Bases de Datos Orientadas a Objetos

(... o bien un esfuerzo por aclarar el ***arroz con mango*** de los SGBDOO)

Universidad de los Andes
Demián Gutierrez
Abril 2011

Condiciones en las que nacen los SGBDR (años 60-70)

UNIVERSIDAD DE LOS ANDES VENEZUELA

Uniformidad

Muchos datos estructurados de forma similar

Orientación a Registros

Datos básicos organizados en registros de longitud fija

Datos Pequeños

Registros cortos, de 80 bytes o menos

Campos Atómicos

Cortos, indivisibles y de longitud fija

Condiciones en las que nacen los SGBDR (años 60-70)

UNIVERSIDAD DE LOS ANDES VENEZUELA

Transacciones Cortas

Tiempo de ejecución medido en fracciones de segundos / sin interacción del usuario

Esquemas Conceptuales Estáticos

El esquema de la BD se cambia con muy poca frecuencia

Procesamiento por Lotes

Poca interacción con el usuario

Aplicaciones Casi Inexistentes

O a muy bajo nivel, embebidas e implementadas en el SGBD

Condiciones en las que nacen los SGBDR (con el tiempo...)

UNIVERSIDAD DE LOS ANDES VENEZUELA

con el tiempo...

debido a la mayor capacidad de cómputo de los procesadores, mayor cantidad de memoria principal y secundaria, y a la reducción generalizada de los costos del hardware fue posible desarrollar nuevos tipos de aplicaciones

Condiciones en las que nacen los SGBDR (con el tiempo...)

UNIVERSIDAD DE LOS ANDES VENEZUELA

- Diseño Asistido por Computador (CAD)
- Ingeniería de Software Asistida por Computador (CASE)
- Bases de Datos de Multimedios
- Sistemas de Información de Oficina
- Sistemas de Información / Aplicaciones Empresariales
- Sistemas Expertos de Bases de Datos

Otras (El cielo es el límite)

Condiciones en las que nacen los SGBDR (con el tiempo...)

UNIVERSIDAD DE LOS ANDES VENEZUELA

nuevos tipos de aplicaciones, más capacidad de cómputo, más memoria principal y secundaria, implica que se producen cambios en la forma en que se ven y usan los SGBD

... nuevas Necesidades

UNIVERSIDAD DE LOS ANDES VENEZUELA

Nuevas capas de aplicación

Reglas más complejas asociadas mucho más a nivel de aplicación (general) que a nivel de tuplas

Mayor interacción (y más compleja) entre el usuario y la aplicación

Transacciones de larga duración (en parte por el punto anterior)

... nuevas Necesidades

UNIVERSIDAD DE LOS ANDES VENEZUELA

Información más complejas -> **Objetos**

Comportamiento asociado a la información -> **Objetos**

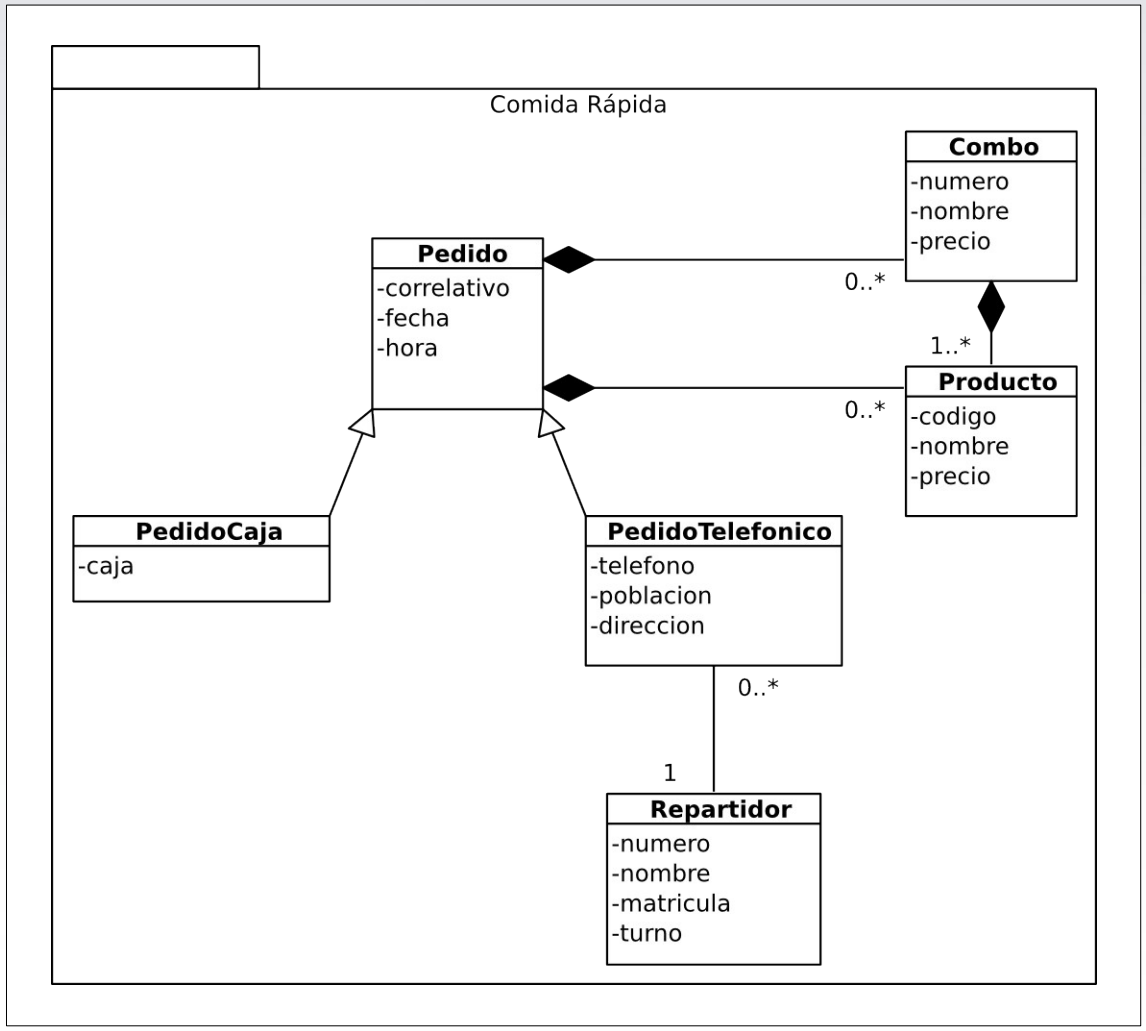
Reducir la impedancia entre las nuevas capas de aplicación (**Objetos**) y el almacenamiento persistente de los datos (**Relacional**)

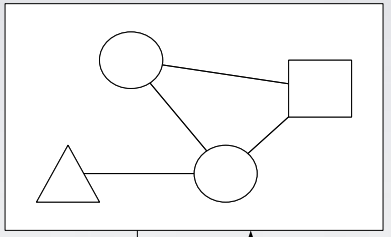
Un Sistema de Gestión de Base de Datos Orientado a Objetos (SGBDOO) es un SGBD que integra de forma transparente ***características de las bases de datos*** (almacenamiento y acceso a la información, entre otros) con ***características de los lenguajes de programación de aplicación orientados a objetos***.

Es decir, el SGBDOO se puede ver como **una extensión que le da características de persistencia** a algunos objetos de un lenguaje orientado a objetos

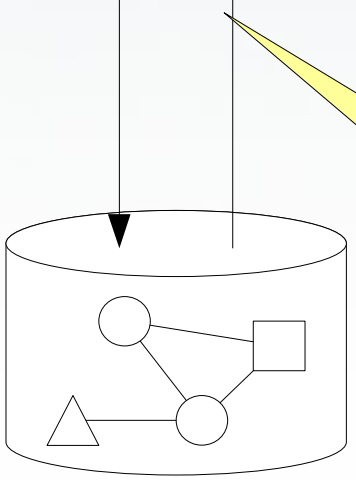
O como una **extensión que añade características orientación a objetos** a un Sistema de Gestión de Bases de Datos

¿cuál es la idea?



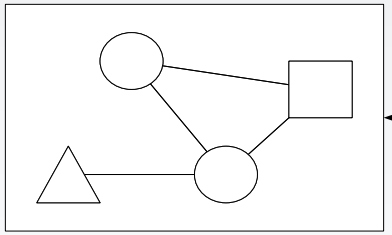


Estructuras de Datos a Nivel de Aplicación (Modelo) Lenguaje OO

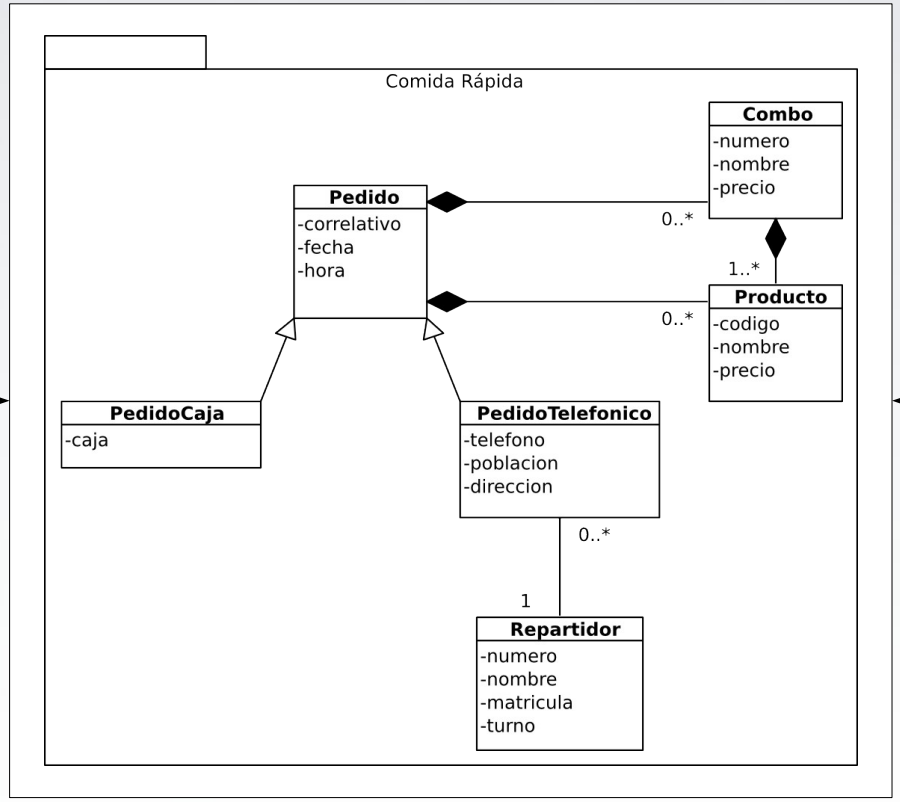


La persistencia está integrada por completo de forma transparente en la aplicación

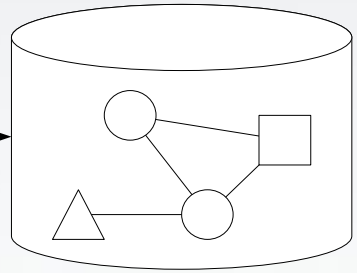
Modelo de Datos OO



Aplicación



Base de Datos OO

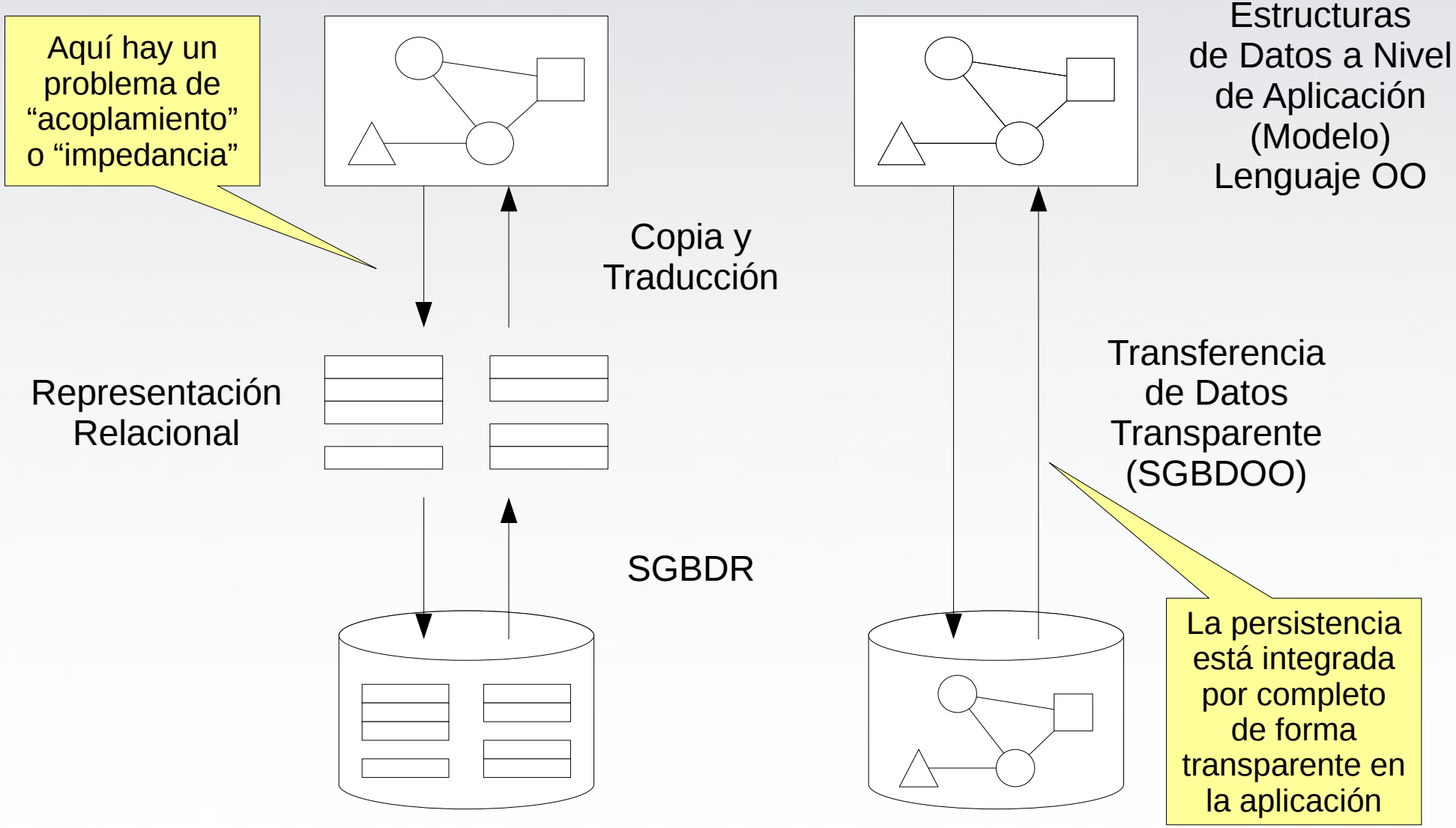


Nuestro trabajo es diseñar e implementar esto...

¿impedancia?

Reducir la **impedancia** entre las nuevas capas de aplicación (**Objetos**) y el almacenamiento persistente de los datos (**Relacional**)

Desarrollo con SGBDR y SGBDOO



Aplicación OO con un SGBDR (1) (Impedancia o Acoplamiento)

UNIVERSIDAD DE LOS ANDES VENEZUELA

```
Class.forName("com.mysql.jdbc.Driver");
Connection connection = DriverManager.getConnection( //
    "jdbc:mysql://localhost:3306/persona", "root", "");

Statement statement = connection.createStatement();

Persona p = new Persona();
p.setCedula("13556901");
p.setNombre("Pedro Perez");

String sql = "INSERT INTO t_persona VALUES (" + //
    getNextId(connection) + ", '" + //
    p.getCedula() + "', '" + p.getNombre() + "')";
System.err.println(sql);

statement.execute(sql);

connection.close();
```

Aquí hay que transformar los datos de un objeto a una sentencia DML en SQL. ***En este caso es simple, pero se puede volver repetitivo y propenso a errores***

Aplicación OO con un SGBDR (2) (Impedancia o Acoplamiento)

UNIVERSIDAD DE LOS ANDES VENEZUELA

```
Class.forName("com.mysql.jdbc.Driver");
Connection connection = DriverManager.getConnection( //
    "jdbc:mysql://localhost:3306/persona", "root", "");

Statement statement = connection.createStatement();

ResultSet rs = statement.executeQuery( //
    "SELECT * FROM t_persona WHERE cedula='13556901'");

Persona p = null;

if (rs.next()) {
  p = new Persona(rs.getString("cedula"), rs.getString("nombre"));
  p.setId(rs.getInt("id"));
} else {
  System.err.println("Persona no encontrada");
}

System.err.println(p.getId() + ";" + //
    p.getCedula() + ";" + p.getNombre());
connection.close();
```

Nuevamente, es necesario transformar los datos resultantes de la consulta a variables u objetos

Aplicación OO con un SGBDOO (3) (Orientado a Objetos / Mínimo Acoplamiento)

UNIVERSIDAD DE LOS ANDES VENEZUELA

```
Session session = CledaConnector.getInstance().getSession();

session.beginTransaction();

Persona p = new Persona();
p.setCedula("13556901");
p.setNombre("Pedro Perez");

session.saveOrUpdate(p);

session.getTransaction().commit();
session.close();
```

¿Que pasó aquí?
¡No puede ser tan fácil!

El la instancia p de tipo persona es persistido automáticamente

Aplicación OO con un SGBDOO (4) (Orientado a Objetos / Mínimo Acoplamiento)

UNIVERSIDAD DE LOS ANDES VENEZUELA

```
Session session = CledaConnector.getInstance().getSession();

session.beginTransaction();

Query q = session.createQuery( //
    "FROM Persona WHERE cedula=:cedula");
q.setString("cedula", "13556901");

Persona p = (Persona) q.uniqueResult();

System.err.println(p.getId() + ";" + //
    p.getCedula() + ";" + p.getNombre());

session.getTransaction().commit();
session.close();
```

La consulta aquí retorna un objeto directamente (Al contrario que SQL)

El lenguaje de consulta es “parecido” a SQL, usa principios similares, pero es orientado a objetos (en este caso es HQL)

El Estándar ODMG-93

(más por razones históricas
que prácticas)

El Estándar ODMG-93 (Release 1.1)

El release 1.1 del estándar del ODMG (Object Database Management Group) es un esfuerzo por estandarizar los conceptos fundamentales de los SGBDOO

El estandar fue desarrollado entre los años 1993 y 1994 por representantes de un amplio conjunto de empresas relacionadas al desarrollo de software y sistemas orientados a objetos

El estandar define:

Condiciones en las que nacen los SGBDR (con el tiempo...)

UNIVERSIDAD DE LOS ANDES VENEZUELA

Modelo de Objetos

Lenguaje de Definición de Objetos (ODL)

Lenguaje de Consulta de Objetos (OQL)

Enlaces con C++

Enlaces con Smalltalk

Enlaces con... (otros lenguajes OO)

El Estándar ODMG-93 (Release 1.1) (El Modelo de Objetos)

El modelo de objetos incluye y define conceptos de:

- Interfaces, Tipos, Implementaciones e Instancias (Objetos).
- Atributos / Métodos (Comportamiento).
- Representación de ***Vínculos entre Tipos*** (Colecciones / Referencias).
- Herencia (Jerarquías de Tipos y Clases / Especialización / Generalización / Supertipos / Subtipos).
- Extensiones de Tipos (Listas de objetos de cierto tipo).
- Claves de Tipos.
- Identidad de los Objetos (OID).

El Estándar ODMG-93 (Release 1.1) (El Modelo de Objetos)

UNIVERSIDAD DE LOS ANDES VENEZUELA

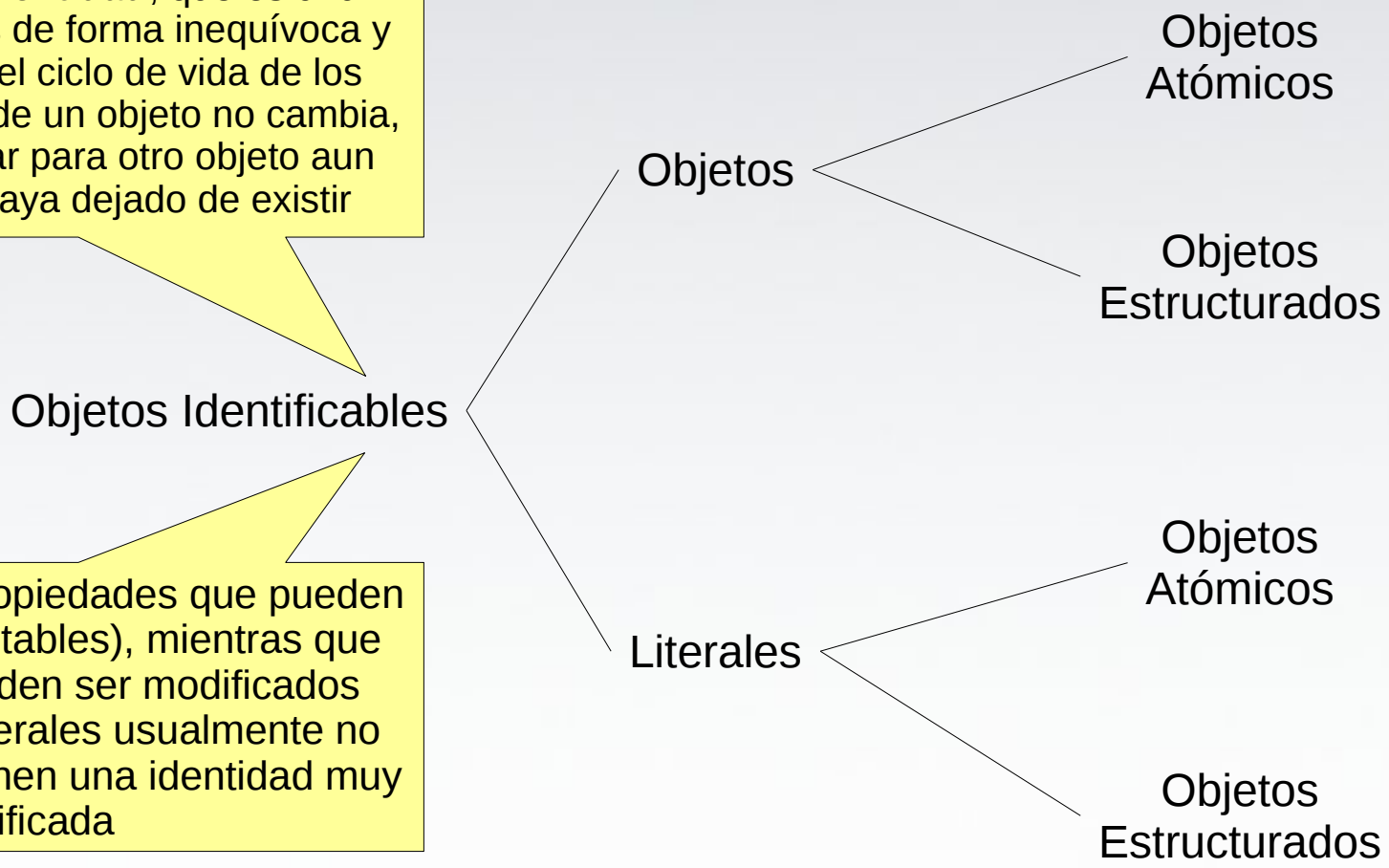
El modelo de objetos incluye y define conceptos de:

- Polimorfismo.
- Encapsulamiento.
- Excepciones.
- Estructura de complejidad arbitraria (vs información dispersa a lo largo de varias relaciones).
- Persistencia (Objetos que "*existen de forma permanentemente*").
- Soporta Transacciones.

El Estándar ODMG-93 (Release 1.1)
(El Modelo de Objetos)

UNIVERSIDAD DE LOS ANDES VENEZUELA

Los objetos tienen ***identidad***, que es una forma de identificarlos de forma inequívoca y a lo largo de TODO el ciclo de vida de los mismos. La identidad de un objeto no cambia, y NO se vuelve a usar para otro objeto aun cuando el original haya dejado de existir

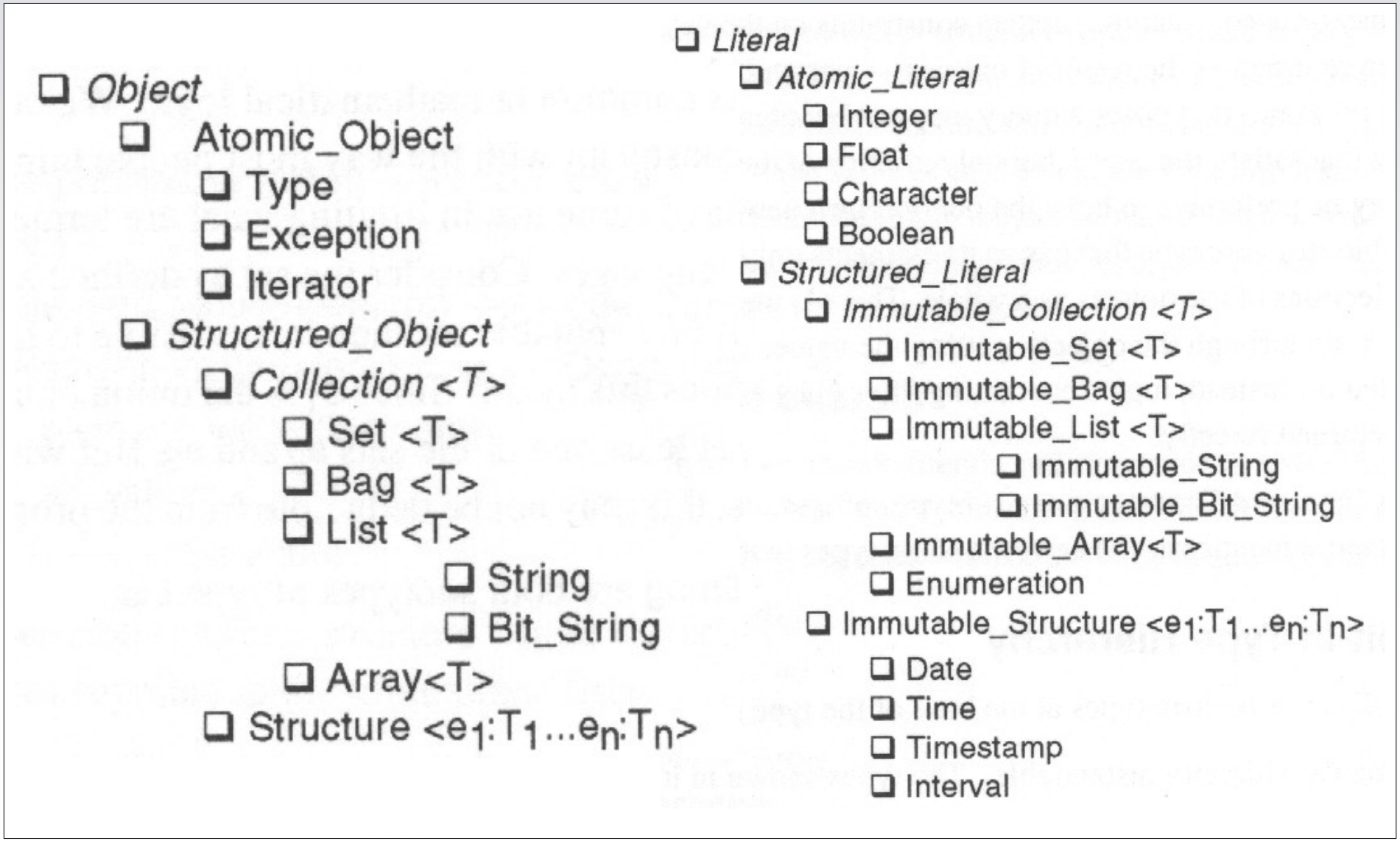


Los objetos tienen propiedades que pueden ser modificadas (Mutables), mientras que los literales no pueden ser modificados (Inmutables). Los literales usualmente no tienen identidad o tienen una identidad muy simplificada

Los tipos atómicos no están compuestos de otros objetos, mientras que los estructurados pueden estar compuestos de otros objetos

El Estándar ODMG-93 (Release 1.1)
(El Modelo de Objetos)

UNIVERSIDAD DE LOS ANDES VENEZUELA

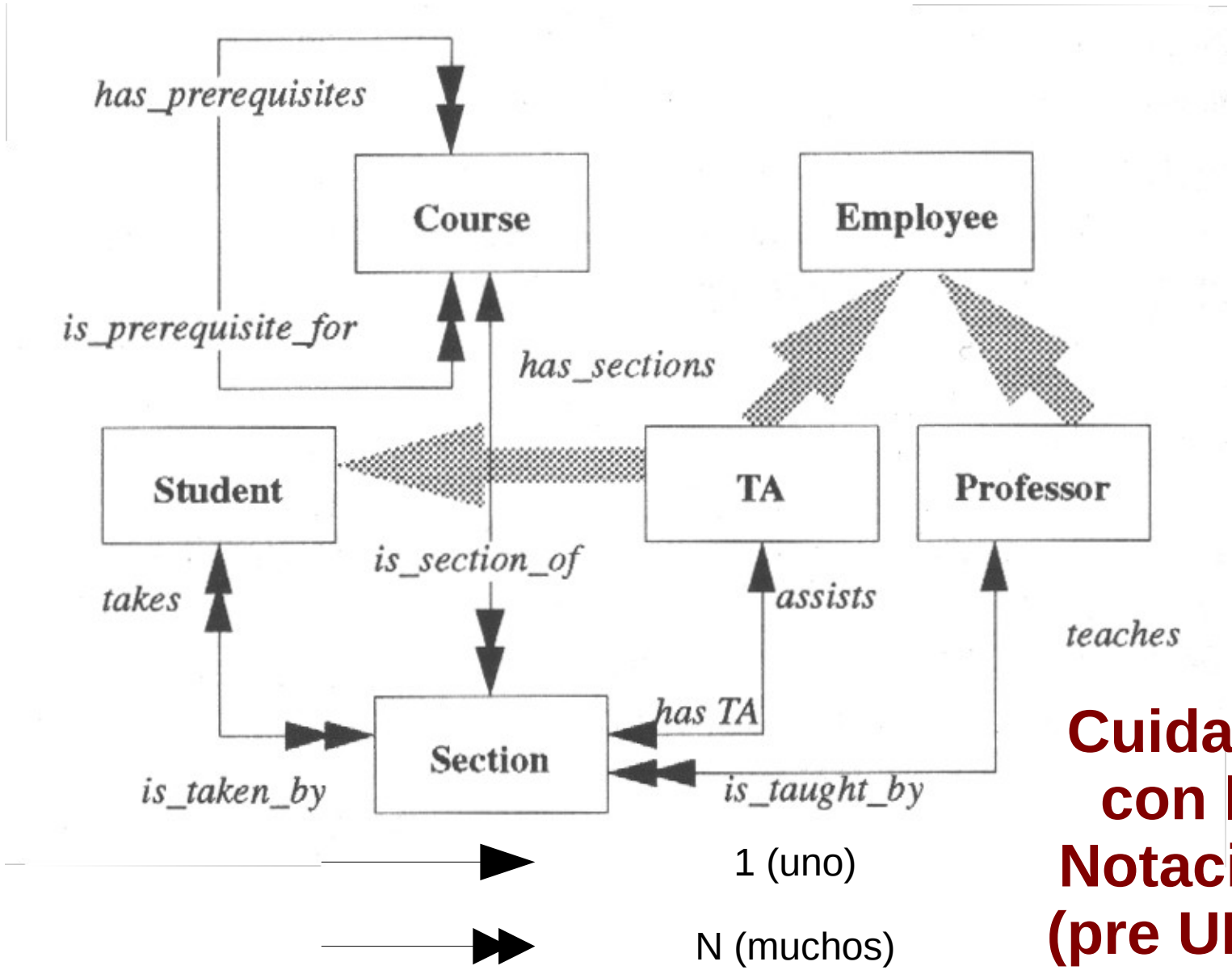


Tomado de “The Object Database Standar ODMG-93)

Object Definition Language ODL (1)

- ODL (Object Definition Language) es un lenguaje usado para definir las interfaces de tipos de objetos. Tiene las siguientes características:
  - Debe ser soportar todos la semántica del modelo de objetos de la ODMG
  - No es un lenguaje de programación completo, sólo un lenguaje de especificación de interfaces (y atributos)
  - Debe ser independiente de cualquier lenguaje de programación (independiente de C/C++, Java u otro)
  - Debe ser compatible con IDL (Interface Definition Language)
  - Debe ser simple y práctico, brindar valor a los desarrolladores de aplicaciones

Object Definition Language ODL (2)
(Un Ejemplo)



Cuidado con la Notación (pre UML)

Tomado de “The Object Database Standar ODMG-93)

Object Definition Language ODL (3)
(Un Ejemplo)

UNIVERSIDAD DE LOS ANDES VENEZUELA

```
interface Course
(       extent courses
        keys name, number)
{
        attribute String name;
        attribute String number;
        relationship List<Section> has_sections
            inverse Section::is_section_of
            {order_by Section::number};
        relationship Set<Course> has_prerequisites
             inverse Course::is_prerequisite_for;
        relationship Set<Course> is_prerequisite_for
             inverse Course::has_prerequisites;

        Boolean offer (in Unsigned Short semester) raises (already_offered);
        Boolean drop (in Unsigned Short semester) raises (not_offered);
};
```


Tomado de “The Object Database Standar ODMG-93)

Object Definition Language ODL (4)
(Un Ejemplo)

UNIVERSIDAD DE LOS ANDES VENEZUELA

```
interface Student
(       extent students
        keys name, student_id)
{
        attribute String name;
        attribute String student_id;
        attribute Struct Address {String college, String room_number}
            dorm_address;
        relationship Set<Section> takes inverse Section::is_taken_by;
        Boolean register_for_course (in Unsigned Short course,
            in Unsigned Short Section)
            raises (unsatisfied_prerequisites, section_full, course_full);
        void drop_course (in Unsigned Short Course)
            raises (not_registered_for_that_course);
        void assign_major (in Unsigned Short Department);
        Short transfer (in Unsigned Short old_section,
            in Unsigned Short new_section)
            raises (section_full, not_registered_in_section);
};
```


Tomado de “The Object Database Standar ODMG-93)

Object Definition Language ODL (5)
(Un Ejemplo)

UNIVERSIDAD DE LOS ANDES VENEZUELA

```
interface Section
(       extent sections
        key (is_section_of, number))
{
        attribute String number;
        relationship Professor is_taught_by inverse Professor::teaches;
        relationship TA has_TA inverse TA::assists;
        relationship Course is_section_of inverse Course::has_sections;
        relationship Set<Student> is_taken_by inverse Student::takes;
};
```


Tomado de “The Object Database Standar ODMG-93)

Object Definition Language ODL (6)
(Un Ejemplo)

UNIVERSIDAD DE LOS ANDES VENEZUELA

```
interface Professor: Employee
(       extent professors)
{
        attribute Enum Rank {full, associate, assistant} rank;
        relationship Set<Section> teaches inverse Section::is_taught_by;
        Short grant_tenure () raises (ineligible_for_tenure);
};
```

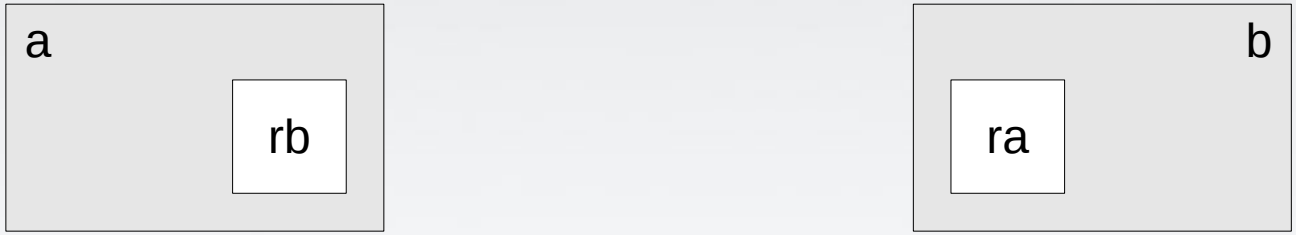
```
interface Professor: Employee
(       extent professors)
{
        attribute Enum Rank {full, associate, assistant} rank;
        relationship Set<Section> teaches inverse Section::is_taught_by;
        Short grant_tenure () raises (ineligible_for_tenure);
};
```


Tomado de “The Object Database Standar ODMG-93)

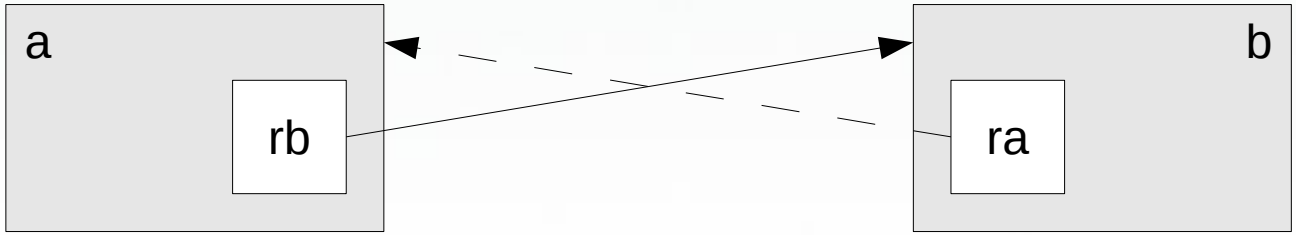
Object Definition Language ODL (7)
(Vínculos)

UNIVERSIDAD DE LOS ANDES VENEZUELA

1-1 Sin relación:



1-1 Con relación

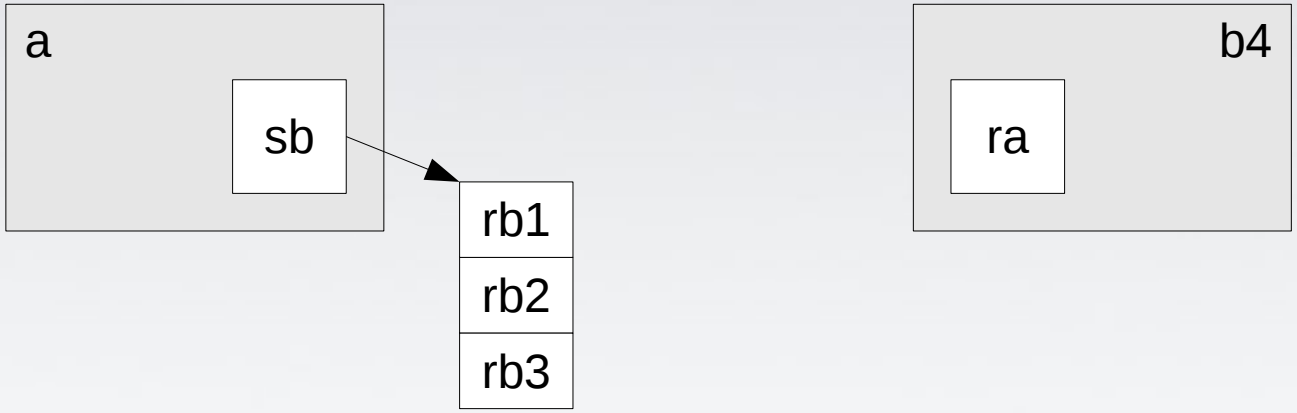


Adaptado de: "The Object Database Standar (ODMG-93)

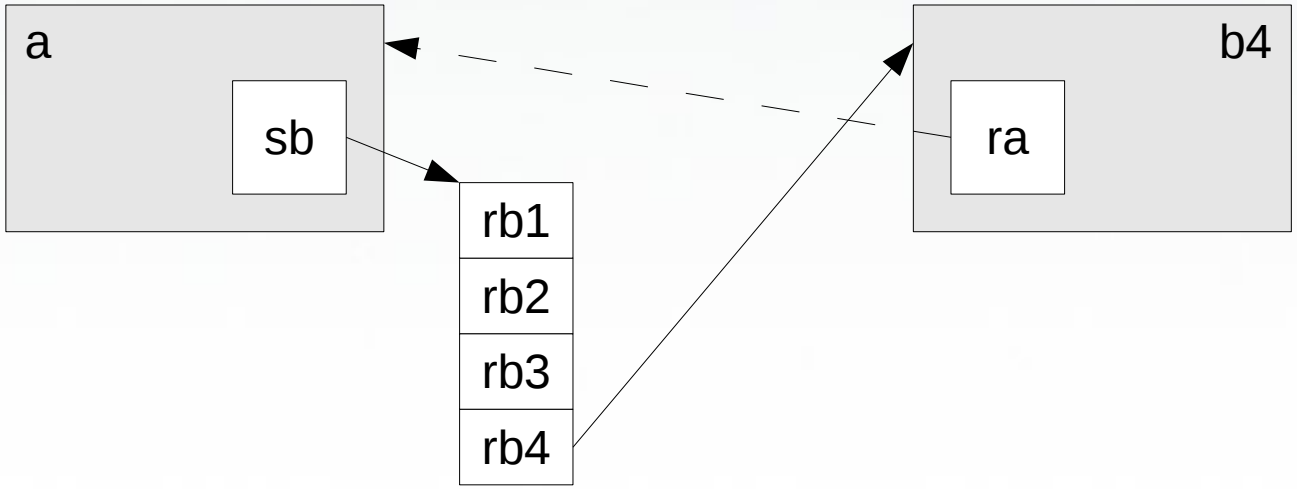
Object Definition Language ODL (8)
(Vínculos)

UNIVERSIDAD DE LOS ANDES VENEZUELA

1-N Sin relación:



1-N Con relación

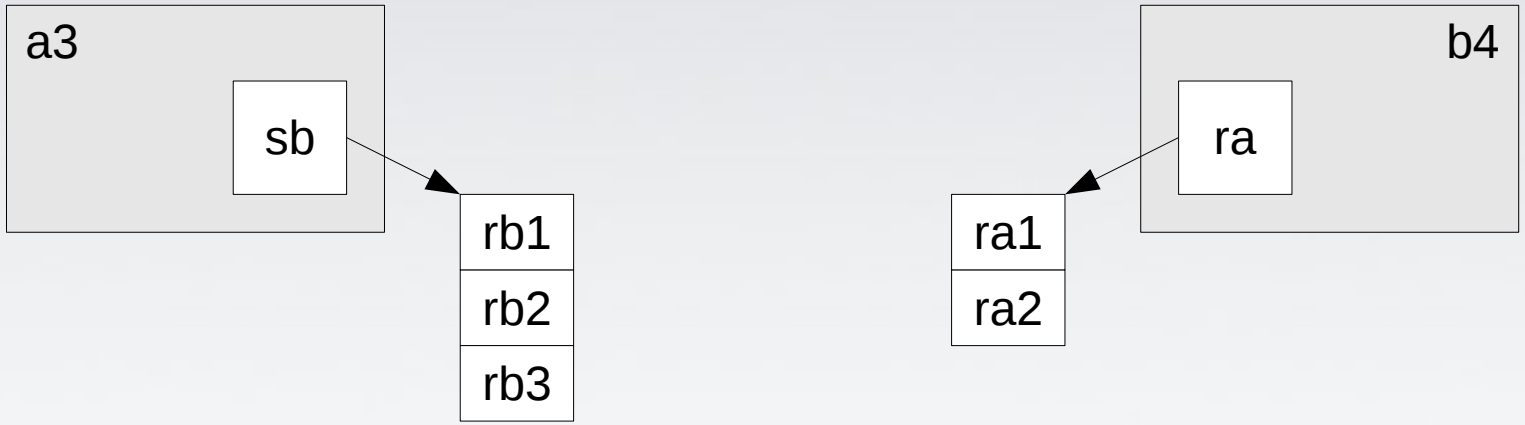


Adaptado de: "The Object Database Standar (ODMG-93)

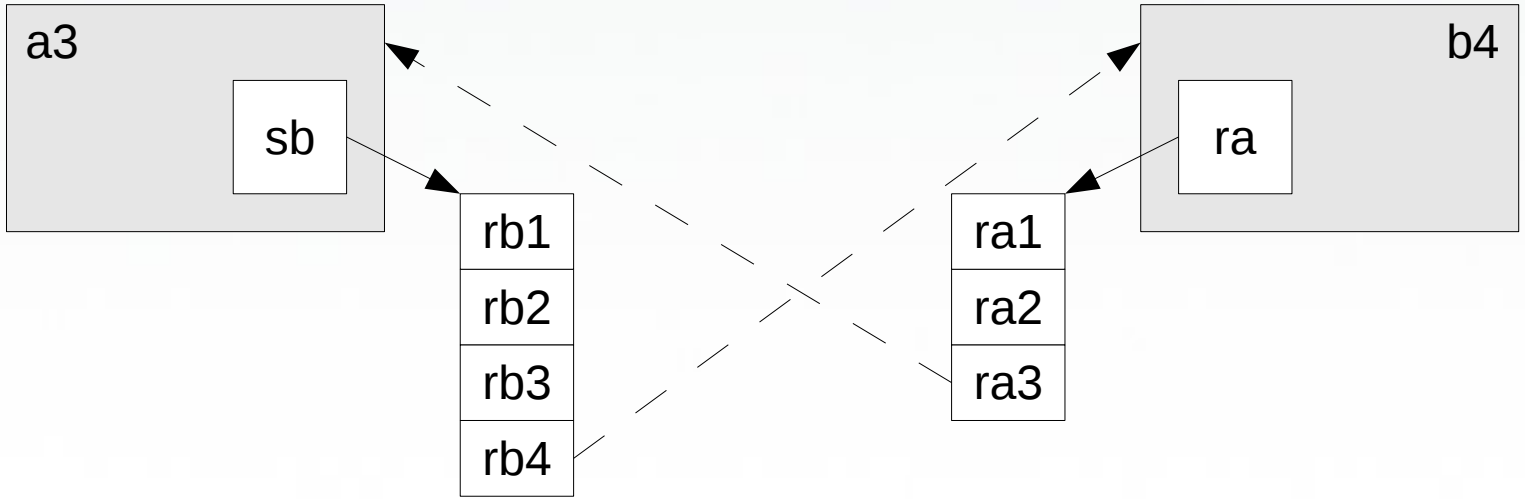
Object Definition Language ODL (9)
(Vínculos)

UNIVERSIDAD DE LOS ANDES VENEZUELA

N-M Sin relación:



N-M Con relación



Adaptado de: "The Object Database Standar (ODMG-93)

OQL (1) (Características)

UNIVERSIDAD DE LOS ANDES VENEZUELA

- OQL (Object Query Language) es un lenguaje de consulta orientado a objetos simple y completo con las siguientes características:
  - Es orientado a objetos
  - Declarativo / Abstracto
    (no es completo desde el punto de vista computacional)
  - Su sintaxis es ***similar*** a SQL (que es uno de los lenguajes de consulta más usados en la industria)
  - Acceso declarativo a objetos (propiedades y métodos)
  - Semántica formal bien definida
  - Basado en el modelo de objetos de la ODMG
  - No incluye operaciones de actualización (sólo de consulta)

OQL (2)
(Algunos Ejemplos)

UNIVERSIDAD DE LOS ANDES VENEZUELA

```
select distinct x.edad from x in Personas where x.nombre="Pedro"

select distinct struct(e : x.edad, s : x.sexo)
  from x in Personas where x.nombre="Pedro"

select distinct struct(nombre : x.nombre,
conjunto_subordinados : (select y
         from y in x.subordinados
         where y.salario >100000)
  from x in Empleados

select struct(e : x.edad, s : x.sexo)
  from x in (select y
               from y in Empleados
               where y.antiguedad = 10)
  where x.nombre="Pedro"

Empleados

Empleados.subordinados
```

La sintaxis es similar a SQL, pero en general, completamente orientada a objetos

Otros Lenguajes de Consulta OO (Criteria Query)

UNIVERSIDAD DE LOS ANDES VENEZUELA

```
ODB odb = ODBFactory.open(ODB_NAME);

IQuery query = new CriteriaQuery( //
    Player.class, Where.equal("name", "pedro"));

Objects<Player> players = odb.getObjects(query);

int i = 1;

while (players.hasNext()) {
  System.out.println((i++) + "\t: " + players.next());
}

odb.close();
```

En un query de tipo criteria, la condición se construye encadenando una serie de “átomos” generados por un conjunto de métodos predefinidos

Tomado del tutorial de NeoDatis (http://www.neodatis.org/)

Otros Lenguajes de Consulta OO (Criteria Query)

UNIVERSIDAD DE LOS ANDES VENEZUELA

```
ODB odb = ODBFactory.open(ODB_NAME);

IQuery query = new CriteriaQuery( //
    Sport.class, Where.equal("name", "volley-ball"));

Sport volleyBall = (Sport) odb.getObjects(query).getFirst();

query = new CriteriaQuery( //
    Player.class, Where.equal("favoriteSport", volleyBall));

Objects<Player> players = odb.getObjects(query);

int i = 1;

while (players.hasNext()) {
  System.out.println((i++) + "\t: " + players.next());
}

odb.close();
```


Tomado del tutorial de NeoDatis (http://www.neodatis.org/)

Otros Lenguajes de Consulta OO (Criteria Query)

UNIVERSIDAD DE LOS ANDES VENEZUELA

```
ODB odb = ODBFactory.open(ODB_NAME);

IQuery query = new CriteriaQuery( //
    Player.class, Where.or().add( //
        Where.equal("favoriteSport.name", "volley-ball")).add( //
            Where.like ("favoriteSport.name", "%nnis"));

Objects<Player> players = odb.getObjects(query);

int i = 1;

while (players.hasNext()) {
  System.out.println((i++) + "\t: " + players.next());
}

odb.close();
```


Tomado del tutorial de NeoDatis (http://www.neodatis.org/)

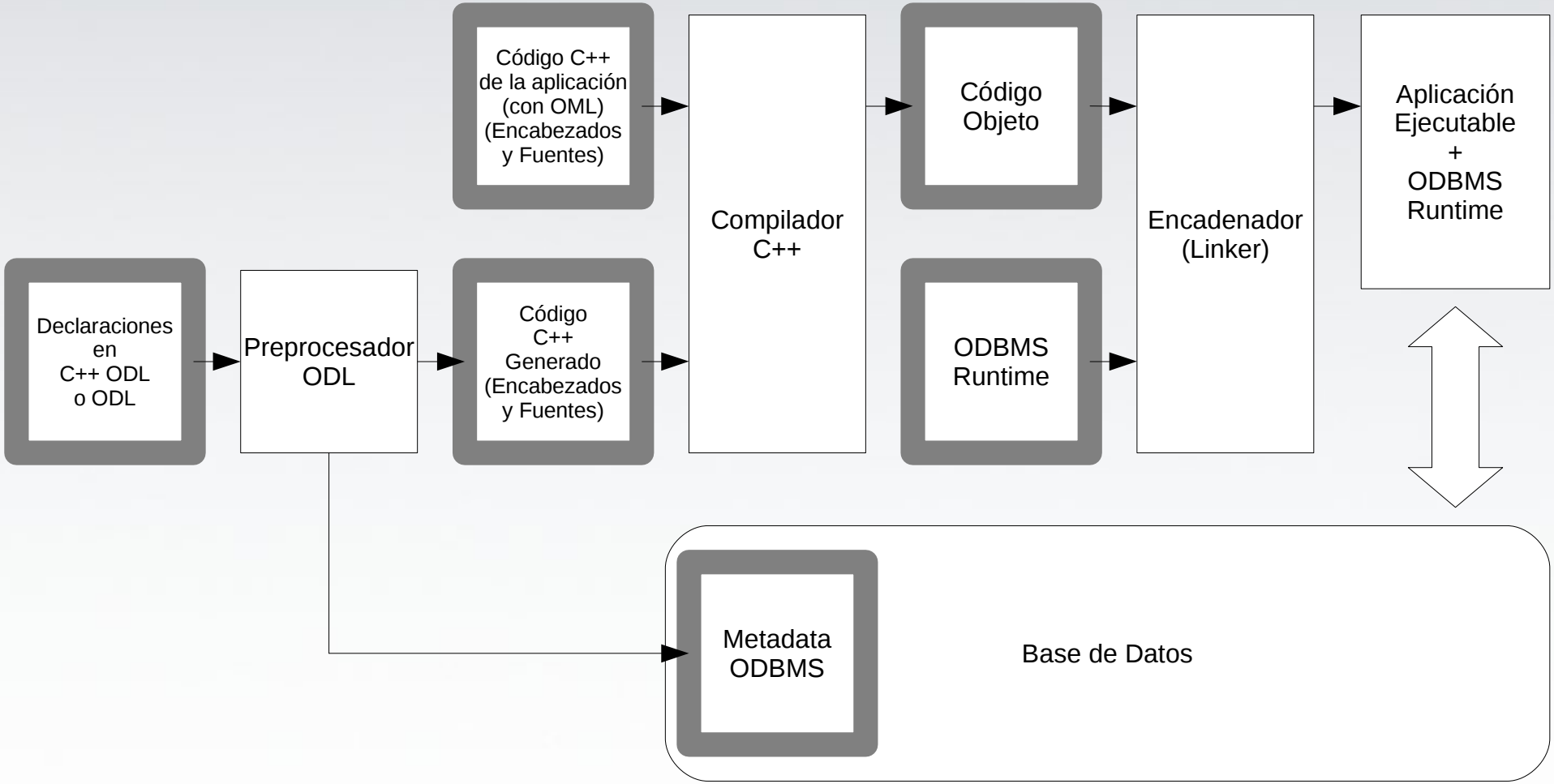
Otros Lenguajes de Consulta OO (Consultas Nativas)

UNIVERSIDAD DE LOS ANDES VENEZUELA

```
ODB odb = ODBFactory.open(ODB_NAME);

IQuery query = new SimpleNativeQuery() {
  public boolean match(Player player) {
    // Se incluye el objeto si retorna true
    return player.getFavoriteSport().getName(). //
        toLowerCase().startsWith("volley");
  }
};

Objects<Player> players = odb.getObjects(query);

int i = 1;

while (players.hasNext()) {
  System.out.println((i++) + "\t: " + players.next());
}

odb.close();
```

Tomado del tutorial de NeoDatis (http://www.neodatis.org/)

ODL / OQL / OML
(Integración con un Lenguaje de Programación)



La idea es que el acceso al SGBDOO sea una “extensión” del lenguaje nativo en el que se va a utilizar

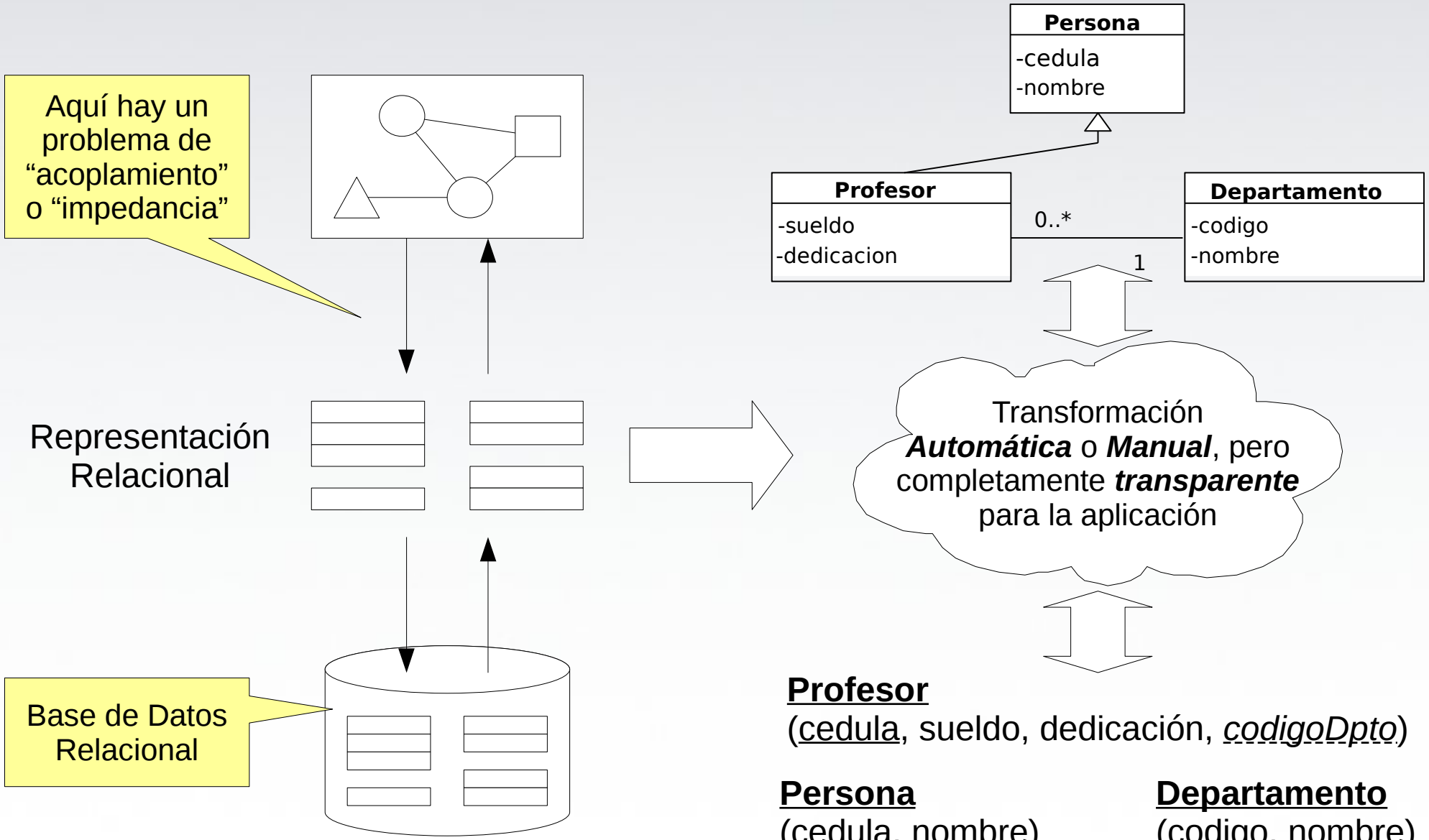
Algunos SGBDOO (Comerciales y Free Software)

Caché, ConceptBase, **Db4o**, eXtremeDB, **eyeDB**, **Facets (previously known as GemStone-J)**, FastDB Main Memory DBMS, Gemstone Database Management System, Generic Object Oriented Database System (GOODS), GigaBASE Database Management System, Haley Systems, JADE programming language, Jasmine Object Database, JDOInstruments, JODB (Java Objects Database), Magma Object Database, Matisse, MyOODB, **NeoDatis ODB**, ObjectDB, **Objectivity/DB**, **Objectstore**, Virtuoso Universal Server, Orient ODBMS, **Ozone Database Project**, Perst, Statice, stSoftware ODBMS, **Versant Object Database**, VOSS (Virtual Object Storage System), **Zope Object Database**, JOAFIP object persistence in file

Tomado de Wikipedia http://en.wikipedia.org/wiki/Comparison_of_object_database_management_systems

¿que alternativas hay a los SGBDOO?

Alternativas a los SGBDOO (1)

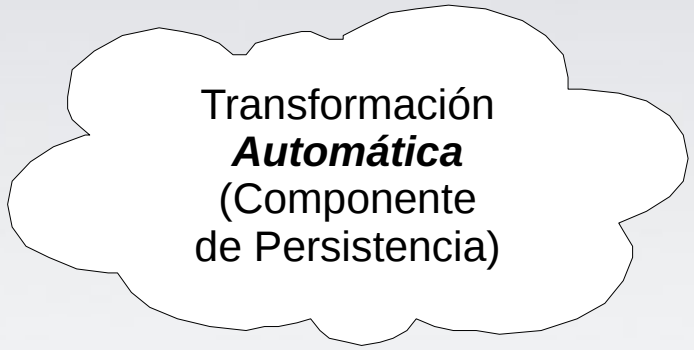


Profesor
(cedula, sueldo, dedicación, *codigoDpto*)

Persona
(cedula, nombre)

Departamento
(codigo, nombre)

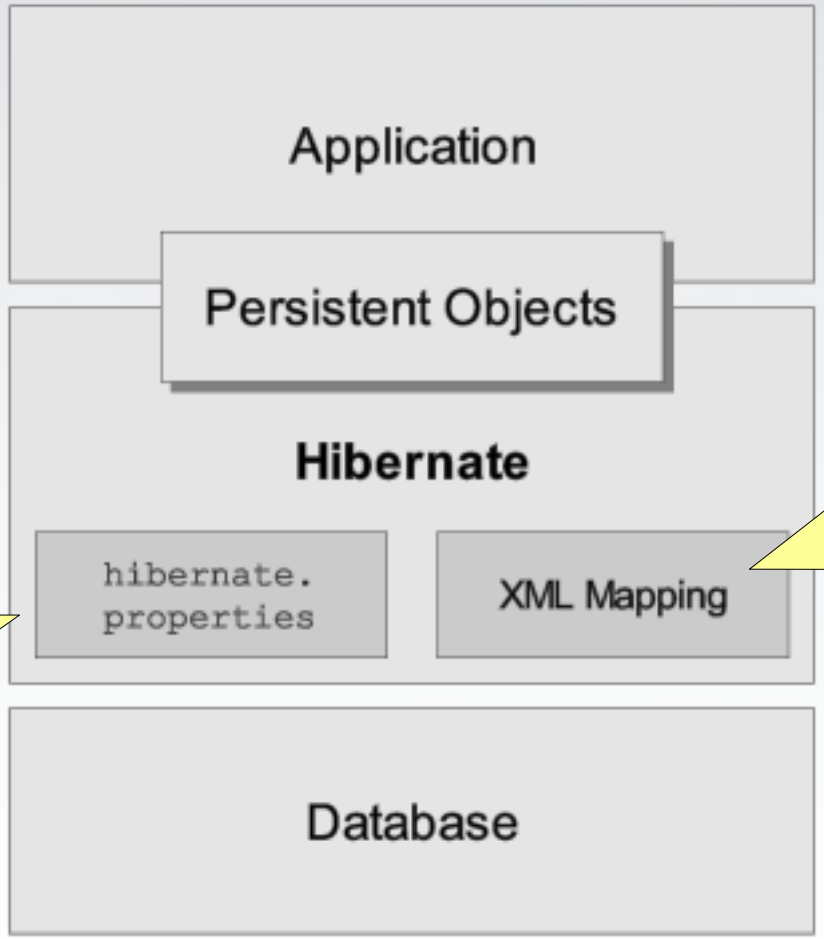
Alternativas a los SGBDOO (3) (Object-Relational Mapping / ORM)



- La correspondencia (mapa) entre las relaciones y los objetos se describe utilizando:
  - XML
  - XDoclet
  - Anotaciones (Dependiente del Lenguaje)
  - Otros

- Hay un componente que en base a la descripción de correspondencia genera de forma automática todos los SQL necesarios para consultar y actualizar la base de datos
- El cliente utiliza / consulta / actualiza los objetos persistentes y el ORM se encarga de sincronizar el estado con la Base de Datos
- Desde el punto de vista del cliente es como si usara un SGBDOO

Alternativas a los SGBDOO (4) (ORM Utilizando Hibernate)



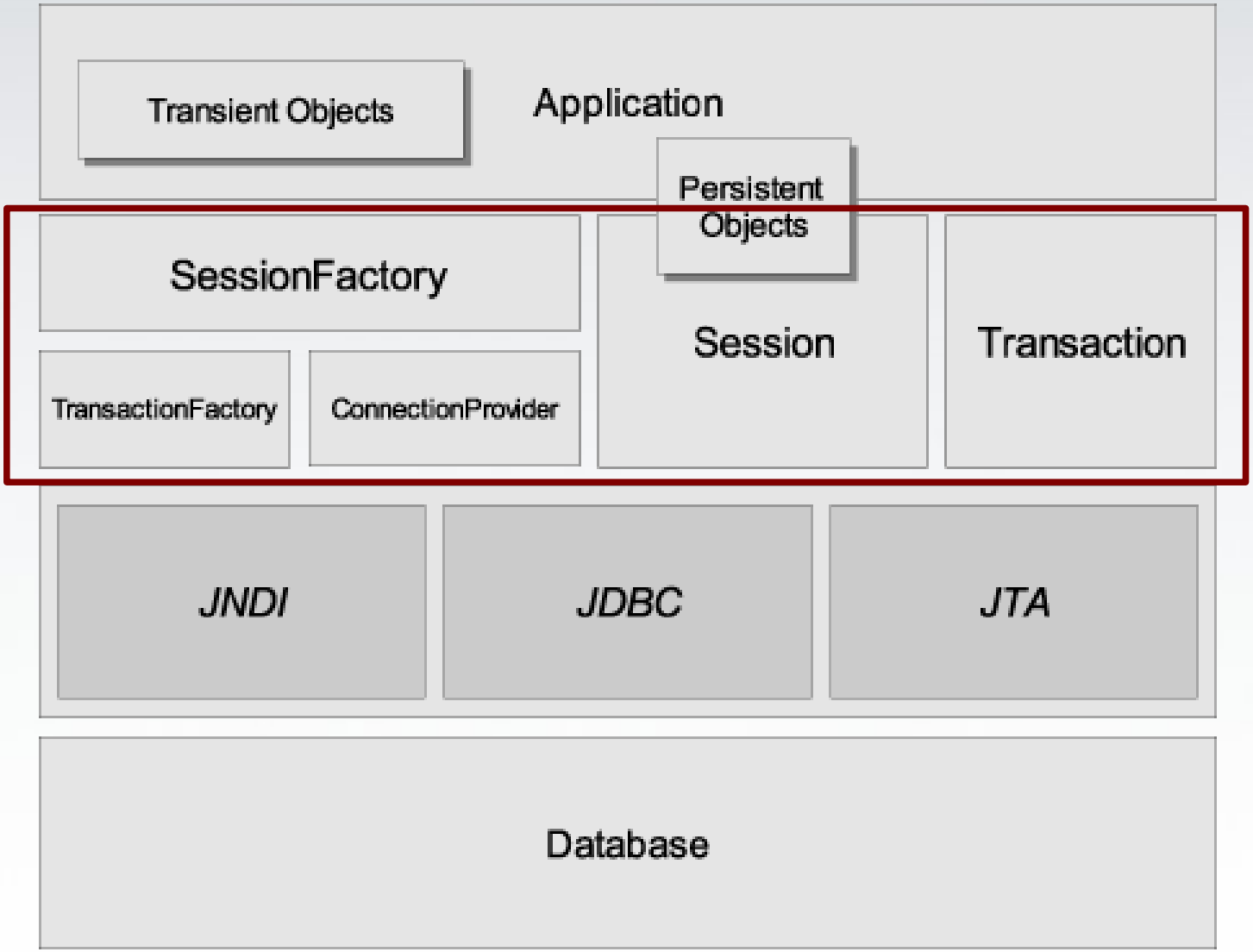
Hibernate permite configurar en un archivo ***.properties*** la base de datos a la que se desea conectar. Nuevas versiones hacen esto en XML

Actualmente, además de XML Hibernate permite usar Java Annotations para definir la correspondencia Objeto-Relacional

Vista general de Hibernate

Tomado del manual de Hibernate (http://www.hibernate.org/)

Alternativas a los SGBDOO (5) (ORM Utilizando Hibernate)



Arquitectura “full cream” de Hibernate

Tomado del manual de Hibernate (http://www.hibernate.org/)

Alternativas a los SGBDOO (6) (ORM Utilizando Hibernate)

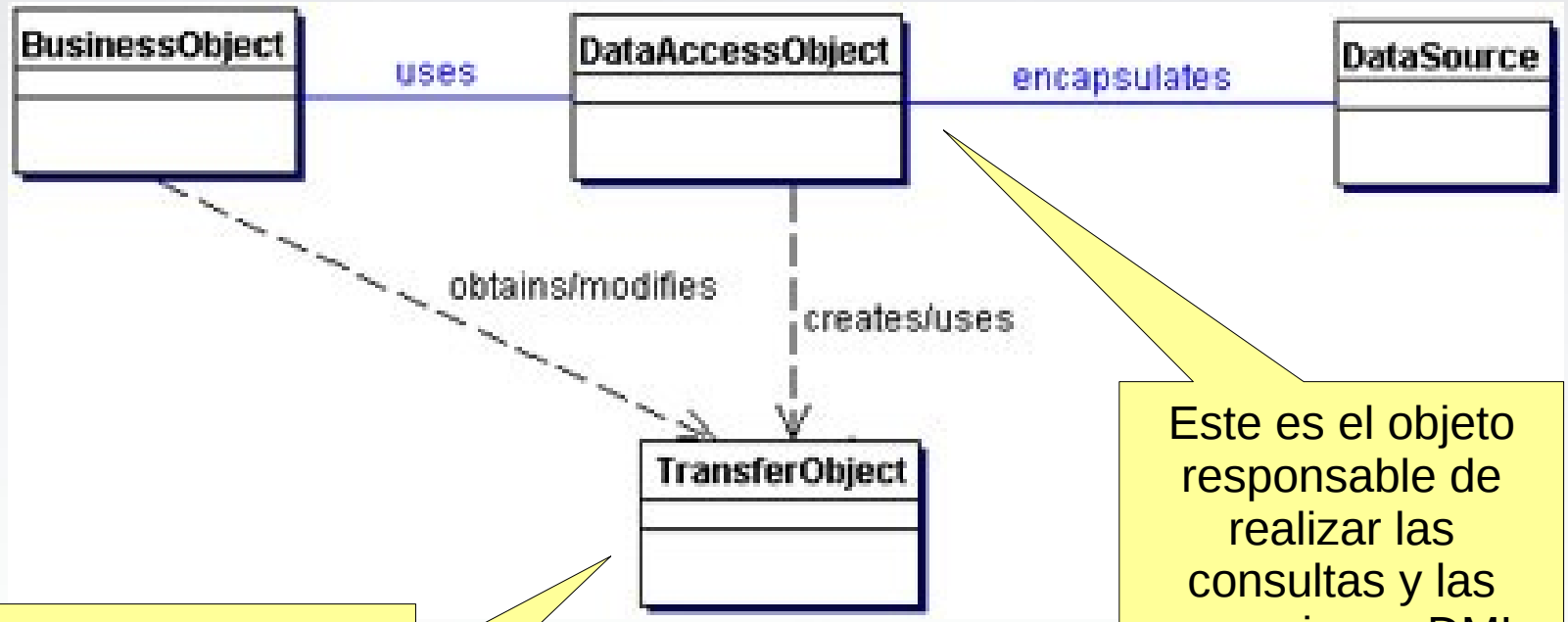
UNIVERSIDAD DE LOS ANDES VENEZUELA

Ver Ejemplos de Código

Alternativas a los SGBDOO (7) (Data Access Objects / DAO)

- Se encapsulan las operaciones de acceso a la base de datos implementando un objeto especial (DAO)
- La clase DAO implementa métodos para hacer CRUD (*Create Read Update Delete*) básico.
- Adicionalmente, la interfaz DAO implementa métodos para consultas (Ej: findByXXX(...) / listByXXX(...))
- Se implementa un TO (Transfer Object) que contiene la información a almacenar y sirve para interactuar con el DAO
- Se implementa una clase Factory que retorna el DAO adecuado según el tipo de Base de Datos a la que se desea acceder

Alternativas a los SGBDOO (8) (Data Access Objects / DAO)

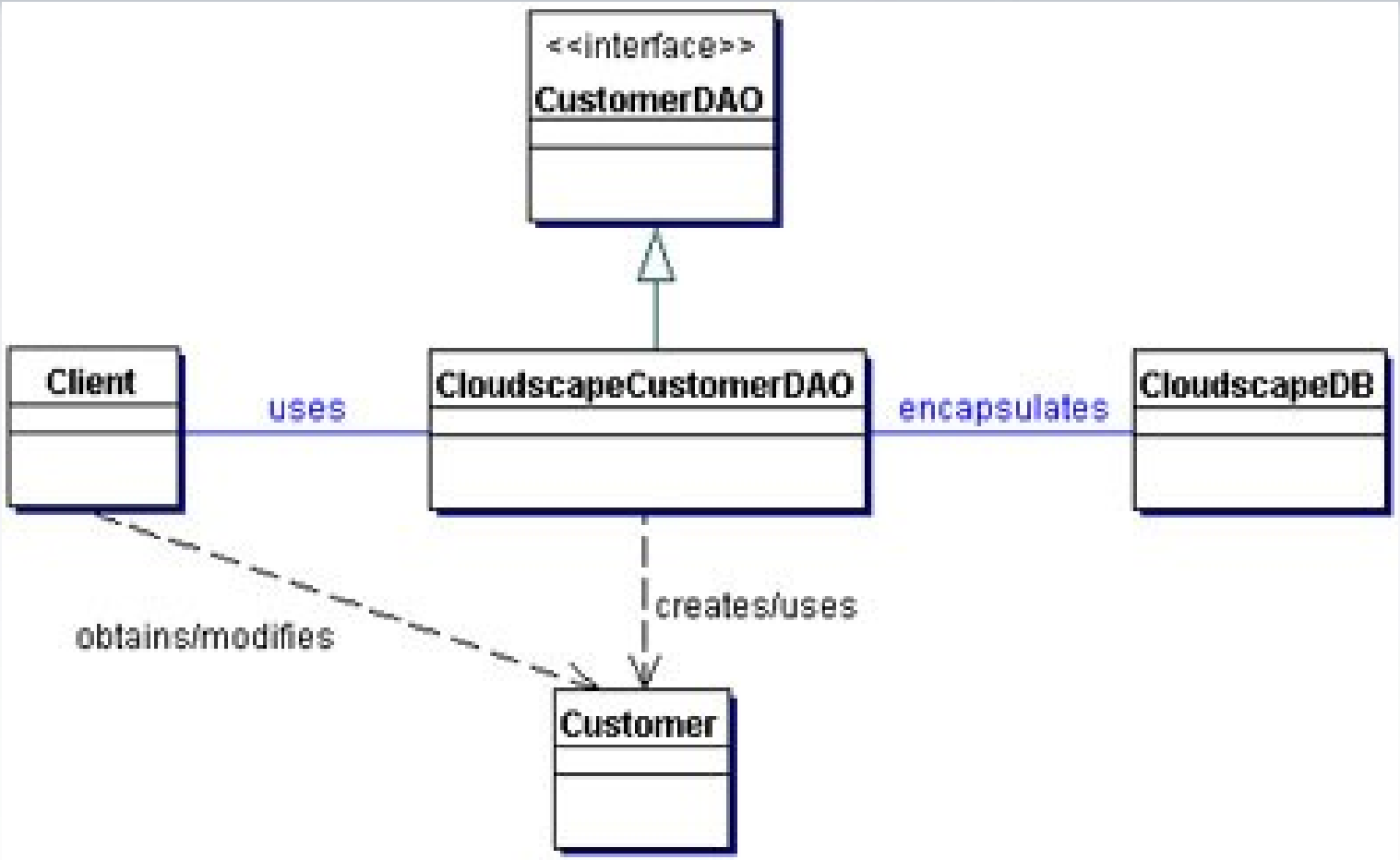


Este es el objeto que contiene los datos que vienen de (o van a) la Base de Datos

Este es el objeto responsable de realizar las consultas y las operaciones DML necesarias para mantener el estado de la BD sincronizado con el del objeto

Tomado de (http://java.sun.com/blueprints/corej2eepatterns/Patterns/DataAccessObject.html)

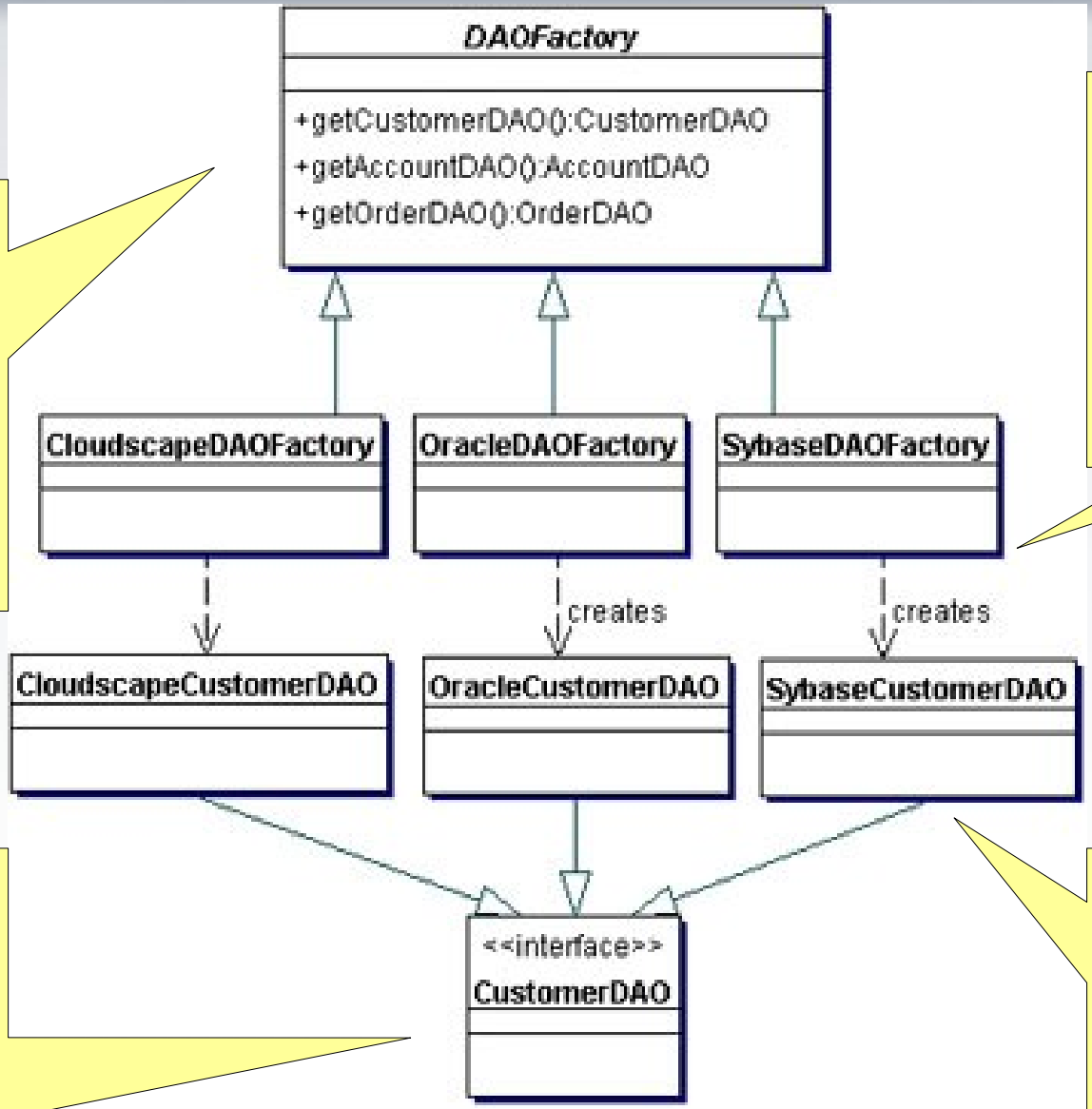
Alternativas a los SGBDOO (9) (Data Access Objects / DAO)



Este es un caso concreto del patrón de la lámina anterior
(Para un objeto “Cliente” (Customer) y para “Cloudscape” como Base de Datos (Data Source / Fuente de Datos)

Tomado de (http://java.sun.com/blueprints/corej2eepatterns/Patterns/DataAccessObject.html)

Alternativas a los SGBDOO (10) (Data Access Objects / DAO)



Esta clase sirve de fábrica para crear los DAOs requeridos según el Modelo a persistir y la fuente de datos específica que se necesite

Estas son implementaciones concretas de la Fábrica dependiendo de la fuente de datos seleccionada

Esta es la interfaz que define las operaciones que se pueden hacer sobre el DAO cliente (Customer)

Estas son implementaciones concretas del DAO dependiendo de la fábrica usada seleccionada

Tomado de (http://java.sun.com/blueprints/corej2eepatterns/Patterns/DataAccessObject.html)

Alternativas a los SGBDOO (11)
(Data Access Objects / DAO)

UNIVERSIDAD DE LOS ANDES VENEZUELA

Ver Ejemplos de Código

Alternativas a los SGBDOO (12)
(Algunos Productos ORM / DAO Software Libre)

UNIVERSIDAD DE LOS ANDES VENEZUELA

Lenguaje	Nombre
C++	LiteSQL, Debea, dtemplatelib (Database Template Library)
Java	Carbonado, **Castor**, Cayenne, Ebean, EclipseLink, Enterprise Objects Framework, ***Hibernate***, **iBATIS**, Java Data Objects (JDO), JPOX (JDO2), Kodo, OpenJPA, TopLink (Oracle), **Torque**
.NET	ADO.NET Entity Framework (Microsoft), Base One Foundation Component Library, BCSEi ORM Code Generator, Business Logic Toolkit for .NET, Castle ActiveRecord, DataObjects.Net v4.0, DevForce, Developer Express, eXpress Persistent Objects (XPO), EntitySpaces, Euss, Habanero, iBATIS, Invist, LLBLGen, LightSpeed, Neo, NConstruct, **NHibernate**, Opf3, ObjectMapper .NET, OpenAccess, TierDeveloper, Persistor.NET, Quick Objects, Sooda, Subsonic
PHP	**Doctrine**, **Propel**, **EZPDO**, DABL, Data Shuffler Data mapper implementation (New BSD), Outlet Open source ORM, Coughphp Open source ORM
Python	**Django**, SQLAlchemy, SQLObject, Storm
Ruby	ActiveRecord, Datamapper, **iBATIS**

Tomado de Wikipedia http://en.wikipedia.org/wiki/List_of_object-relational_mapping_software

UNIVERSIDAD DE LOS ANDES VENEZUELA

¡Gracias!