currentTransaction();
try {
    tx.begin();
    Extent e = pm.getExtent(Product.class, true);
    Query q = pm.newQuery(e,"price < 1500.00");
    q.setOrdering("price ascending");
    Collection c = (Collection) q.execute();
    for (Product p: c) { … }
    tx.commit();
} finally {
    if (tx.isActive()) tx.rollback();
    pm.close();
}
```

Herramientas de O/R Mapping

DataNucleus (JDO & JPA)

5. Uso desde una aplicación: DELETE

```
Transaction tx = pm.currentTransaction();
try {
    tx.begin();
    ...
    pm.deletePersistent(product);
    tx.commit();
} finally {
    if (tx.isActive()) tx.rollback();
    pm.close();
}
```


Herramientas de O/R Mapping

Hibernate

1. Clase en Java [POJO: Plain Old Java Object]

```
public class Employee
{
  private int id;
  private String firstName;
  private String lastName;
  private int level;
  …
  // Métodos get & set
  …
}
```

Herramientas de O/R Mapping

Hibernate

HIBERNATE

2. Tabla en la base de datos relacional (p.ej. MySQL)

```
create table EMPLOYEE (
   id INT NOT NULL auto_increment,
   first_name VARCHAR(20) default NULL,
   last_name  VARCHAR(20) default NULL,
   level INT default NULL,
   PRIMARY KEY (id)
);
```


Herramientas de O/R Mapping

Hibernate

3. Fichero de configuración (Employee.hbm.xml)

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD//EN"
 "http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">

<hibernate-mapping>
   <class name="Employee" table="EMPLOYEE">
      <meta attribute="class-description">…</meta>
      <id name="id" column="id" type="int">
         <generator class="native"/>
      </id>
      <property name="firstName" column="first_name" type="string"/>
      <property name="lastName" column="last_name" type="string"/>
      <property name="level" column="level" type="int"/>
   </class>
</hibernate-mapping>
```

HIBERNATE

Herramientas de O/R Mapping

Hibernate: CREATE

HIBERNATE

```
public int addEmployee (String fname, String lname, int level) {
      Session session = sessionFactory.openSession();
      Transaction tx = null;
      Integer employeeID = null;
      try{
         tx = session.beginTransaction();
         Employee employee = new Employee(fname, lname, level);
         employeeID = (Integer) session.save(employee);
         tx.commit();
      } catch (HibernateException e) {
         if (tx!=null) tx.rollback();
      } finally {
         session.close();
      }
      return employeeID;
}
```


Herramientas de O/R Mapping

Hibernate: READ

HIBERNATE

```
public List listEmployees () {
  List employees;
  Session session = factory.openSession();
  Transaction tx = null;
  try {
    tx = session.beginTransaction();
    employees = session.createQuery("FROM Employee").list();
    tx.commit();
  } catch (HibernateException e) {
    if (tx!=null) tx.rollback();
  } finally {
    session.close();
  }
  return employees:
}
//  for (Employee employee: employees) …
```

Herramientas de O/R Mapping

Hibernate: UPDATE

HIBERNATE

```
public void updateEmployee (int EmployeeID, int level) {
  Session session = factory.openSession();
  Transaction tx = null;
  try {
    tx = session.beginTransaction();
    Employee employee = (Employee)session.get(Employee.class, EmployeeID);
    employee.setLevel ( level );
    session.update(employee);
    tx.commit();
  } catch (HibernateException e) {
    if (tx!=null) tx.rollback();
  } finally {
    session.close();
  }
}
```


Herramientas de O/R Mapping

Hibernate: DELETE

HIBERNATE

```
public void deleteEmployee (int EmployeeID) {
  Session session = factory.openSession();
  Transaction tx = null;
  try {
    tx = session.beginTransaction();
    Employee employee = (Employee)session.get(Employee.class, EmployeeID);
    session.delete(employee);
    tx.commit();
  } catch (HibernateException e) {
    if (tx!=null) tx.rollback();
  } finally {
    session.close();
  }
}
```

Herramientas de O/R Mapping

iBATIS → MyBatis

MyBatis

Acopla objetos en Java
con sentencias SQL
o llamadas a procedimientos almacenados

Opción A: Usando anotaciones

```
public interface BlogMapper {
    @Select("select * from Blog where id = #{id}")
    Blog selectBlog(int id);
}

BlogMapper mapper = session.getMapper(BlogMapper.class);
Blog blog = mapper.selectBlog(101);
```

Herramientas de O/R Mapping

iBATIS → MyBatis

MyBatis

Acopla objetos en Java
con sentencias SQL
o llamadas a procedimientos almacenados

Opción B: Usando ficheros XML

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="BlogMapper">
    <select id="selectBlog" parameterType="int" resultType="Blog">
        select * from Blog where id = #{id}
    </select>
</mapper>

Blog blog = session.selectOne("BlogMapper.selectBlog", 101);
```

Herramientas de O/R Mapping

JPA [Java Persistence API]

Clases anotadas

```
@Entity
public class Book {
    @Id
    private Integer id;
    private String title;
    private String isbn;

    @ManyToOne
    private Publisher publisher;

    @ManyToMany
    private List<Author> authors;
}
```

```
@Entity
public class Publisher {
    @Id
    private Integer id;
    private String name;
    private String address;
    @OneToMany(mappedBy = "publisher")
    private List<Book> books;
}
```

```
@Entity
public class Author {
    @Id
    private Integer id;
    private String firstName;
    private String lastName;
    @ManyToMany
    private List<Book> books;
}
```


Herramientas de O/R Mapping

JPA [Java Persistence API]

Lenguaje de consulta JPQL

```
import javax.persistence.EntityManager;
import javax.persistence.Query;
…
public List<Author> getAuthorsByLastName(String lastName)
{
  String queryString = "SELECT a FROM Author a" +
                       " WHERE a.lastName IS NULL" +
                       " OR LOWER(a.lastName)=LOWER(:lastName)";
  Query query = getEntityManager().createQuery(queryString);
  query.setParameter("lastName", lastName);
  return query.getResultList();
}
```

Bibliografía recomendada

- Martin Fowler:
  Patterns of Enterprise Application Architecture
  Addison-Wesley, 2003.
  ISBN 0321127420
  http://martinfowler.com/eaaCatalog/