

# Bases de datos relacionales y orientadas a objeto

Tema 4. Acceso a datos

# Objetivos

- Características de las bases de datos objeto-relacionales.
- Gestores de bases de datos objeto relacionales.
- Características de las bases de datos orientadas a objetos.
- ObjectDB.

# Objetos en bases de datos

## Relacional

Fecha	Código	Ruta
01/10/01	24	I-95
15/12/01	23	I-495
17/03/02	24	I-66

Clave: 24

Código	Nombre actividad
23	Parcheado
24	Asfaltado
25	Sellado de grietas

Nombre actividad	Fecha	Ruta
Asfaltado	01/10/01	I-95
Asfaltado	17/03/02	I-66

# Objetos en bases de datos

## Orientado a objetos

Fecha	
Actividad	
Ruta	
Producción diaria	
Horas equipamiento	
Horas labor	

Objeto 1: Informe de mantenimiento

01/12/01
24
I-95
2.5
6
6

Instancia del objeto 1

Código actividad
Nombre actividad
Unidad de producción
Producción diaria media

Objeto 2: Actividad de mantenimiento

# Base de datos objeto relacional (BDOR)

- La principal característica de estas bases de datos es que el usuario puede crear sus propios tipos de datos y los métodos necesarios para trabajar con dichos tipos de datos.
- Son bases de datos que han evolucionado desde el modelo relacional incorporando conceptos del paradigma orientado a objetos.

# Ventajas de los SGBDOR

- Permiten gestionar tipos de datos complejos sin que esto conlleve un gran esfuerzo. Además, mantienen parte de la aplicación en el servidor de base de datos.
- Permiten almacenar datos complejos de una aplicación en la base de datos objeto-relacional sin necesidad de forzar los tipos de datos tradicionales.
- Son compatibles en sentido ascendente con los SGBDR tradicionales, lo cual aporta una gran facilidad de aprendizaje y utilización.

# Tipos de objetos: atómicos

- **Boolean:** un valor booleano es aquel que puede tomar uno de los siguientes valores: verdadero o falso.
- **Short:** entero con signo, normalmente de 8 o 16 bits.
- **Long:** entero con signo, normalmente de 32 o 64 bits.
- **Unsigned short:** entero sin signo, normalmente de 8 o 16 bits.
- **Unsigned long:** entero sin signo, normalmente de 32 o 64 bits.
- **Float:** valor real en coma flotante de simple precisión.

# Tipos de objetos: atómicos

- **Double:** valor real en coma flotante de doble precisión.
- **Octet:** almacén de 8 bits.
- **Char:** carácter ascii o unicode.
- **String:** cadena de caracteres.
- **Enum:** tipo enumerado donde los valores se especifican explícitamente cuando se declara el tipo.



# Tipos de objetos: colecciones

- **Set y bag:** grupo desordenado de objetos del mismo tipo. La diferencia está en que set no permite duplicados y bag sí.
- **List:** grupo ordenado de objetos del mismo tipo.
- **Array:** grupo ordenado de objetos del mismo tipo. Se puede acceder a ellos mediante su posición.

## Tipos de objetos: Tipos estructurados.

- **Date:** para las fechas.
- **Time:** para las horas.
- **Timestamp:** hora de una fecha.
- **Interval:** periodos de tiempo.

# PostgreSQL



Sistema de gestión de bases de datos relacional libre que destaca por agilizar la interacción de cliente, servidor y base de datos, ya que realiza la mayoría del trabajo referente a bases de datos cuando se hacen las peticiones.

# Ventajas de PostgreSQL

- Es un SGBD fácil de administrar.
- Facilidad de aprendizaje ya que usa sintaxis SQL estándar.
- Es multiplataforma.
- Permite la replicación de datos.
- Permite la realización de lo que se denomina copias de seguridad en caliente.
- Es un SGBDOR, por lo que entre sus mecanismos incluye herencia entre tablas, así como otros mecanismos orientados a objetos.

# Ventajas de PostgreSQL

- Es multiplataforma, por lo que está disponible para Linux, Unix, Windows, etc, en todas sus distribuciones y versiones.
- Permite, gracias al sistema MVCC, que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos.
- Permite la creación de tipos de datos, además de soportar distintos tipos diferentes de los tipos base como, por ejemplo, fechas, monedas, cadenas de bits, etc.
- Soporta juegos de caracteres internacionales, así como Unicode.


# Inconvenientes de PostgreSQL


- Actualmente no permite la realización de puntos de recuperación dentro de las transacciones por lo que, si se encuentra un fallo durante su ejecución, las transacciones abortan completamente.
- Aunque incorpora mecanismos de herencia y algunos otros de la orientación a objetos, pero cabe decir que no ofrece completas prestaciones de orientación a objetos.
- No soporta tablespaces para definir dónde almacenar la base de datos, el esquema, los índices, etc.
- Es más lento en inserciones y actualizaciones que MySQL, ya que cuenta con cabeceras de intersección que no tiene MySQL.

# Inconvenientes de PostgreSQL

- Aunque existen multitud de foros oficiales donde exponer dudas y cuestiones varias, no ofrece un sistema de ayuda.
- En comparación con otros SGBD como por ejemplo MySQL, consume más recursos.
- La sintaxis de algunos de sus comandos o sentencias no es nada intuitiva, lo cual dificulta su aprendizaje.

# Descarga

 [Home](#) [About](#) [Download](#) [Documentation](#) [Community](#) [Developers](#) [Support](#) [Donate](#) [Your account](#)

Search for... 

12th November 2020: [PostgreSQL 13.1, 12.5, 11.10, 10.15, 9.6.20, & 9.5.24 Released!](#)

## Quick Links

- Downloads
  - Packages
  - Source
- Software Catalogue
- File Browser

## Windows installers

### Interactive installer by EDB

**Download the installer** certified by EDB for all supported PostgreSQL versions.

This installer includes the PostgreSQL server, pgAdmin; a graphical tool for managing and developing your databases, and StackBuilder; a package manager that can be used to download and install additional PostgreSQL tools and drivers. Stackbuilder includes management, integration, migration, replication, geospatial, connectors and other tools.

This installer can run in graphical or silent install modes.

The installer is designed to be a straightforward, fast way to get up and running with PostgreSQL on Windows.

*Advanced users* can also download a **zip archive** of the binaries, without the installer. This download is intended for users who wish to include PostgreSQL as part of another application installer.

### Platform support

The installers are tested by EDB on the following platforms. They can generally be expected to run on other comparable versions:

PostgreSQL Version	64 Bit Windows Platforms	32 Bit Windows Platforms
13	2019, 2016	
12	2019, 2016, 2012 R2	
11	2019, 2016, 2012 R2	

<https://www.postgresql.org/download/windows/>



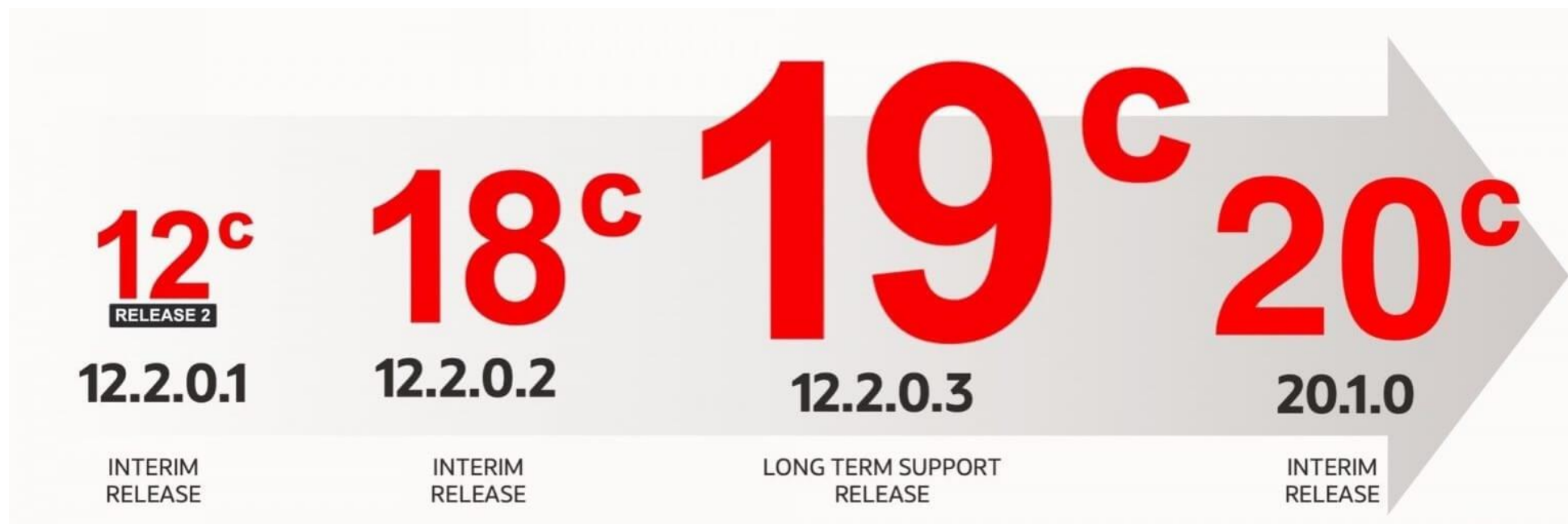


# Oracle

**19<sup>c</sup>** **ORACLE<sup>®</sup>**  
Database

Sistema de gestión de bases de datos objeto-relacional fabricado por Oracle Corporation, que destaca en el mercado por su portabilidad y compatibilidad con distintos sistemas.

# Oracle



# Ventajas de Oracle

- Utiliza la arquitectura cliente servidor.
- Destaca por gestionar a la perfección bases de datos que almacenan grandes cantidades de datos y a las que acceden multitud de usuarios concurrentes.
- Proporciona un alto rendimiento en las transacciones.
- Sistemas de alta disponibilidad.
- Disponibilidad controlada de los datos de las aplicaciones.

# Ventajas de Oracle

- Adaptación a estándares de la industria, como SQL-92.
- Gestión de la seguridad.
- Autogestión de la integridad de los datos.
- Opción distribuida.
- Conectabilidad.
- Replicación de entornos.

# Inconveniente de Oracle

Al ser un sistema propietario, como principal inconveniente cabe destacar su elevado precio, tanto al adquirirlo como en cuanto a soporte técnico y mantenimiento. Además, es necesario aplicar parches de seguridad debido a las vulnerabilidades en la seguridad de la plataforma.



<https://www.oracle.com/es/database/technologies/appdev/xe.html>

<https://www.oracle.com/tools/downloads/sqldev-downloads.html>

# SQL orientado a objetos

- En 1999 fue desarrollado el lenguaje SQL 3, que venía a mejorar las anteriores versiones e incorporar la orientación a objetos. Por tanto, este lenguaje es considerado como la base de la mayoría de los SGBD.
- Como principal característica se puede destacar la posibilidad de definir tipos de datos.

# SQL orientado a objetos

- Grandes objetos. SQL99 define dos tipos de objetos grandes, el llamado BLOB (Binary Large Object), adecuado para almacenar datos binarios como por ejemplo imágenes, videos,... y CLOB (Character Large Object), ideal para datos extensivos de tipos texto como por ejemplo informes, páginas web...
- Colecciones. Permite almacenar de forma directa colecciones enteras de datos tanto de tipo básico como de tipo estructurado.

# SQL orientado a objetos

- Tipos compuestos o estructurados. Gracias a la incorporación de estos tipos de datos es posible crear tipo de datos definidos por el usuario.
- Referencias a tipos estructurados. Se trata de un tipo especial que actúa como apuntador de tipos compuestos.



# SQL orientado a objetos

- La manipulación de estos tipos de datos no se realiza volcándolas a tipos de datos estándares, sino que se utilizan unas clases especiales (`java.sql.Blob`, `java.sql.Clob` o `java.sql.Array`) que actúan como punteros lógicos a la información deseada a la base de datos.
- Dispondremos del método `getBinaryStream` o `getAsciiStream`, respectivamente, para ir obteniendo la información parcialmente, a medida que la necesitamos. Los métodos devuelven un objeto `InputStream` que trataremos de forma habitual.

# Bases de datos orientadas a objetos

- En una base de datos orientada a objetos, la información se representa mediante objetos como los presentes en la programación orientada a objetos.
- Un ODBMS hace que los objetos de la base de datos aparezcan como objetos de un lenguaje de programación en uno o más lenguajes de programación a los que dé soporte.
- La iniciativa ODMG (Object Database Management Group) proporciona estándares sobre base de datos orientada a objetos.

# Bases de datos orientadas a objetos

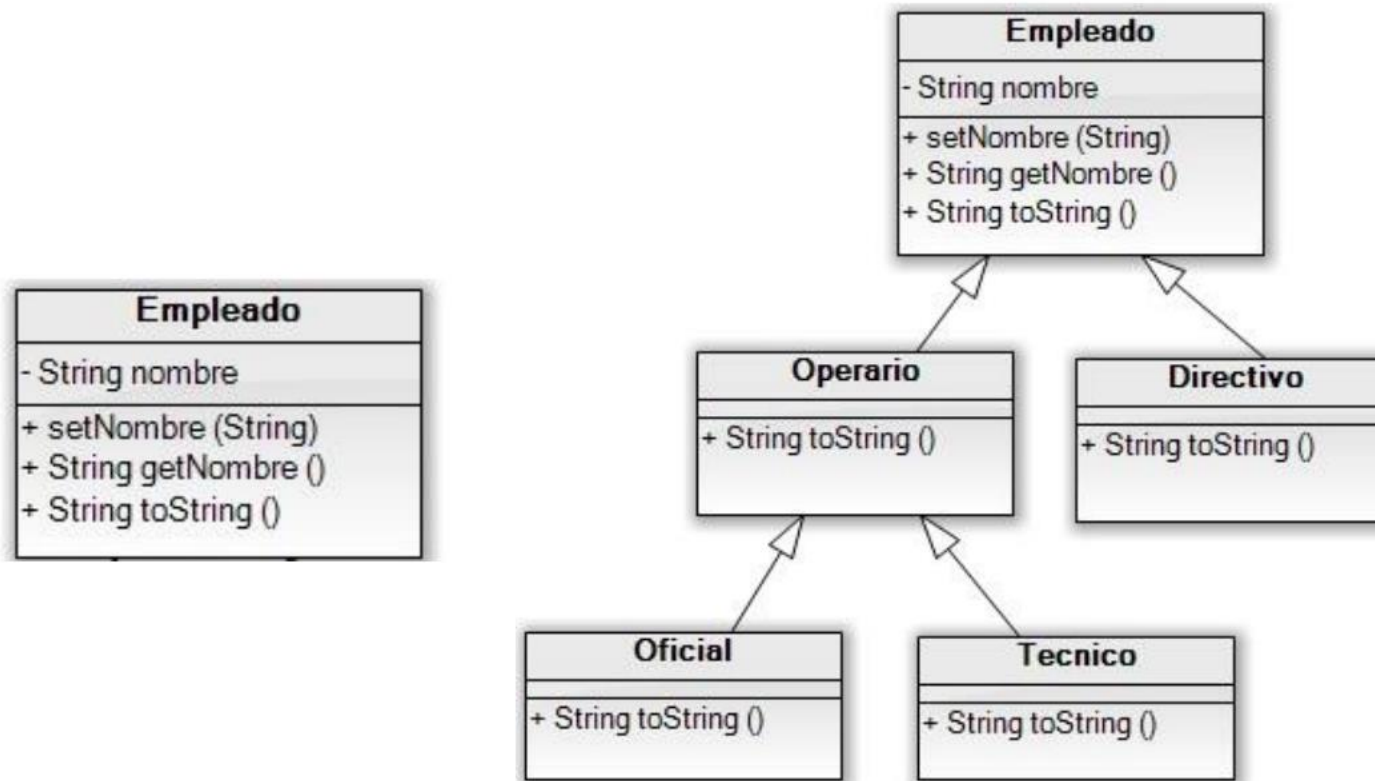
El ODMG define:

- Un modelo de objetos estándar para el diseño de estas BBDD.
- El lenguaje de definición de objetos o ODL (Object Definition Language).
- Lenguaje de consultas denominado **OQL**.

# Bases de datos orientadas a objetos

BD OBJETO-RELACIONAL	BD OBJETOS PUROS
<p>Estándar</p> <p>SQL: 1999 (SQL3), Melton 1999</p> <p>SQL: 2003 , Melton 2003</p> <p>Productos</p> <p>Oracle (a partir de la V8)</p> <p>POSTGRES</p> <p>Universal Server de INFORMIX.</p>	<p>Estándar</p> <p>ODMG-93, Cattell (1994), Cattell (1995)</p> <p>ODMG V.2.0, Cattell (1997)</p> <p>ODMG V.3.0, Cattell (2000)</p> <p>Productos</p> <p>ObjectStore de Object Design. Persistencia de objetos en C++ y Java.</p> <p>POET. Persistencia de objetos en C++ y Java.</p> <p>Gemstone de Servi Logia, Meier y Stone (1987). Persistencia de objetos Smalltalk, soporta también C++ y Java.</p>

# Bases de datos orientadas a objetos



# SGBDOO



**wakanda**  
JS.everywhere()



# ObjectDB

- ObjectDB es una base de datos orientada a objetos.
- Es software propietario, pero el fabricante, ObjectDB Software, ofrece una versión gratuita limitada que es suficiente para su estudio.
- Ofrece JPA como interfaz para el programador.
- Dispone también de un entorno gráfico para consultar y actualizar los datos. Este entorno gráfico admite la realización de queries con JPQL

# ObjectDB



*Java Persistence API (JPA) at the speed of light*

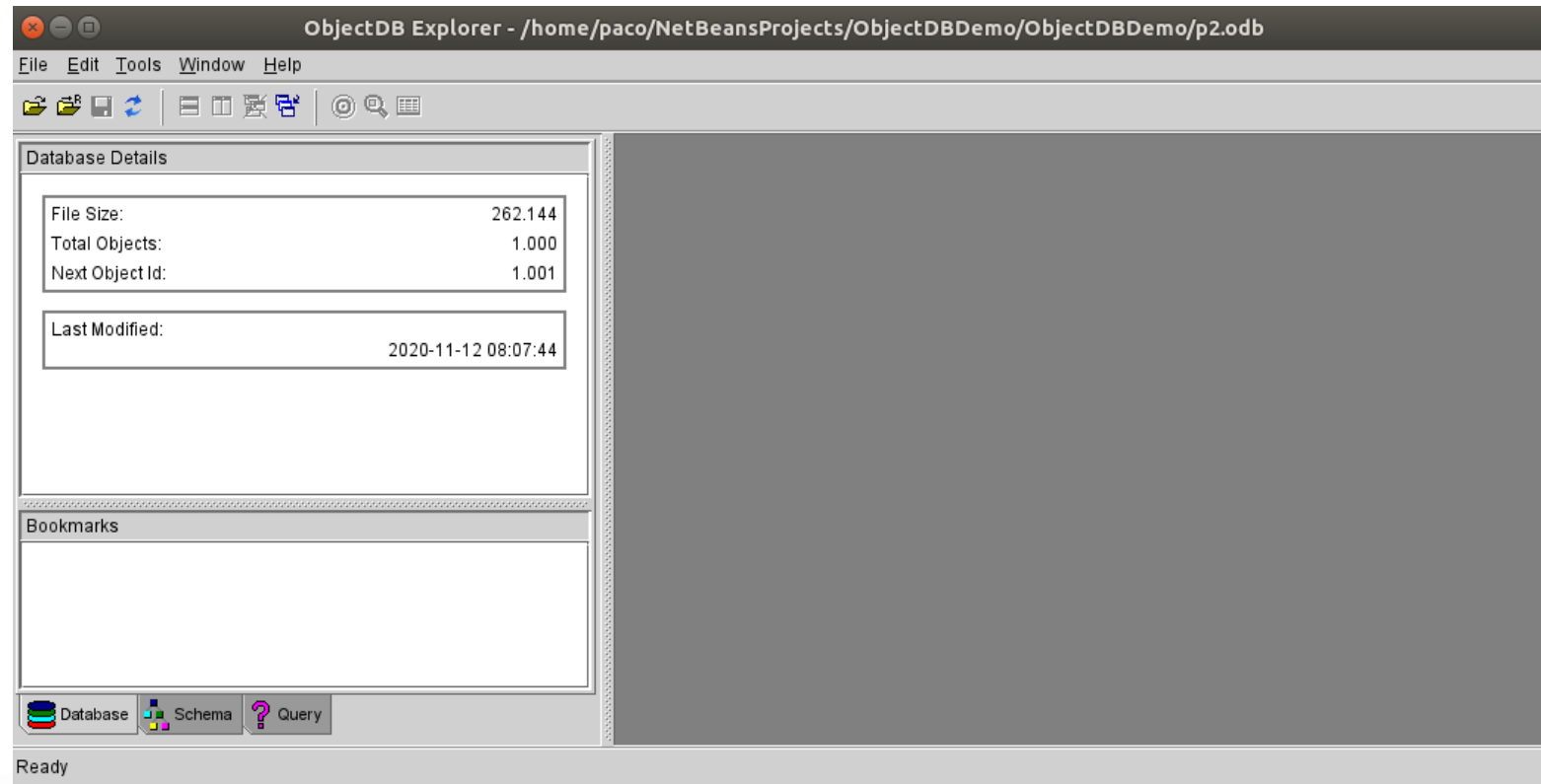
ObjectDB is about 10 times faster than other JPA/DBMS solutions - See the [JPA Performance Benchmark](#)

<https://www.objectdb.com/download>





# ObjectDB



# ObjectDB POM.xml

```
<repositories>
  <repository>
    <id>objectdb</id>
    <name>ObjectDB Repository</name>
    <url>https://m2.objectdb.com</url>
  </repository>
</repositories>
```

```
<dependencies>
  <dependency>
    <groupId>com.objectdb</groupId>
    <artifactId>objectdb</artifactId>
    <version>2.8.1</version>
  </dependency>
  <dependency>
    <groupId>org.eclipse.persistence</groupId>
    <artifactId>javax.persistence</artifactId>
    <version>2.1.0</version>
  </dependency>
  <dependency>
    <groupId>javax.transaction</groupId>
    <artifactId>jta</artifactId>
    <version>1.1</version>
  </dependency>
</dependencies>
```

# Definición de clase POJO

- `import javax.persistence.*;`
- `@Entity`
- `implements Serializable`
- `private static final long serialVersionUID = 1L;`
- `@Id @GeneratedValue`

```
package com.paco.objectdbdemo;

import java.io.Serializable;
import javax.persistence.*;

@Entity
public class Point implements Serializable {
    private static final long serialVersionUID = 1L;

    @Id @GeneratedValue
    private long id;

    private int x;
    private int y;

    public Point() {...2 lines }

    Point(int x, int y) {...4 lines }

    public Long getId() {...3 lines }

    public int getX() {...3 lines }

    public int getY() {...3 lines }

    @Override
    public String toString() {...3 lines }
}
```

# Uso del API de Persistence

OVERVIEW **PACKAGE** CLASS USE TREE DEPRECATED INDEX HELP

PREV PACKAGE NEXT PACKAGE FRAMES NO FRAMES ALL CLASSES

## Package javax.persistence

Java Persistence is the API for the management for persistence and object/relational mapping.

See: Description

### Interface Summary

Interface	Description
<b>AttributeConverter&lt;X,Y&gt;</b>	A class that implements this interface can be used to convert entity attribute state into database column representation and back again.
<b>AttributeNode&lt;T&gt;</b>	Represents an attribute node of an entity graph.
<b>Cache</b>	Interface used to interact with the second-level cache.
<b>EntityGraph&lt;T&gt;</b>	This type represents the root of an entity graph that will be used as a template to define the attribute nodes and boundaries of a graph of entities and entity relationships.
<b>EntityManager</b>	Interface used to interact with the persistence context.
<b>EntityManagerFactory</b>	Interface used to interact with the entity manager factory for the persistence unit.
<b>EntityTransaction</b>	Interface used to control transactions on resource-local entity managers.
<b>Parameter&lt;T&gt;</b>	Type for query parameter objects.
<b>PersistenceUnitUtil</b>	Utility interface between the application and the persistence provider managing the persistence unit.
<b>PersistenceUtil</b>	Utility interface between the application and the persistence provider(s).
<b>Query</b>	Interface used to control query execution.
<b>StoredProcedureQuery</b>	Interface used to control stored procedure query execution.
<b>Subgraph&lt;T&gt;</b>	This type represents a subgraph for an attribute node that corresponds to a Managed Type.



# Uso del API de Persistence

```
EntityManagerFactory emf =  
    Persistence.createEntityManagerFactory("p2.odb");  
EntityManager em = emf.createEntityManager();  
  
em.getTransaction().begin();  
/* Operaciones */  
em.getTransaction().commit();
```

# Uso del API de Persistence

```
Query query = em.createQuery("SELECT c FROM Country c");  
List results = query.getResultList();
```

```
Query q1 = em.createQuery("SELECT COUNT(p) FROM Point p");  
System.out.println("Total: " + q1.getSingleResult());
```

# Uso del API de Persistence

```
TypedQuery<Point> query =  
    em.createQuery("SELECT p FROM Point p", Point.class);  
List<Point> results = query.getResultList();
```

```
TypedQuery<Point> query =  
    em.createQuery("SELECT p FROM Point p WHERE p.x>p.y", Point.class);  
List<Point> results = query.getResultList();
```

```
TypedQuery<models.Contacto> query = em.createQuery(  
    "SELECT c FROM Contacto c WHERE c.name = :name", models.Contacto.class);  
query.setParameter("name", "mi nombre").getSingleResult();
```

# Agenda de contactos personal

Crear una aplicación en JavaFX, usando como base de datos ObjectDB que permita la gestión de contactos personales, incluyendo los siguientes datos: nombre, telefono, correo, y empresa.

La aplicación debera poder crear, editar, mostrar y borrar contactos.