Otros modelos de datos

Tipos de datos complejos

- Modelos semánticos
  - Entity Relatioship
  - Generic Semantic Model
- Datos complejos
- Bases de Datos OO
  - Lenguajes persistentes
  - Sistemas objeto-relacionales

Tipos de datos complejos

- Modelo relacional -> Tipos atómicos

- Aplicaciones con tipos de datos complejos.
  - Atributos multivaluados
  - Aplicaciones CAD
  - Multimedia
  - Documentos e Hipertexto

Bases de datos orientadas a objetos

- Los lenguajes de las bases de datos trabajan directamente con datos que son persistentes.
- Los datos de los lenguajes de programación generalmente no son persistentes
- La manera tradicional de realizar las interfaces de las bases de datos con los lenguajes de programación es incorporar SQL dentro del lenguaje de programación

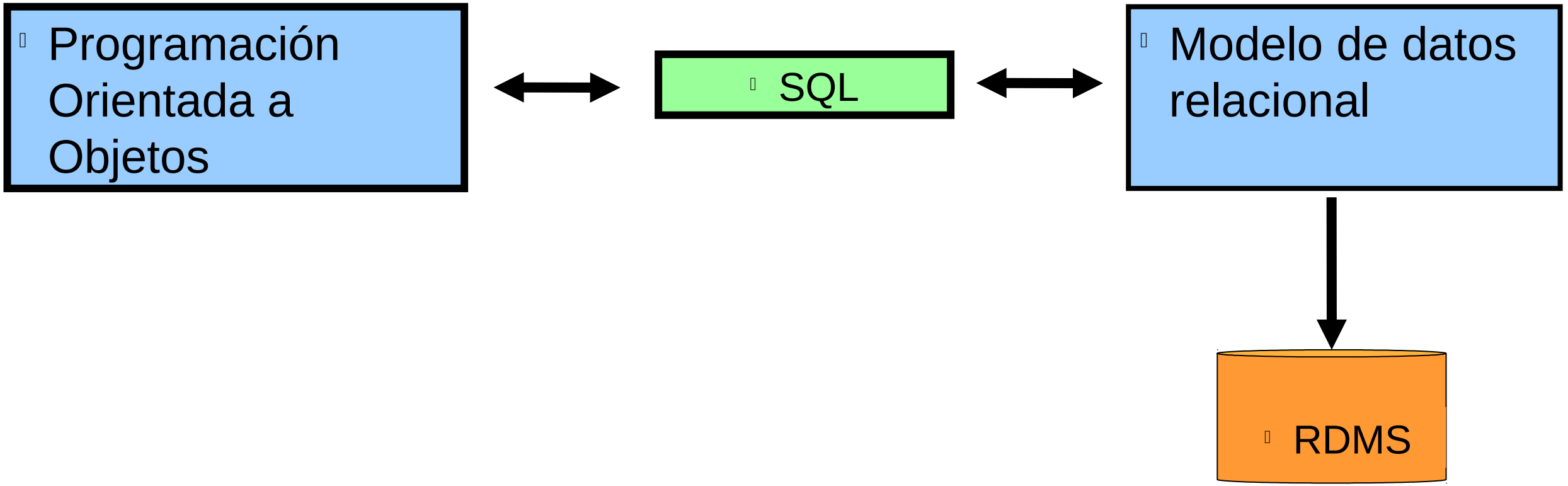
SQL embebido

- Los programadores son responsables de las conversiones de tipos entre el **lenguaje anfitrión y SQL**.
- El código para la conversión entre objetos y tuplas opera **fuera** del sistema de tipos orientado a objetos
- La conversión en la base de datos entre el formato orientado a objetos y el formato relacional de las tuplas **necesita gran cantidad de código**

Algunos Problemas con RDMS

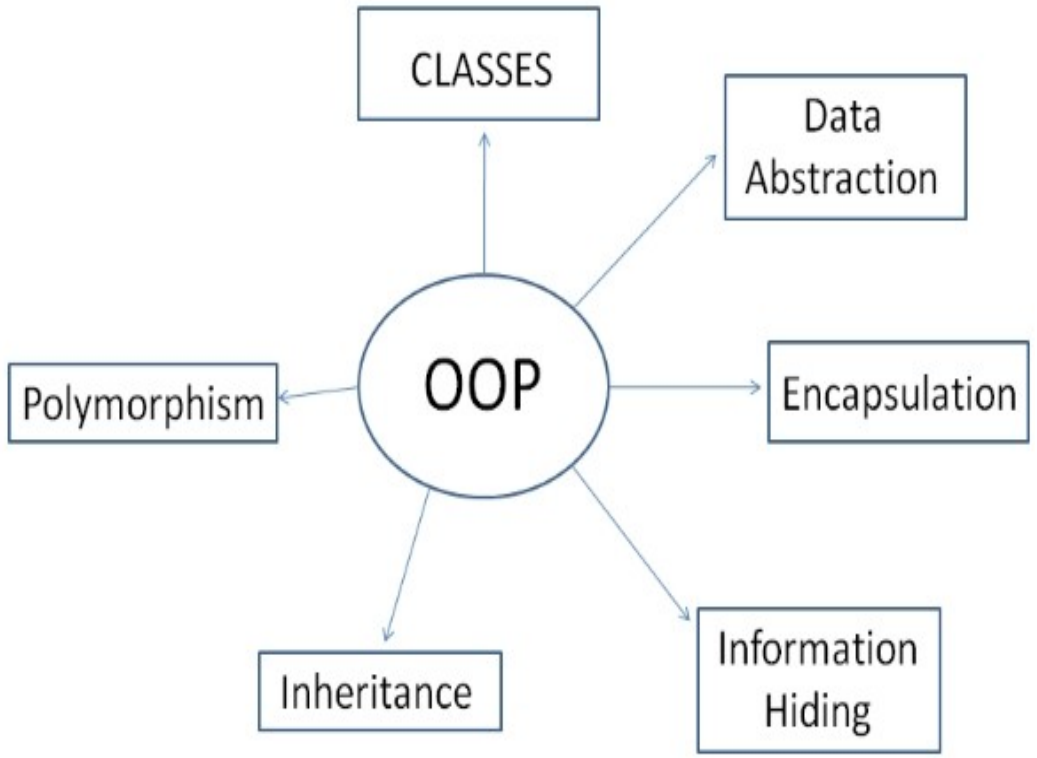
- Impedance Mismatch
- Semantic Gap
- Nuevas aplicaciones: CAD, CAM

• Impedance Mismatch



BASES DE DATOS ORIENTADAS A OBJETOS

- Es una base de datos que soporta el modelado y almacenamiento de datos como Objetos.



Lenguajes de programación persistentes

- Los **lenguajes de programación persistentes** son lenguajes de programación extendidos con constructores para el tratamiento de datos persistentes
- El lenguaje de consulta se halla totalmente integrado con el lenguaje anfitrión y ambos comparten el mismo sistema de tipos
- C++ y JAVA

Persistencia de los objetos

Persistencia por clases.

- Declarar que una clase es persistente.

- Todos los objetos de la clase son persistentes de manera predeterminada
- Todos los objetos de las clases no persistentes son transitorios

- **No es flexible**: resultar útil disponer en una misma clase tanto de objetos transitorios como de objetos persistentes

Persistencia de los objetos

Persistencia por creación:

Los objetos son persistentes o transitorios en función de la manera de crearlos. Este enfoque se sigue en varios sistemas de bases de datos orientados a objetos.

Persistencia de los objetos

Persistencia por marcas

Marcar los objetos como persistentes después de haberlos creado.

Todos los objetos se crean como **transitorios**, pero, si un objeto tiene que persistir más allá de la ejecución del programa, hay que marcarlo de manera explícita.

La decisión sobre la **persistencia o la transitoriedad** se retrasa hasta después de la creación del objeto

Persistencia de los objetos

Persistencia por alcance

Uno o varios objetos se declaran objetos persistentes (objetos raíz) de manera explícita.

Todos los demás objetos serán persistentes si (y sólo si) son alcanzables desde el objeto raíz.

La decisión sobre la persistencia o la transitoriedad se retrasa hasta **después de la creación** del objeto

Identidad de los objetos (*OID*)

Identificadores de objetos transitorios. Los identificadores de objetos transitorios sólo son válidos mientras se ejecuta el programa que los creó.

Cuando se crean objetos persistentes se les asignan identificadores de objetos persistentes

Hay una relación con los **punteros** de los lenguajes de programación y los objetos persistentes.

Identidad de los objetos (*OID*)

Grados de persistencia.

- Dentro de los procedimientos
- Dentro de los programas
- Entre programas
- Persistente (Puntero persistente)

Almacenamiento

Guardar por separado la parte de **datos de cada objeto**.

El código que implementa los métodos de las clases debe guardarse en la base de datos como parte de su esquema

Acceso

- Dar nombres a los objetos
- exponer los identificadores de los objetos, que pueden guardarse de manera externa
- Guardar las colecciones de objetos y permitir que los programas iteren sobre estos . (Multiconjuntos)

Sistemas persistentes

- C++ y JAVA

Object Data Management Group (ODMG)

Formado por dos componentes:
- ODL (*Object Definition Language)*
- OML (*Object ManipulationLanguage)*

Lenguaje de consulta OQL (Object Query Language)

OODBS Ejemplos

Object Store

Objectivity

Db4o

Object Data Management Group (ODMG)

ODL: Object definition language

Sistema de tipos:

- Tipos primitivos: (int, varchar, boolean, double)
- Estructuras de registro
- Colecciones (Listas, conjuntos)
- Referencias a otros tipos de objetos (Punteros)

ODL: Object definition language

Sistema de tipos:

Sets: Conjuntos de elementos **desordenados y sin repetición**

Bags: Conjuntos de elementos desordenados y **con repetición**

List: Lista de elementos **ordenados**, con repetición

ODL: Object definition language

Definición de clases:

```
class Movie {
    attribute string title;
    attribute integer year;
    attribute integer length;
    attribute enum Film {color,blackAndWhite} filmType;
};
```

```
class Star {
    attribute string name;
    attribute Struct Addr
        {string street, string city} address;
};
```

ODL: Object definition language

Definición de relaciones:

```
class Movie {
    attribute string title;
    attribute integer year;
    attribute integer length;
    attribute enum Film {color,blackAndWhite} filmType;
    relationship Set<Star> stars
                 inverse Star::starredIn;
    relationship Studio ownedBy
                 inverse Studio::owns;
};

class Star {
    attribute string name;
    attribute Struct Addr
        {string street, string city} address;
    relationship Set<Movie> starredIn
                 inverse Movie::stars;
};
```

Relaciones directas

Relaciones inversas

ODL: Object definition language

Definición de relaciones:

```
class Movie {
    attribute string title;
    attribute integer year;
    attribute integer length;
    attribute enum Film {color,blackAndWhite} filmType;
    relationship Set<Star> stars
                 inverse Star::starredIn;
    relationship Studio ownedBy
                 inverse Studio::owns;
};
```

Relaciones directas

```
class Studio {
    attribute string name;
    attribute string address;
    relationship Set<Movie> owns
                 inverse Movie::ownedBy;
};
```

Relaciones inversas

ODL: Object definition language

Definición de métodos:

- En ODL se definen las declaraciones de los métodos (firmas), la implementación del método se hace en el lenguaje Host (Ej. C++, Java).
- Los métodos pueden recibir parámetros de entrada y salida
- Los métodos pueden lanzar excepciones

ODL: Object definition language

Definición de métodos:

```
class Movie {
    attribute string title;
    attribute integer year;
    attribute integer length;
    attribute enumeration(color,blackAndWhite) filmType;
    relationship Set<Star> stars
                 inverse Star::starredIn;
    relationship Studio ownedBy
                 inverse Studio::owns;
    float lengthInHours() raises(noLengthFound);
    void starNames(out Set<String>);
    void otherMovies(in Star, out Set<Movie>)
                 raises(noSuchStar);
};
```

Object Relational Model

Modelo de datos Objeto Relacional

Es esencialmente es un modelo relacional que permite a los usuarios **integrar** características orientadas a objetos a las bases de datos.

Extiende el modelo de datos relacional proporcionando un sistema de tipos más rico e incluyendo tipos de datos complejos y elementos de la **POO**

Bases de datos – Objeto relacionales

Extienden el modelo de datos relacional proporcionando un sistema de tipos más rico e icluyendo la programación orientada a objetos

Los lenguajes de consulta relacionales como SQL también necesitan ser extendidos para trabajar con el sistema de tipos enriquecido

En el modelo de datos objeto-relacional los dominios pueden ser **atómicos o de relación**

El valor de las tuplas de los atributos puede ser una relación, y las relaciones pueden guardarse **en otras relaciones**

Bases de datos – Objeto relacionales

Extienden el modelo de datos relacional proporcionando un sistema de tipos más rico e icluyendo la programación orientada a objetos

Los lenguajes de consulta relacionales como SQL también necesitan ser extendidos para trabajar con el sistema de tipos enriquecido

En el modelo de datos objeto-relacional los dominios pueden ser **atómicos o de relación**

El valor de las tuplas de los atributos puede ser una relación, y las relaciones pueden guardarse **en otras relaciones**

Bases de datos – Objeto relacionales

Extienden el modelo de datos relacional proporcionando un sistema de tipos más rico e icluyendo la programación orientada a objetos

Los lenguajes de consulta relacionales como SQL también necesitan ser extendidos para trabajar con el sistema de tipos enriquecido

En el modelo de datos objeto-relacional los dominios pueden ser **atómicos o de relación**

El valor de las tuplas de los atributos puede ser una relación, y las relaciones pueden guardarse **en otras relaciones**

Modelo de datos Objeto Relacional

Modelo de datos Objeto Relacional

Los dominios pueden ser atómicos o de **relación.**

name	*address*	*birthdate*	*movies*
Fisher	*street* / *city*: Maple / H'wood; Locust / Malibu	9/9/99	*title* / *year* / *length*: Star Wars / 1977 / 124; Empire / 1980 / 127; Return / 1983 / 133
Hamill	*street* / *city*: Oak / B'wood	8/8/88	*title* / *year* / *length*: Star Wars / 1977 / 124; Empire / 1980 / 127; Return / 1983 / 133

Modelo de datos Objeto Relacional

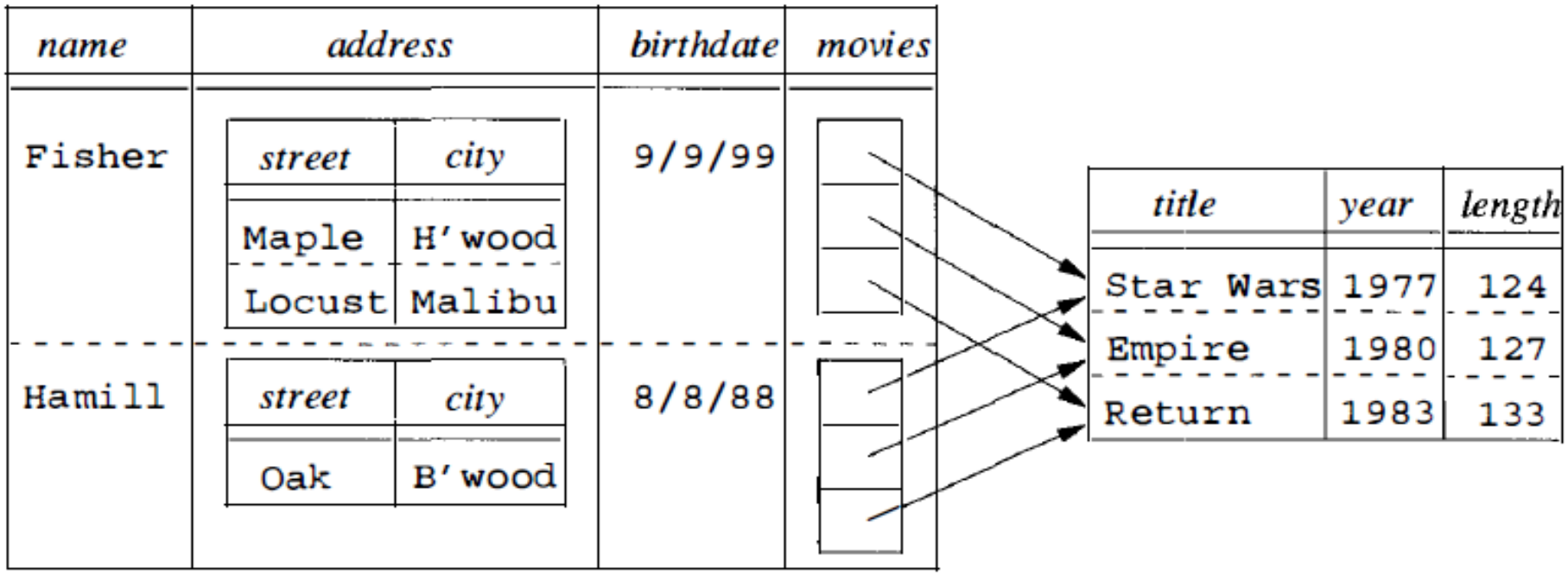
Tipos estructurados (structs): Tipos de datos definidos por el usuario

```
create type Editorial as
        (nombre varchar(20),
        sucursal varchar(20))
```

```
create table libros
        (título varchar(20),
        array-autores varchar(20) array [10],
        fecha-pub date,
        editorial Editorial,
        lista-palabras-clave setof(varchar(20)))
```

Modelo de datos Objeto Relacional

Referencias a otros objetos:



Modelo de datos Objeto Relacional

Métodos:

```
class Movie {
   attribute string title;
   attribute integer year;
   attribute integer length;
   attribute enumeration(color,blackAndWhite) filmType;
   relationship Set<Star> stars
                inverse Star::starredIn;
   relationship Studio ownedBy
                inverse Studio::owns;
   float lengthInHours() raises(noLengthFound);
   void starNames(out Set<String>);
   void otherMovies(in Star, out Set<Movie>)
                raises(noSuchStar);
};
```

create type *Empleado* **as** (
    nombre **varchar**(20),
    sueldo **integer**)
method *incrementar*(*porcentaje* **integer**)

Modelo de datos Objeto Relacional

Herencia: **Se puede hacer herencia de tipos o de tablas**

```
create type Persona
        (nombre varchar(20),
        dirección varchar(20))
```

```
create type Estudiante
        under Persona
        (curso varchar(20),
        departamento varchar(20))
create type Profesor
        under Persona
        (sueldo integer,
        departamento varchar(20))
```

Herencia de tipos

```
create table estudiantes of Estudiante
        under persona
create table profesores of Profesor
        under persona
```

Herencia de tablas

Modelo de datos Objeto Relacional

Tercer manifiesto

¿Cuál es el camino que se debe seguir para llegar a la próxima generación de DBMS?

Una transición gradual de la segunda generación de DBMS a una tercera.

Modelo de datos Objeto Relacional

Tercer manifiesto

- Además de la gestión de bases de datos tradicionales los DBMS tercera generación deben proporcionar apoyo a las **estructuras de objetos más ricos**

- Los DBMS de tercera generación deben incorporar los de segunda generación.

Soportar herencia

•

Modelo de datos Objeto Relacional

Tercer manifiesto

- Soportar **funciones definidas** por el usuario, métodos y encapsulación

- Identificadores únicos (**OI**D) sólo se deben asignarse si la clave primaria definida no está disponible

- Los **indicadores de desempeño** no deben aparecer en los modelos de datos

Modelo de datos Objeto Relacional

Tercer manifiesto

- Objetos persistentes
- Acceso a las bases de datos desde múltiples lenguajes
- SQL debe permanecer y evolucionar

Modelo de datos Objeto Relacional

Tercer manifiesto

- Las BDOO no es la solución a los retos que se encuentran en el uso de los DBMS segunda generación

- La experiencia y los conocimientos técnicos existentes en las bases de datos relacionales existentes deberían utilizarse tanto como sea posible

- El desarrollo basado en la actual segunda generación DBMS es más factible que un nuevo desarrollo desde cero

Resumen

Query	RDBMS	**ORDBMS**
No Query	File System	OODBMS
Data	Simple Data	Complex Data

Resumen

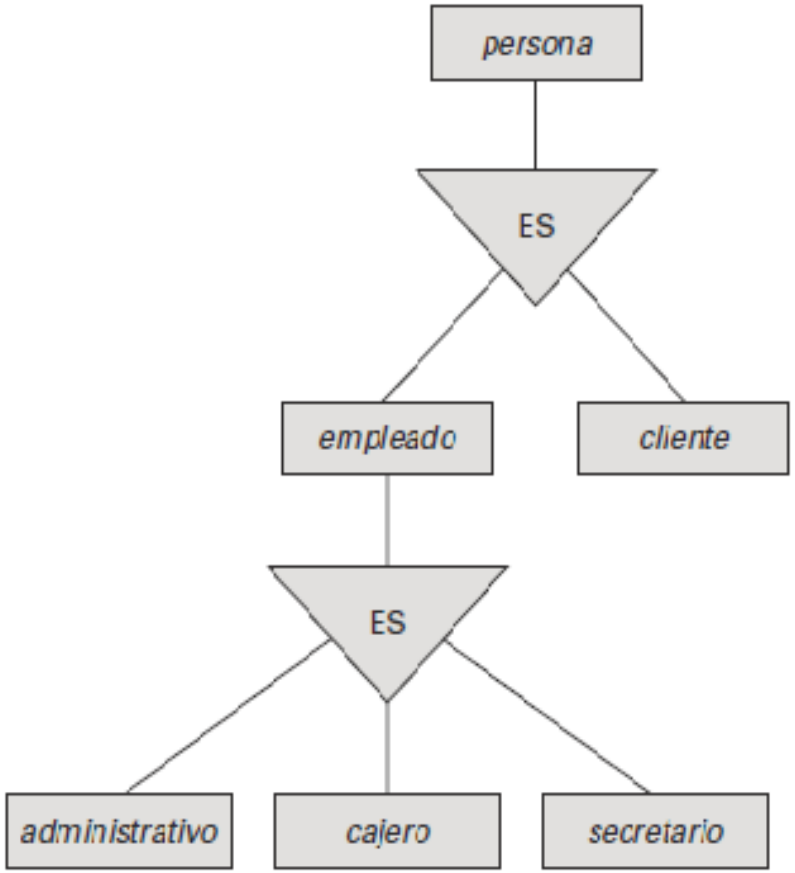
RDBMS Vs ORDMS

ORDMS Vs OODMS (Similitudes)

ORDMS Vs OODMS (Diferencias)

EJERCICIO

- Dado el siguiente Diagrama E-R, transfórmelo a un esquema del modelo Relacional y a un esquema del modelo objeto Relacional.



Object Relational Mapping (ORM)

Una **estrategia de persistencia** de objetos en una base de datos.

Una herramienta para **comunicar** la el mundo de la POO con una base de datos relacional.

Una herramienta para soportar la independencia de los datos y los lenguajes de programación **orientados a objetos.**

Object Relational Mapping (ORM)