Laboratorio IV

Trabajo Práctico

Capas de Persistencia

1. Qué es Persistencia?

La persistencia es la parte más crítica en una aplicación de software. Si la aplicación está diseñada bajo la orientación a objetos, la persistencia se logra por: serialización del objeto ó, almacenando en una base de datos.

"Persistencia es la habilidad que tiene un objeto de sobrevivir al ciclo de vida del proceso en el que reside. Los objetos que mueren al final de un proceso se llaman transitorios."

Para reducir la incongruencia entre un modelo orientado a objetos y una base de datos relacional recurrimos a una capa auxiliar que mapeará entre los éstos dos mundos, adaptándolo según las especificaciones hechas. Esa capa auxiliar se denomina ORM, object relational mapping.

Si queremos cargar un objeto persistido en memoria, los pasos a seguir serían: abrir la conexión a la base de datos, crear una sentencia sql parametrizada, llenar los parámetros (por ejemplo la clave primaria), recién allí ejecutarlo como una transacción y luego cerrar la conexión a la base de datos.

3. El auge de la orientación a objetos

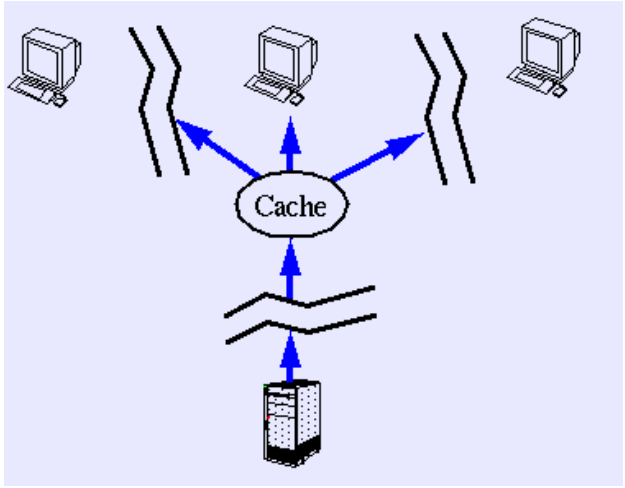
Que pasó con el modelo relacional frente al auge del desarrollo orientado a objetos. Analogías entre uno y otro. Qué es lo que no se persiste en OO? (métodos)

.Las Bases de Datos Orientadas a Objetos permiten que múltiples usuarios compartan objetos complejos y los manipulen en un ambiente seguro y estructurado. Las bases de datos convencionales fueron diseñadas para manejar tipos de datos alfanuméricos y por esto difícilmente pueden manipular objetos y métodos (los métodos son los comportamientos definidos de los objetos)

4. Capa de persistencia/Características

- Patron CRUD. (Create/Read/Update/Delete)
  - Usuario

- Caché.



Capa de persistencia/Características

- Concurrencia.
  - Bloqueo Optimista.
  - Bloqueo Pesimista.
- Transaccion.
  - Cada transacción requiere una sesión separada.
  - Aislación en la transacción y bloqueo basado en páginas.

Capa de persistencia/Características

- Identidad.

Ej_1:

```
Persona prsnPerezJuan = new Persona(”Perez”, “Juan”)
Persona prsnPerezAlberto = new Persona(”Perez”, “Alberto”)
Persona prsnPerez = ORM.LoadByApellido(“Perez”) //¿?
```

Ej_2:

```
Persona prsnJuan = (Persona) ORM.Load(“PersonaOID”,100)
Persona prsnPerez = (Persona) ORM.Load(“PersonaOID”,100)
If (prsnJuan != prsnPerez) // problemas !!!
```

Capa de persistencia/Caracterísitca

- Carga de las relaciones.
  - Carga Retardada.
  - Carga Directa.
- Proxy. (Listado y Selección, Reportes)
  - Evitar mostrar al usuario todos los objetos al principio.
  - Proveer el manejo de solo lectura en masa.
  - La consulta a la base de datos debería usar un cursor.
  - Unir datos a través de múltiples tablas.

Capa de persistencia/Caracterísitca

- Referencia circular.

- Información oculta.

5.3 Lenguajes de Consulta y Manipulación

Propios del entorno (ámbito general)

- .Net Framework: **LinQ**

Propios de una herramienta (ámbito del framework)

- Hibernate: **HQL**

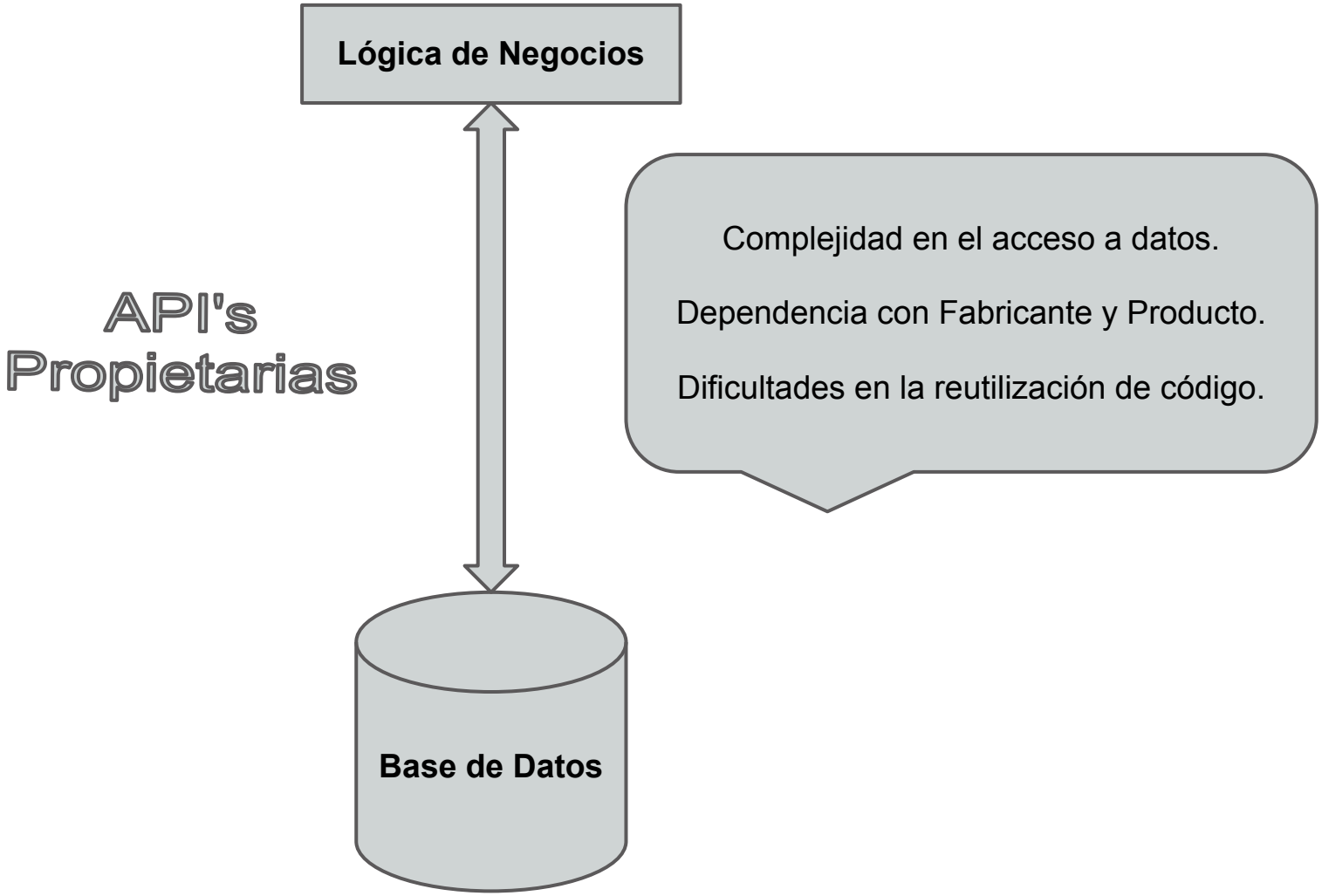
5.4 Actualización en Cascada

5.5 Operaciones En Masa

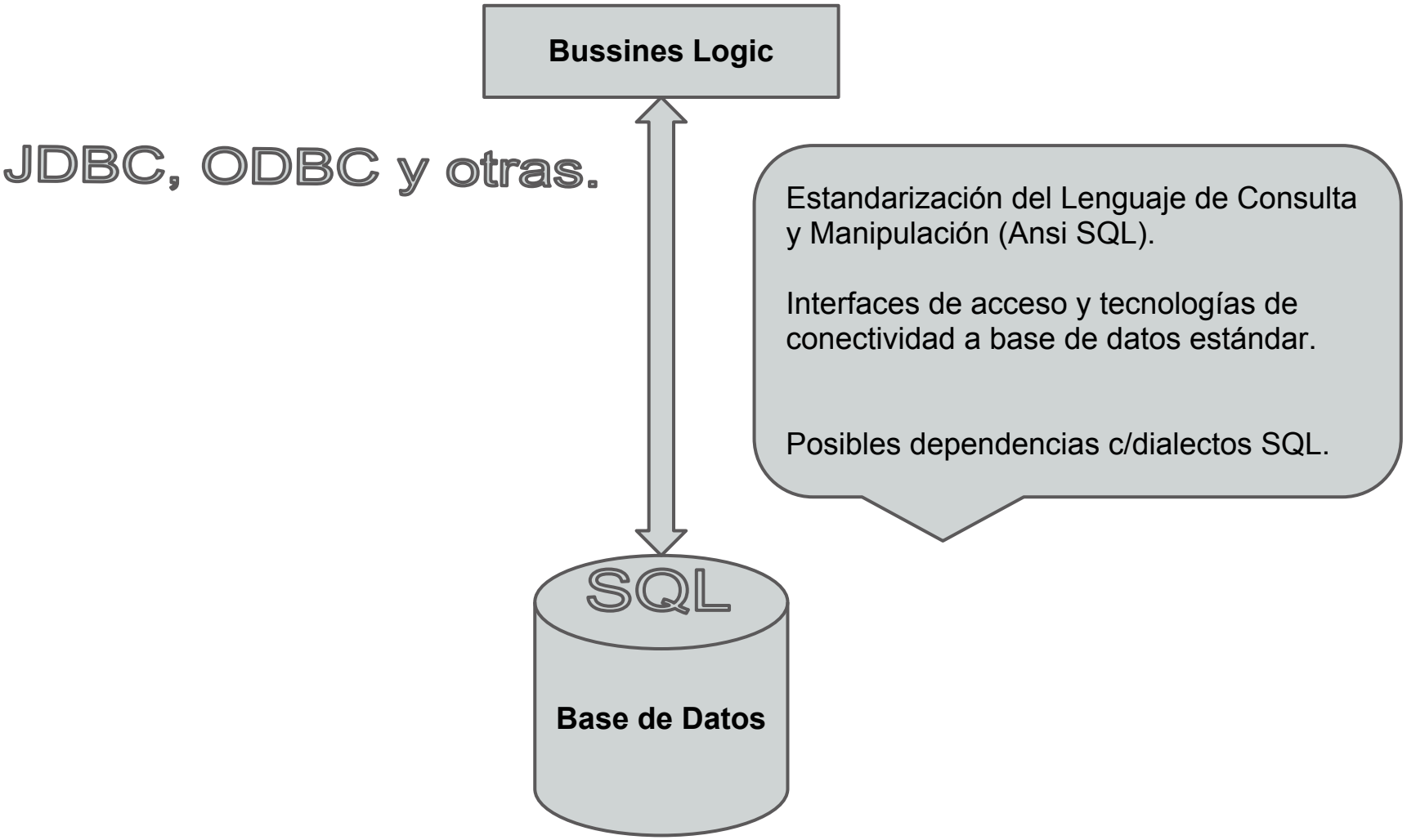
6. Persistencia en un modelo de N-Capas

- La capa de **Abstracción a Datos (Data Source Layer)** será la interfaz con la capa de la lógica de negocio, haciendo las veces de “facade”. Se implementa en múltiples capas y mediante el patrón Data Access Object (DAO), Repositorios y apoyandose en otros patrones de implementación también.
    - La capa de **Framework de Persistencia** será la interfaz con el gestor de Base de Datos ocupándose de la gestión de los datos mediante un API ó interfaz que proporcione el lenguaje de programación, que en Java es generalmente JDBC.
    - A más bajo nivel, un **Driver** ú Conector se ocupa de la comunicación con la propia Base de Datos utilizando un Driver específico para la misma.
- Evolución desde el modelo Cliente-Servidor. Problema del cliente gordo.

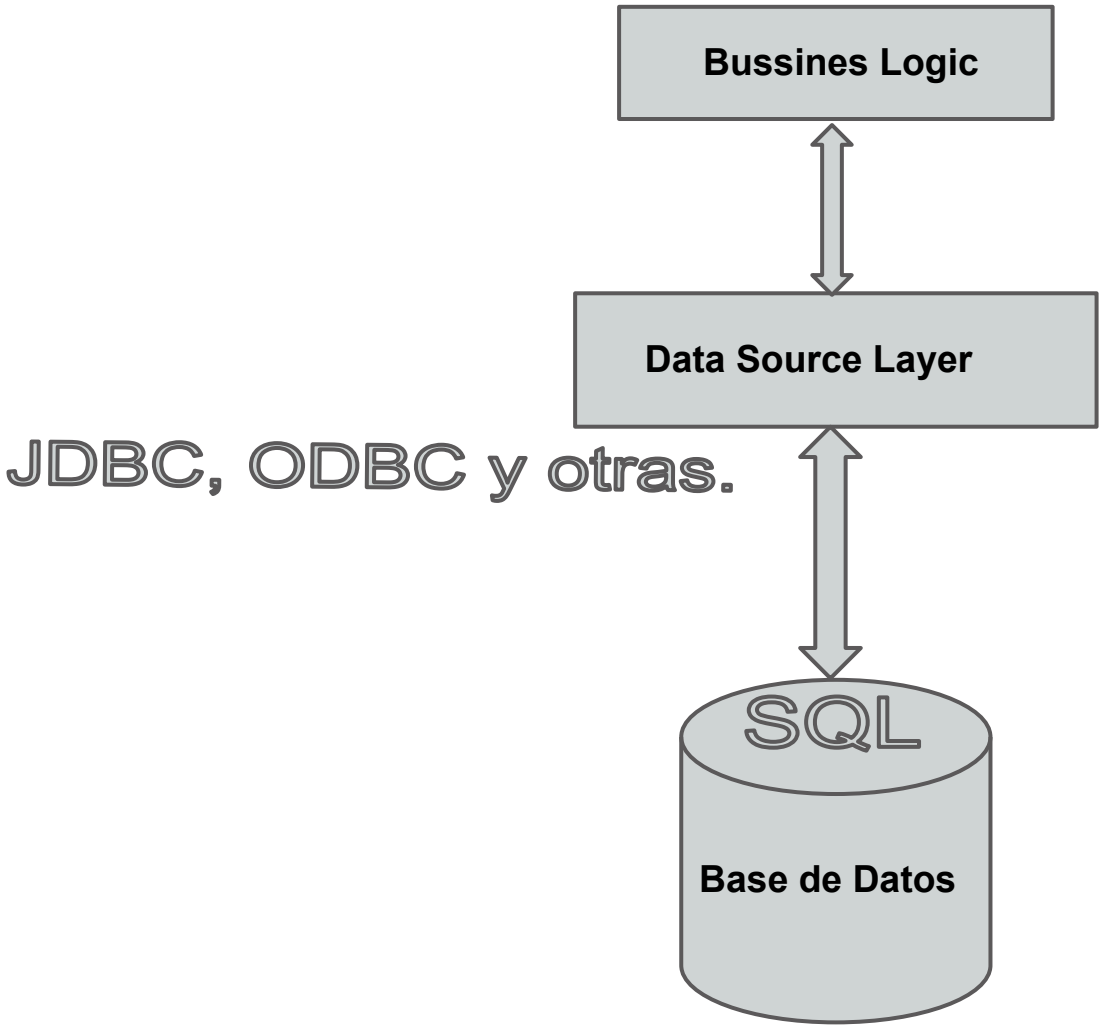
Estandares de conectividad



Estándares de conectividad



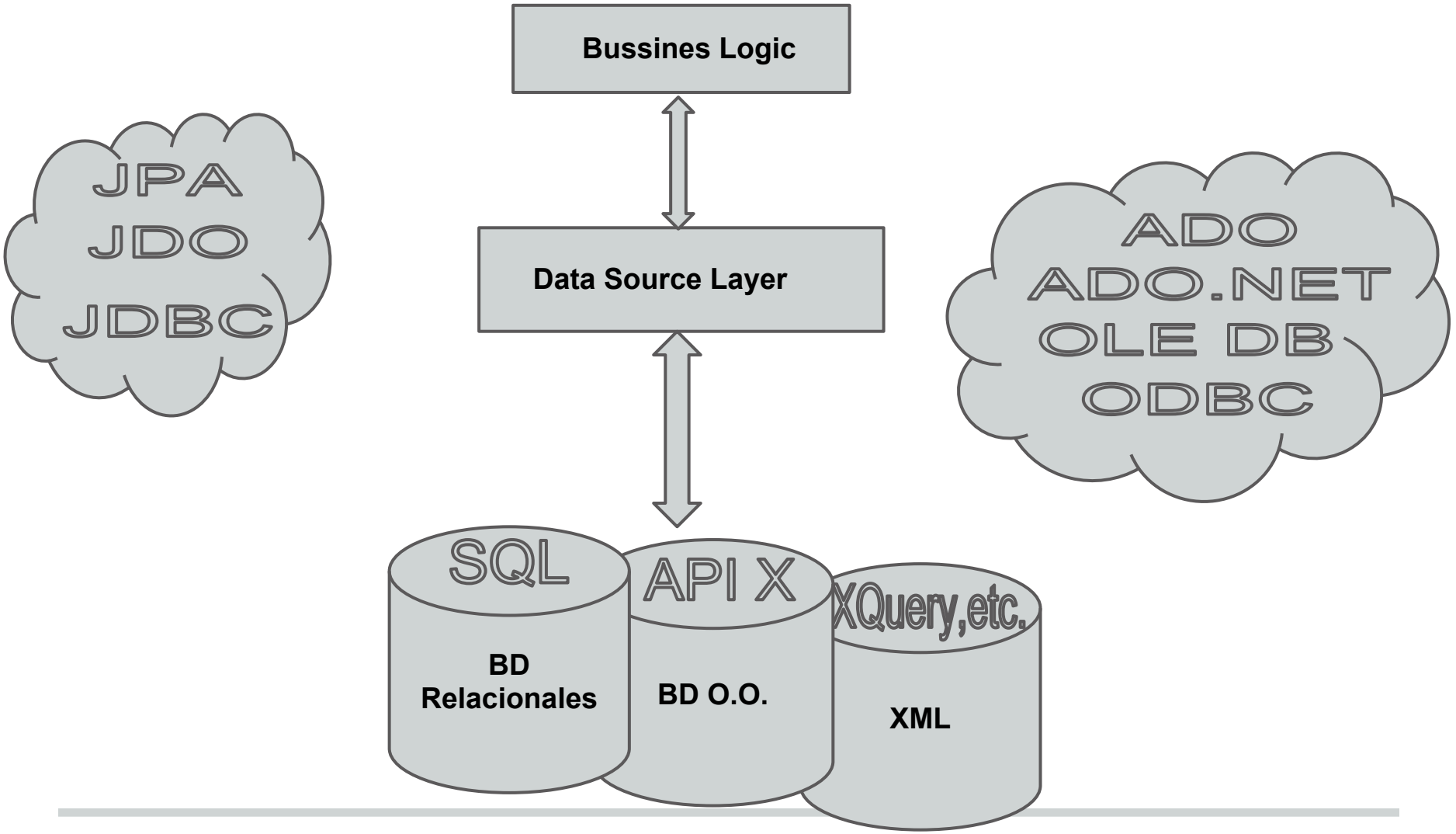
Introducción de las capas de persistencia



Se agrega un primer nivel de indirección.

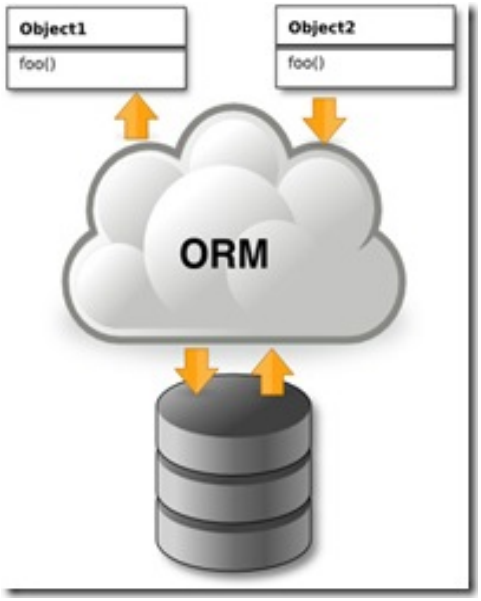
Se intenta abstraer la lógica de negocios del motor en particular que se está usando.

APIs de Persistencia y Tecnologías de Acceso a Datos



Estándares en la capa de persistencia

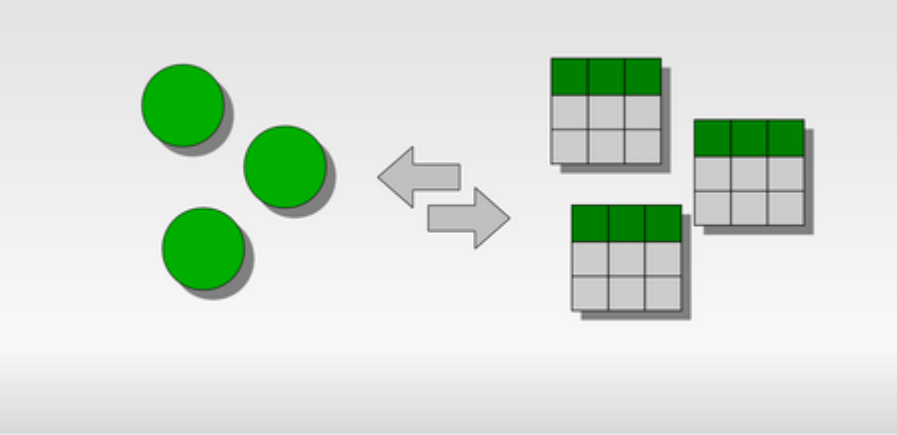
Mapeo Objeto-Relacional



Técnica de programación que me permite convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y el utilizado en una base de datos relacional, utilizando un motor de persistencia.

¿Que me permite realizar ORM?

- Mapear clases a tablas: propiedad a columna, clase a tabla.
- Persistir objetos. A través de un método Orm.Save(objeto). Encargándose de generar el SQL correspondiente.
- Recuperar objetos persistidos. A través de un método objeto = Orm.Load (objeto.class, clave_primaria).
- Recuperar una lista de objetos a partir de un lenguaje de consulta especial. A través de un método.



Ventajas

- Rapidez en el desarrollo.
- Abstracción de la base de datos.
- Seguridad.
- Mantenimiento del código.
- Lenguaje propio para realizar las consultas.

Desventajas

- Configuración.
- Genera una gran dependencia con el Framework utilizado.
- SQL dinámicamente
- Rendimiento
- Caches

Frameworks de Persistencia (Paquetes Comerciales)

Entity Framework

propel
smart, easy object persistence

doctrine

HIBERNATE

NHIBERNATE

Hibernate

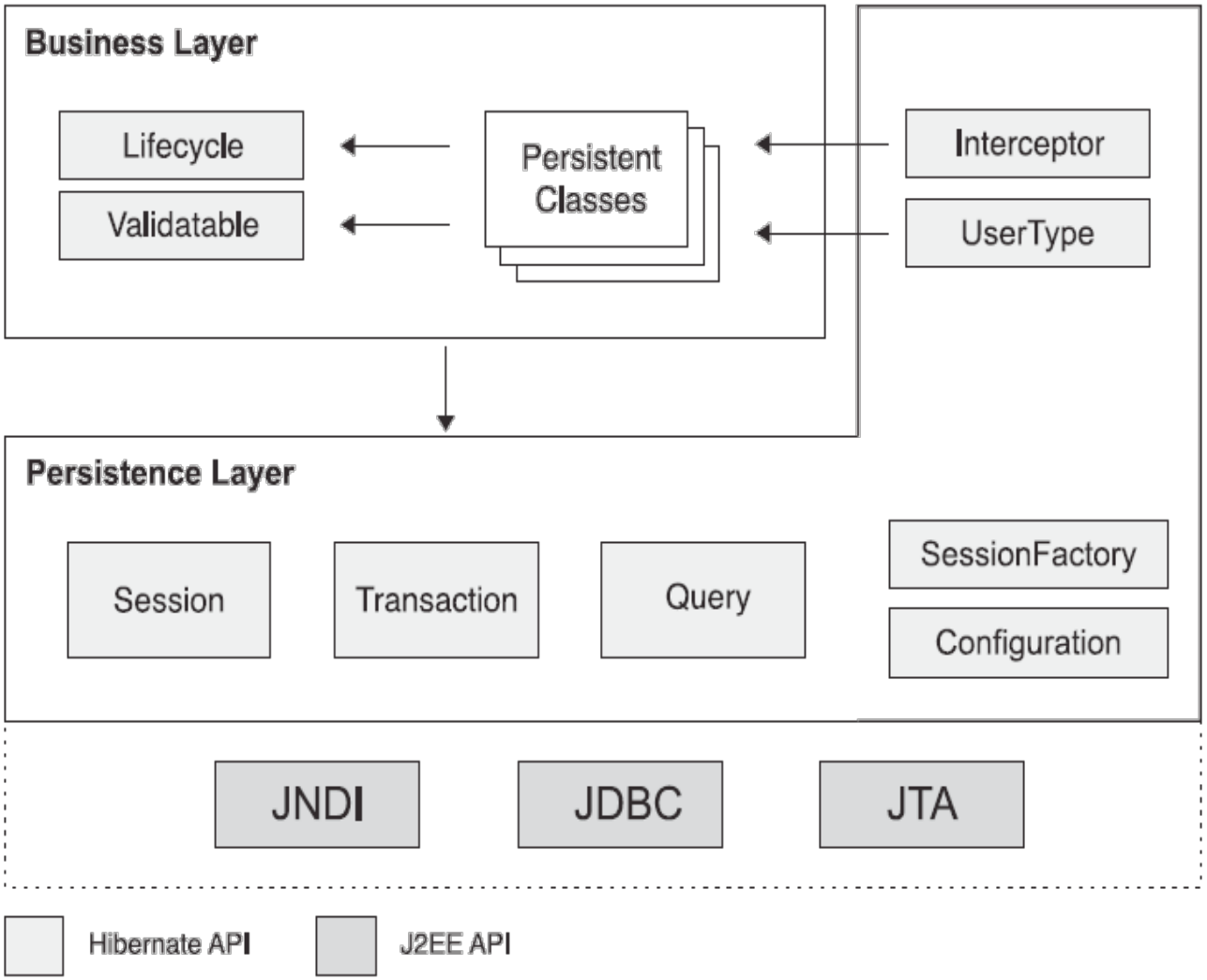
Es una herrmienta ORM para la tecnología JAVA y disponible también para la tecnología .NET con el nombre de Nhibernate.

Es software libre bajo la licencia GNU LGPL.

Es el mapeador objeto-relacional de código abierto más maduro y más completo que hay ahora.

Se utiliza muy ampliamente y es desarrollada activamente. Hibernate también soporta una de las mayores comunidades de Open Source.

Hibernate


Business Layer
Lifecycle
Validatable
Persistent Classes
Interceptor
UserType
Persistence Layer
Session
Transaction
Query
SessionFactory
Configuration
JNDI
JDBC
JTA
Hibernate API
J2EE API

Ejemplo

Preguntas?