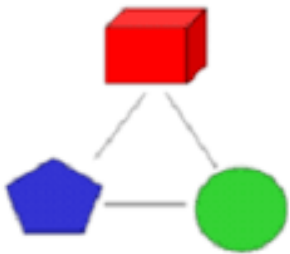


BASES DATOS ORIENTADA A OBJETOS

Conceptos Teóricos

“El paradigma de la complejidad” Edgar Morin



**ORIENTADA
A OBJETOS**

Docentes: Ing. Fernando Sato
A.S. Sebastian Trossero

Versión: 20181101

RESUMEN

Introducción a las Bases de Datos O.Objetos:

- ✓ Concepto de BDOO
- ✓ OMDG 2.0, el estándar
- ✓ ODL
- ✓ OQL

Bases de Datos Orientadas a Objetos

Motivación

El modelo relacional (BDR) se ha consolidado y ha probado su versatilidad para el desarrollo de aplicaciones de negocios tradicionales.

Las aplicaciones con mas complejidad como CAD/CAM, telecomunicaciones, CIM, GIS, multimedia, etc, ponen en jaque las estructuras de las BDR.

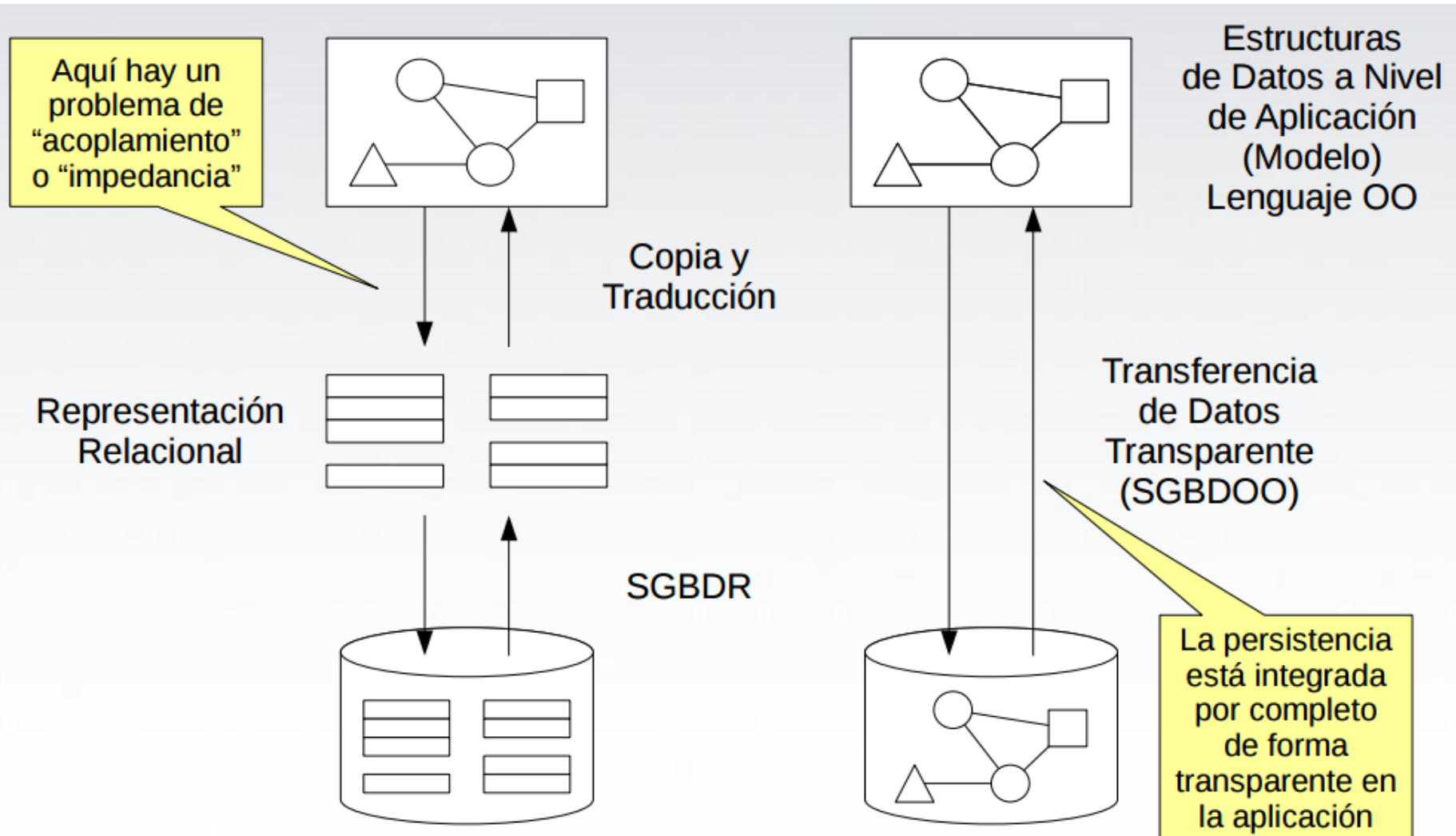
Los lenguajes OO (LPOO) han desplazado a los lenguajes estructurados basados en funciones.

Para utilizar LPOO con BD relacionales es preciso una capa que una los modelos Objeto \leftrightarrow Relacional :

- Materializar objetos a partir de la BD
- desmaterialización hacia la BD
- manejo de referencias entre objetos

Bases de Datos Orientadas a Objetos

Motivación



Bases de Datos Orientadas a Objetos

Desinflar objetos para INSERT(impedancia)

```
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**

Bases de Datos Orientadas a Objetos

Inflar objetos desde SELECT (impedancia)

```
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

Bases de Datos Orientadas a Objetos

Propuesta Paradigma p/persistir

```
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();
```

Acoplamiento Mínimo y Necesario.

la instancia p de tipo persona es persistido automáticamente

Bases de Datos Orientadas a Objetos

Motivación

Esto motivó a utilizar Orientación a Objetos tanto en los lenguajes como en las bases de datos.

El Modelo de Objetos puede representar mejor el mundo real.

Los objetos tienen un nombre, un estado (valor de los atributos y relaciones) y un comportamiento (operaciones que sabe hacer).

La Orientación a Objetos
Potencia el Reuso del Software.

Bases de Datos Orientadas a Objetos

Motivación

Se distingue la necesidad de 2 tipos de objetos:

Objetos que solo existan durante la ejecución del sistema, de un método o un programa.

Objetos que deban perdurar incluso ante el apagado del sistema y sus partes .

Objetivo de BDOO: mantener una correspondencia directa entre el mundo real y los objetos de la base de datos para que los objetos no pierdan su integridad e identidad y puedan ser fácilmente identificados y operados (**Perdurables**).

Bases de Datos Orientadas a Objetos

Características del Modelo – Los Objetos

El elemento mas importante del modelo es el **Objeto**:

- Tienen estado (valores) y comportamiento (operaciones)
 - Estado estructura datos complejas
- La BDOO debe proveer la persistencia mas allá de la ejecución de un programa, el objeto debe perdurar.
- Para que pueda ser recuperado y compartido por otras aplicaciones mas adelante se han incorporado otros mecanismos (indexado, control de concurrencia y recuperación), características típicas de los SGBDR.
- OID: Object Identifier valor único que no cambia para cada objeto creado (distinto al modelo relacional)
- Objeto permiten encapsulamiento.

Bases de Datos Orientadas a Objetos

Características del Modelo – Los Objetos

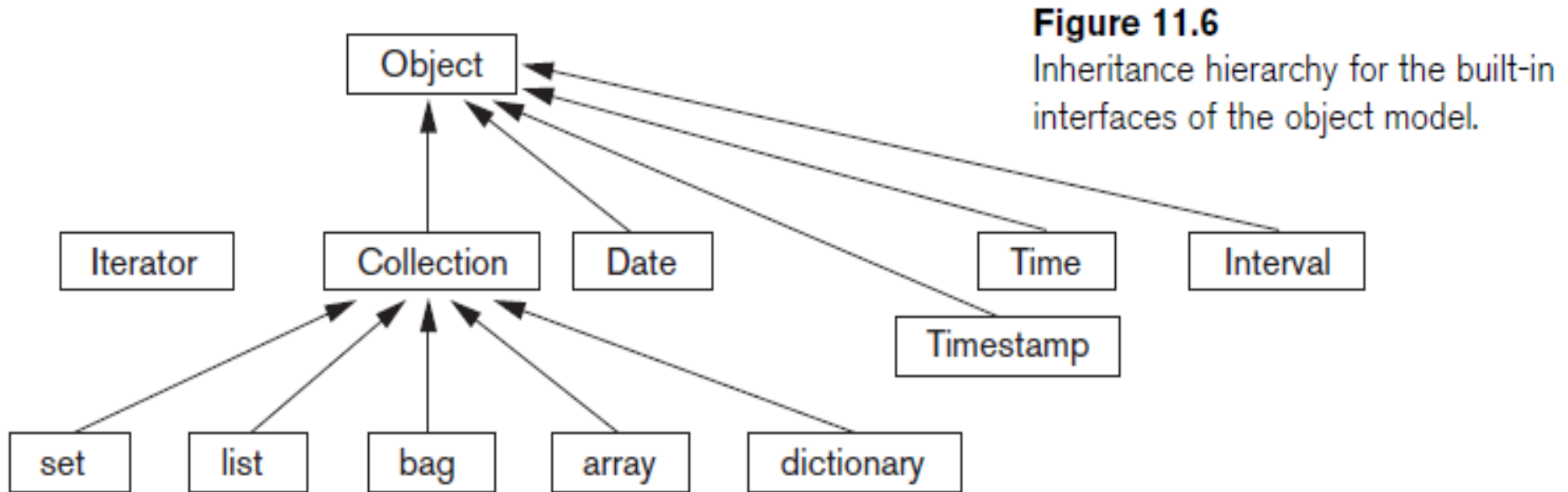
- OID: identidad única de cada objeto independiente de la BD
- OID es inmutable: el valor no cambia (como lo puede hacer una clave de la BD)
- OID es único para ese objeto, si el objeto es removido se lleva el OID con él y ningún otro objeto lo puede usar
- El objeto se identifica con una tripla (i, c, v)
 - ❑ i = OID
 - ❑ c = constructor de tipo
 - ❑ v = valor corriente que tiene el objeto

El modelo puede incluir varios constructores de tipo: **atom**, **tupla**, **set**, **lista**, **bag**, **array**, descripción:

- ❑ set/bag: conjunto de objetos sin orden
- ❑ lista: conjunto de objetos con orden sin tamaño definido
- ❑ array o vector: conjunto de objetos con orden y tamaño definido
- ❑ tupla: tipo estructurado (struct de C ó C++)

Bases de Datos Orientadas a Objetos

Jerarquía de Constructores



Fuente: FUNDAMENTALS OF Database Systems - Elmasri / Navathe

Bases de Datos Orientadas a Objetos

Características del Modelo – Los Objetos

➤ Herencia:

- simple: un supertipo puede ser heredado por uno o más subtipos
- múltiple: un subtipo puede heredar de más de un supertipo
- selectiva: un subtipo hereda solo una parte del supertipo (cláusula EXCEPT)

➤ Polimorfismo:

Sobrecarga de funciones y operadores

➤ Versiones y configuraciones

En algunas aplicaciones se necesitan generar nuevas versiones de algunos objetos sin descartar los existentes (diseño), muchos ODBMS tienen algunos sistemas que facilitan la generación de versiones del objeto.

Bases de Datos Orientadas a Objetos

Características del Modelo – Los Objetos

- **Comportamiento:** conjunto de operaciones (funciones) que se pueden aplicar externamente a un objeto. Estructura interna oculta – sólo se accede al objeto por medio de su comportamiento.
- **Constructores de tipos:** Métodos (función/operación) que se ejecuta cuando se crea un objeto a partir de la especificación del mismo (clase). La elección se realiza por los parámetros de entrada y sus tipos.
- **Destruyores:** Métodos que se ejecutan cuando se destruye un objeto.

Interfaz de una operación: nombre, tipo y parámetros

Cuerpo de una operación: implementación de la operación
(código de la operación)

Bases de Datos Orientadas a Objetos

Persistencia de Objetos

¿Cómo proveer persistencia a los objetos?

Problemas en la implementación de persistencia

- ❑ *¿Cómo representar objetos que pueden ser removidos y objetos que deben ser almacenados?*
- ❑ *¿Cómo acceder a objetos eficientemente?*
- ❑ *¿Dónde almacenar el código de una aplicación?*

Bases de Datos Orientadas a Objetos

Persistencia de Objetos

Objeto persistente

Objeto que sobrevive a la ejecución del proceso que lo creó.

Objeto transiente

Objeto que deja de existir cuando el proceso que lo creó termina su ejecución.

LPOO

Manipula Objetos Transientes y Persistentes

ODBMS

Manipula solo Objetos Persistentes

Bases de Datos Orientadas a Objetos

Modalidad de Gestionar la Persistencia

Persistencia explícita

Explícitamente se informa al sistema cuáles objetos son persistentes por medio de un mecanismo de nominación o de herencia de una clase de objetos persistentes.

Persistencia inferida

- El sistema infiere cuáles objetos son persistentes, por el hecho de estar asociados a otros objetos persistentes.
 - También denominada persistencia por alcanzabilidad:
 - si un objeto es persistente, todo objeto al cual el mismo referencia también es persistente
 - El mecanismo comienza con un conjunto de raíces :
 - todos los objetos "alcanzables" desde una raíz son persistente, ej.: "extent" (población) de una clase

Bases de Datos Orientadas a Objetos

Persistencia Explícita

Persistencia explícita

Consiste en dar al objeto un nombre único persistente (debe ser único dentro de la bd), de aquí en más el objeto es persistente.

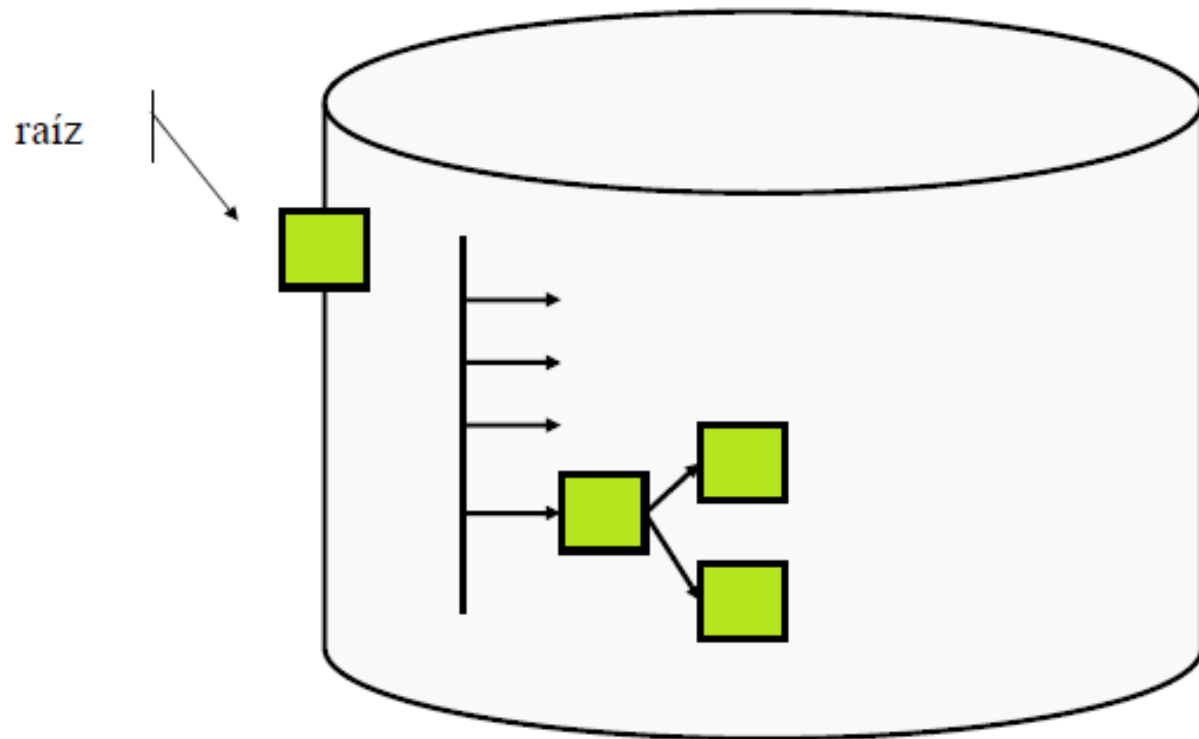
¿Como accedemos?

Asequibilidad: actúa haciendo que el objeto sea posible accederlo desde algún objeto persistente o desde el nombre.

Bases de Datos Orientadas a Objetos

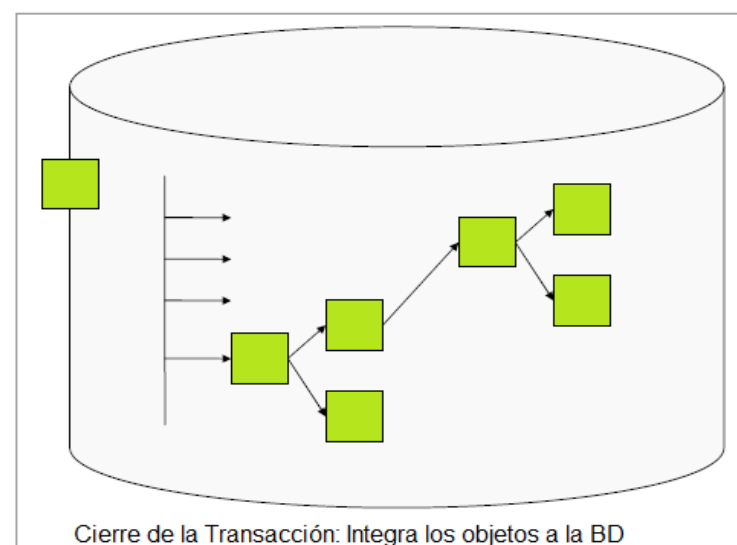
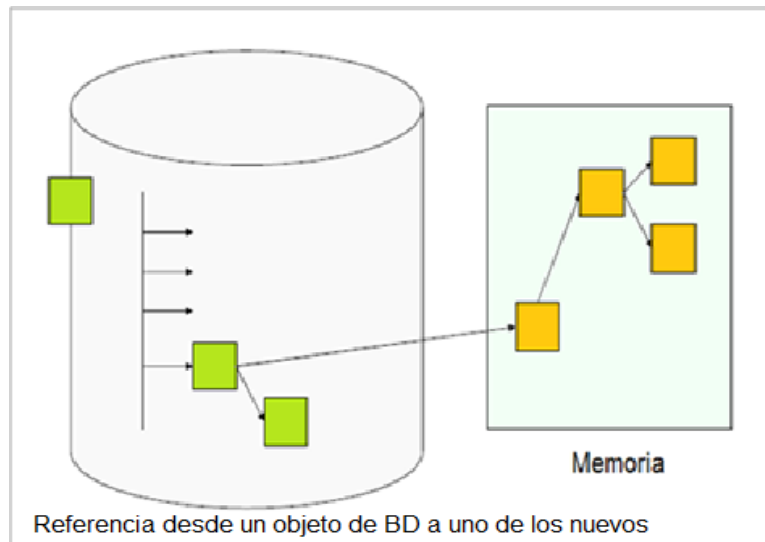
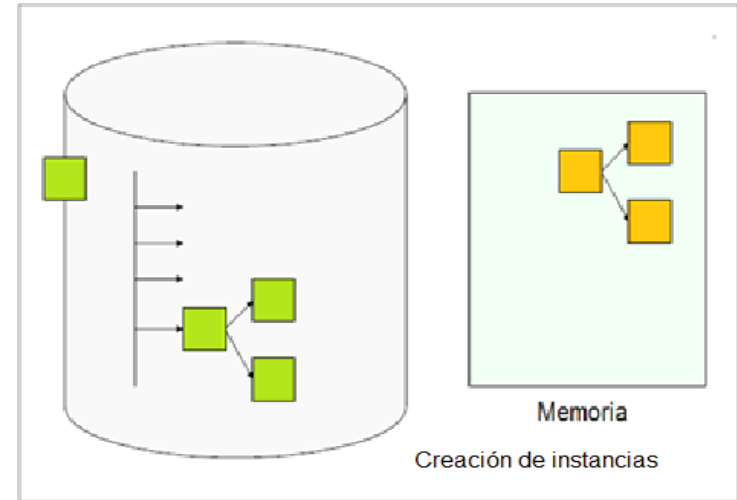
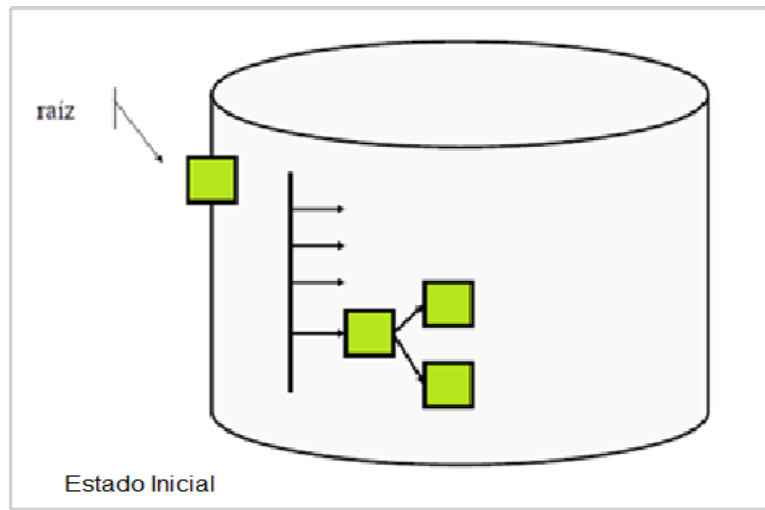
Persistencia Inferida

Existe un punto de acceso de la bd denominado raíz



Bases de Datos Orientadas a Objetos

Ciclo de Agregado Persistencia Inferida



Bases de Datos Orientadas a Objetos

ODMG 2.0 o superior, el estándar

Importante tener un estándar para un tipo específico de bases de datos

- Da soporte a la **portabilidad** de las aplicaciones de BD (ejecutar un programa en sistemas diferentes con mínimas modificaciones)
- Lograr **interoperabilidad** (acceder a múltiples sistemas diferentes)
- Permite que los clientes **comparen** productos comerciales más fácilmente

Bases de Datos Orientadas a Objetos

Estándares de BDOO

ODMG - Object Data(Base) Management Group

- ✓ consorcio formado por fabricantes de ODBMS
- ✓ define estándar de interface para ODBMS
- ✓ Abril 98: estándar de interface de LPOO para BD

Versiones

- ☐ ODMG 1.0 (1993)
- ☐ ODMG 1.2 (1995)
- ☐ ODMG 2.0 (1997)
- ✓ SQL/3 ó SQL:1999 (ORDBMS)
 - ☐ estándar para BD Objeto-Relacional
 - ☐ ODMG 3.0 (2001) – (Enlace a c++, java, smalltalk)

Bases de Datos Orientadas a Objetos

ODMG 3.0, el estándar

Esta compuesto de varias partes:

- El modelo de objetos
- El lenguaje de definición de objetos (ODL)
- El lenguaje de consulta de objetos (OQL)
- Enlaces con LPOO (C++, SMALLTALK, JAVA)

Bases de Datos Orientadas a Objetos

Modelo de Objetos ODMG

El Modelo de Objetos, trataremos:

- **Objetos y Literales**
- Interfaces predefinidas de construcción para objetos de colección
- Objetos atómicos (definidos por el usuario)
- Herencia: Interfaces y clases
- Extensiones, claves y objetos factoría

Bases de Datos Orientadas a Objetos

Objetos y Literales

Un **objeto** se describe con 4 características:

- **Identificador de objeto (OID)** único en el sistema
- Pueden recibir opcionalmente un **nombre** único, para hacer referencia al objeto desde un programa
- **Tiempo de vida**: objeto persistente u objeto transitorio
- **Estructura**: objeto atómico u objeto de colección

Bases de Datos Orientadas a Objetos

Objetos y Literales

Un literal no tiene OID y su estructura puede ser compleja o simple

Tres tipos :

- **Atómicos** (tipos de datos básicos: long, short, unsigned long, float, double, boolean, char, string y enum)
- **Estructuras** : son los que se crean mediante el constructor de tuplas "**struct**".
- **Colección** : conjunto de objetos o valores (NO tienen OID)

Bases de Datos Orientadas a Objetos

Objetos y Literales

Todos los objetos heredan la interfaz básica de **Object**, y por tanto unas operaciones básicas:

- Copy (copiar)
- Delete (eliminar)
- Same_as (igual a)

Las operaciones se aplican a los objetos mediante:

- Notación de punto (`o.same_as(p)`, `p=o.copy()`)
- Notación de flechas (`o→same_as(p)`, `p→copy()`)

Bases de Datos Orientadas a Objetos

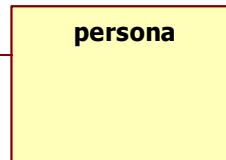
Interfaces

Persona implementa las interfaces huesped y pasajero.

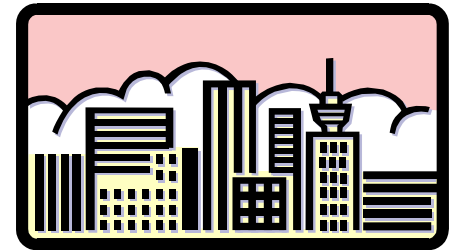


Aerolinea

Pasajero



huesped



Hotel

```
Interfaz Pasajero {  
    DespacharEquipoje()  
    Embarcar()  
}
```

```
Interfaz Huesped {  
    Checkin()  
    Checkout()  
}
```

Bases de Datos Orientadas a Objetos

Itfz definidas para Colección de Objetos

Todo objeto colección hereda la interfaz **Collection** y sus operaciones:

- `o.cardinality()`
- `o.is_empty()`
- `o.insert_element(e)`
- `o.remove_element(e)`
- `o.contains_element(e)`

Con `i=o.create_iterator()` se crea un objeto **iterador** y tiene estas operaciones:

- `i.reset()`
- `i.next_position()`
- `i.get_element()`

Bases de Datos Orientadas a Objetos

Itfz definidas para Colección de Objetos

Los objetos colección se especializan en Set, List, Bag, Array y Dictionary que heredan las operaciones de la interfaz Collection

- **Set<t>**: conjunto de elementos de tipo, incluyen:

```
o.create_union(s),  
o.create_intersection(s),  
o.create_difference(s),  
o.is_subset_of(s)
```

- **Bag<t>**: permite la repetición de elementos (unión, intersección y diferencia) y

```
o.occurrences_of(e) : numero de veces que  
aparece el elemento e
```

Bases de Datos Orientadas a Objetos

Itfz definidas para Colección de Objetos

- **List<t>**: lista ordenada, elementos son de tipo t

```
l.insert_element_first/last(e),  
l.insert_element_after/before(e,i), o  
l.remove_first/last_element(),  
l.remove_element_at(i),  
l.retrieve_first/last_element(),  
l.retrieve_element_at(o), l.concat(o),  
l.append(o)
```

- **Array<t>**: como la lista, pero con n° fijo de elementos

```
a.repalce/remove/retrieve_element_at(i,e),  
a.resize(n)
```

Bases de Datos Orientadas a Objetos

Itfz definidas para Colección de Objetos

- **Dictionary** $\langle k, v \rangle$: colección de parejas de asociaciones $\langle k, v \rangle$, cada K es una clave única asociada con un valor v

`o.bind(k, v)`

`o.unbind(k, v)`

`o.lookup(k)`

`o.contains_key(k)`

Bases de Datos Orientadas a Objetos

Itfz definidas para Colección de Objetos

ODMG también utiliza **excepciones** para informar sobre errores o condiciones concretas:

- `ElementNotFound`
- `NoMoreElements`
- `InvalidIndex`
- `KeyNotFound`

Bases de Datos Orientadas a Objetos

Objetos Atómicos

Es cualquier objeto que no sea un objeto colección y se crean a partir de una clase (definición de objeto) con la palabra reservada ***new***.

Básicamente especifican sus propiedades que definen el estado (atributos y relaciones) y las operaciones que definen el comportamiento

- **Atributo:** es una propiedad que describe algún aspecto de la clase (por ende de los objetos). Los valores suelen ser literales, aunque también pueden ser objetos.

Bases de Datos Orientadas a Objetos

Objetos Atómicos

- **Relación:** es una propiedad que especifica si dos objetos están **relacionados** entre sí. Sólo se representan las relaciones binarias bidireccionales mediante referencias inversas (relationship)
- **Operaciones:** cada tipo de objetos puede tener varias **signaturas de operaciones**, que especifican el nombre (único en cada tipo de objeto), tipos de argumentos y el valor devuelto. También se pueden especificar **excepciones** que pueden darse durante la ejecución de la operación

Bases de Datos Orientadas a Objetos

Herencia de Clases e Interfaces

- ❑ **Interfaz**: especificación del comportamiento abstracto de un tipo de objeto, **no es instanciable**
- ❑ **Clase**: especificación tanto del comportamiento como del estado y sí son **instanciables**
- ❑ Dos tipos de **herencia**:
 - Herencia de comportamiento (:) :** para heredar sólo operaciones (Interfaz -> Interfaz/Clase)
puede ser múltiple
 - Extensión(extends) :** para heredar el estado y el comportamiento (Clase -> Clase), **NO SOPORTA HERENCIA MULTIPLE.**

Bases de Datos Orientadas a Objetos

Extensiones, Claves y Objetos Factoría

Extent: se puede declarar una extensión para cada tipo de objetos.

- Tiene nombre y contendrá todos los objetos persistentes de la clase (se comporta como un conjunto) y puede tener una o más claves.

Objetos fabrica o factoría: heredan la interfaz **ObjectFactory** (**new()**) y sirven para generar o crear objetos mediante sus operaciones (proporciona las operaciones constructoras para la creación de nuevos objetos)

Bases de Datos Orientadas a Objetos

Extensiones, Claves y Objetos Factoría

Clave: está compuesta por una o más propiedades y sus valores están restringidos a ser únicos

Base de datos: como puede crear muchas bases de datos diferentes cada una debe tener

- Nombre de base de datos
- **Bind (ligar)** para asignar nombre únicos a objetos persistentes
- **Lookup** (buscar)
- **Unbind** (desligar)

Bases de Datos Orientadas a Objetos

Seguimos con

-ODL

Bases de Datos Orientadas a Objetos

ODL Lenguaje de Definición de Objetos

El ODL está diseñado para soportar constructores semánticos de ODMG y no depende de ningún lenguaje de programación específico

No es un lenguaje de programación completo, es un lenguaje de definición.

Se pueden utilizar ligaduras específicas para transformar los constructores de ODL en elementos de un LPOO como C++, SMALLTALK o JAVA

Bases de Datos Orientadas a Objetos

ODL

La implementación de operaciones de clases en un sistema ODMG puede tener su código escrito en uno de estos lenguajes: c++, java, smalltalk.

Sintaxis similar a SQL pero con características adicionales propias del paradigma objetos:


- *Identidad de objetos*
- *Objetos complejos*
- *Operaciones*
- *Herencia*
- *Polimorfismo*
- *Relaciones*

Bases de Datos Orientadas a Objetos

ODL – Objetos Atómicos

Son Objetos definidos por el usuario y se definen a través de la palabra clave `class`.

```
class Persona (extent personas key dni) {  
    /* Definición de atributos */  
    attribute string nombres;  
    attribute string apellidos;  
    attribute string dni;  
    attribute date fecha nacim;  
    attribute enum Genero{F,M} sexo;  
    /* Definición de operaciones */  
    float edad();  
}
```



Contenedor
con clave
dni

Bases de Datos Orientadas a Objetos

ODL – Objetos Atómicos - Extent

El `Extent` es equivalente a la tabla de BDR:

- ❑ Cada `Extent` se le da un nombre y guarda los objetos persistidos.
- ❑ Por EJ: Para `Persona` el extent se denomina `Personas`.
- ❑ Esto es similar a la creación de un objeto de tipo `Set <Persona>` y hacerlo persistente.
Es decir, `personas` guarda un Set de objetos persona.

Bases de Datos Orientadas a Objetos

ODL – Objetos Atómicos - Key

Una **key** de una clase consiste de uno o mas atributos únicos. Concepto similar a Primary key de Bdr.

- ❑ Para la **class** Persona la clave es dni.
 - ❑ Por lo tanto se espera que cada empleado tenga un dni único.
- ❑ Las claves pueden ser compuestas:
 - ❑ Ejemplo (key documento, tipoDocumento)

Bases de Datos Orientadas a Objetos

ODL – Objetos Atómicos – Objeto Fabrica

Un objeto fabrica (object factory) se utiliza para generar los objetos individuales a través de sus operaciones. Un ejemplo:

```
Interface ObjectFactory {  
    Object new() ;  
};
```

`new()` devuelve nuevos objetos con un `object_id`

Uno puede crear su propia interfaz de fábrica al heredar la interfaz anterior

Bases de Datos Orientadas a Objetos

ODL – Clases e Interfaces

ODMG soporta dos conceptos para especificar los tipos de objetos:

- Interfaz
- Clase

- ❑ Existen similitudes y diferencias entre las interfaces y las clases
- ❑ Ambos tienen comportamiento (operaciones), las interfaces pueden tener atributos o no.

Bases de Datos Orientadas a Objetos

ODL –Interfaces

Una **interfaz** es una especificación de comportamiento abstracto de un objeto.

- Una interfaz no se puede instanciar.
- Ejemplo interfaz *Object*: aporta métodos básicos que comparten todos los objetos, pero no puede existir o crearse un objeto del tipo *Object*.

Bases de Datos Orientadas a Objetos

ODL – Ejemplo de Interfaz

ODL creación de interface.

```
Interface Forma {  
    /* Definición de atributos */  
    attribute struct spunto  
        { integer x, integer y} puntoReferencia;  
    /* Definición de operaciones */  
    float perimetro();  
}
```


Bases de Datos Orientadas a Objetos

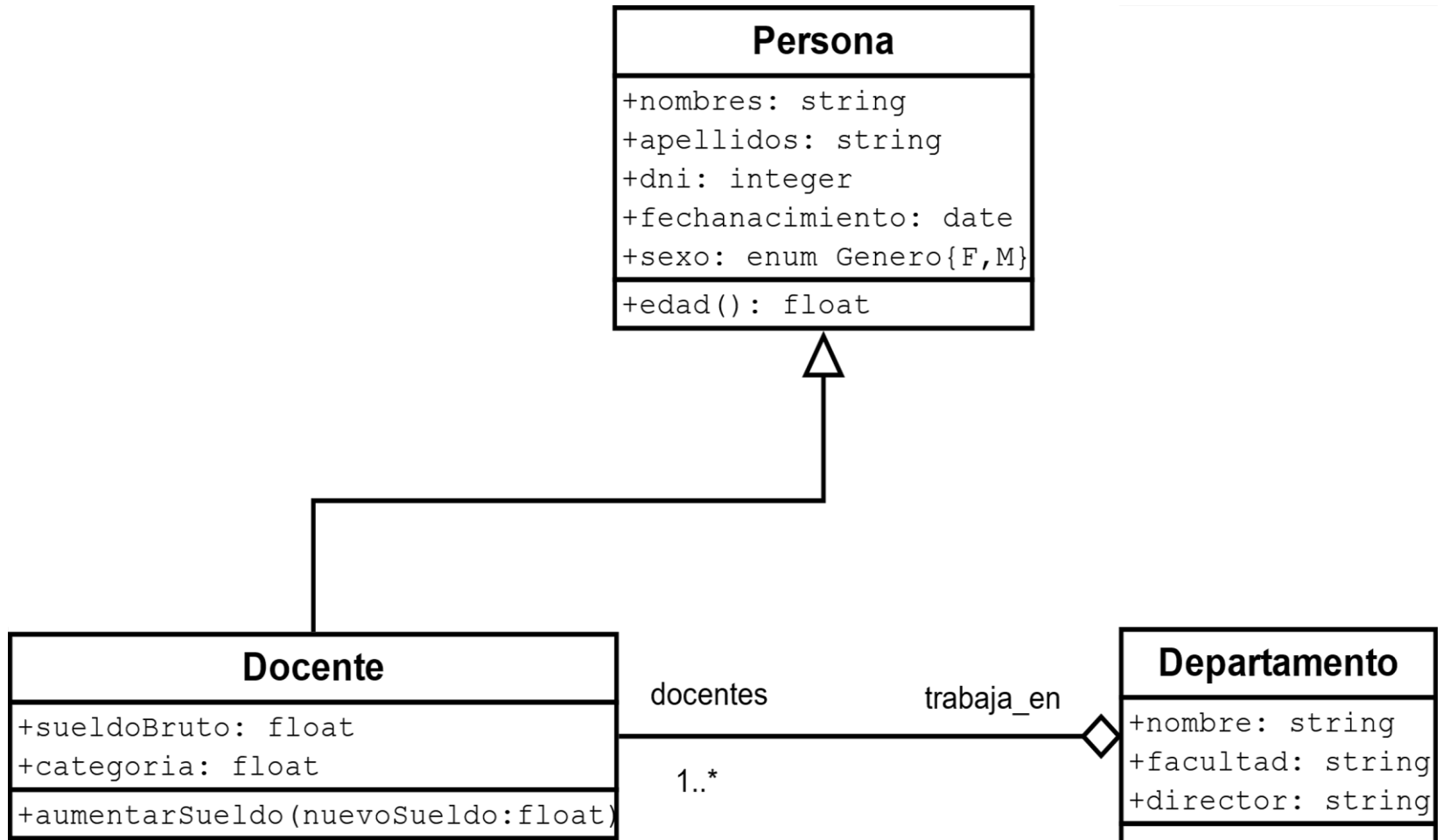
ODL – Clases

Una **clase** es una especificación de comportamiento y estado.

- Una clase es instanciable.
- Soporta la clausula **Extent**.

Bases de Datos Orientadas a Objetos

ODL – Casos



Bases de Datos Orientadas a Objetos

ODL – Class

ODL creación de class.

```
class Persona (extent personas key dni) {  
    /* Definicion de atributos */  
    attribute string nombres;  
    attribute string apellidos;  
    attribute string dni;  
    attribute date fechaNacimiento;  
    attribute enum Genero{F,M} sexo;  
    /* Definición de operaciones */  
    float edad();  
}
```

Bases de Datos Orientadas a Objetos

ODL – Herencia de clases

La clase Docente hereda de la class Persona. Clausula **extends**.

```
class Docente extends Persona(extent docentes) {  
    /* Definición de atributos */  
    attribute float sueldoBruto;  
    attribute string categoria;  
  
    /* Relaciones */  
    relationship Departamento trabaja_en inverse  
    Departamento::docentes;  
    /* Operaciones */  
    void aumentarSueldo( float nuevoSueldo);    }
```

Bases de Datos Orientadas a Objetos

ODL – Relación 1 a muchos

La clase Departamento tiene varios docentes.

```
class Departamento (extent Departamentos) {  
    /* Definición de atributos */  
    attribute string nombre;  
    attribute string facultad;  
    attribute string director;  
  
    /* Relaciones */  
    relationship set<Docente> docentes inverse  
    Departamento::trabaja_en;  
}
```

Bases de Datos Orientadas a Objetos

ODL – Herencia de Interfaz

La clase Triangulo hereda de la Interfaz Forma. Operador ":".

```
class Triangulo : Forma (extent Triangulos) {  
  
    /* Definición de atributos propios */  
    attribute float lado1;  
    attribute float lado2;  
    attribute float lado3;  
  
}
```

Bases de Datos Orientadas a Objetos

Seguimos con

- OQL

Bases de Datos Orientadas a Objetos

OQL

Lenguaje consulta objetos propuesto para ODMG.

Diseñado para operar C++, SMALLTALK y JAVA.

Una consulta insertada en uno de estos lenguajes de programación puede obtener como resultado objetos que coinciden con el sistema de tipos de dicho lenguaje.

Bases de Datos Orientadas a Objetos

Consultas OQL

Sigue estructura

```
SELECT d.nombre  
FROM d in departamentos  
WHERE d.facultad = 'Ingenieria';
```

Se necesita un punto de entrada que puede ser cualquier objeto persistente, para muchas es el nombre de extensión de su clase.

Cuando usamos como entrada nombre de una extensión se debe definir una variable iterador(d)

Bases de Datos Orientadas a Objetos

Consultas OQL

Un iterador se puede definir también como:

- ❑ `d in departamentos` (además de este)
- ❑ `departamentos d`
- ❑ `departamentos as d`

```
SELECT d.nombre  
FROM departamentos d  
WHERE d.facultad = 'Ingenieria';
```

En todos los casos `d` es un iterador del set de objetos `departamento` que tiene el **extent** `departamentos`.

Bases de Datos Orientadas a Objetos

Consultas OQL

Una consulta no tiene que seguir la estructura `select..from..where...`

Cualquier nombre constituye una consulta, cuyo resultado es una referencia a dicho objeto.

C1: `Departamentos;`

Devuelve referencia a la colección todos objetos persistentes de departamento.

Bases de Datos Orientadas a Objetos

Consultas OQL

C2: `Departamentoinf;`

Devuelve referencia a ese objeto individual de tipo departamento apuntado por el nombre persistente `Departamentoinf`.

Bases de Datos Orientadas a Objetos

Consultas OQL

Una especificación de camino permite navegar entre las relaciones y atributos. Se separa con punto

C3: `departamento.inf.director;`

Devuelve la referencia al objeto profesor que es director del departamento de informática.

Bases de Datos Orientadas a Objetos

Consultas OQL – Restricciones de Tipo

Supongamos que se quiere obtener la categoría de los docentes de informática.

```
Departamentoinf.docentes.categoria;
```

Seria invalido, no se sabe si devuelve
set<string> o bag<string>

C3: Select distinct di.categoria **from** di **in**
departamentoinf.docentes; **→ set**

C3: Select di.categoria **from** di **in**
departamentoinf.docentes; **→ bag**

Validas

Bases de Datos Orientadas a Objetos

Otras Características de OQL

Vistas

Extracción elementos únicos.

Operadores de colección

Expresiones de colección ordenadas.

El operador agrupación

Bases de Datos Orientadas a Objetos

Vistas

Definición: una vista especifica el identificador para consultas con nombre, debe ser único entre todos los objetos. Puede tener parámetros

```
V1: DEFINE valumnosasignaturas(nombre_dept) as  
    SELECT a FROM a in alumnos  
  
    WHERE a.asig_secun_en.nombre = nombre_dept;
```

Esta vista define una consulta con nombre `valumnosasignaturas` para obtener el conjunto de alumnos con asignaturas en un departamento dado.

Bases de Datos Orientadas a Objetos

Extracción de Elementos Únicos (OQL)

Operador **ELEMENT** que garantiza la obtención de un único elemento e de una colección c que contenga un solo elemento.

`e = ELEMENT(Colección c)`

Si c contiene mas de un elemento o si vacío
ELEMENT genera una excepción

```
C4: ELEMENT (SELECT d  
               FROM d in departamentos  
               WHERE d.nombre = 'Informatica');
```

C4 Devuelve la referencia única al departamento de informática o una excepción.

Bases de Datos Orientadas a Objetos

Op de colección (OQL)

C5: **COUNT** (a **in**
tiene_asig_secun('Informatica'));

C5 obtiene el número de alumnos que tienen asignaturas secundarias de informática.

C6: **AVG**(**SELECT** a.nota
 FROM a **in** alumnos
WHERE a.especialidad_en.nombre = 'Informatica'
 and a.clase = 'ultimo curso');

C6 obtiene el valor medio de las notas de todos los alumnos de ultimo curso de informática.

Estas devuelven literales.

Bases de Datos Orientadas a Objetos

Expresiones de Colección Ordenadas(OQL)

```
C7: FIRST ( SELECT
    STRUCT (profesorado:
p.nombre.apellido,  salario:p.salario)
    FROM p in profesorado
    ORDER BY p.salario desc);
```

C7 obtiene el miembro del profesorado con el salario mas alto.

Bases de Datos Orientadas a Objetos

El Operador agrupación (OQL)

```
C8: SELECT STRUCT (nombre_dept,  
    num_alum_con_especialidad:count(partition))  
    FROM a in alumnos  
    GROUP BY nombre_dept:a.especialidad_en-nombred;
```

C8 obtiene el numero de alumnos con especialidad en cada departamento.

Los alumnos se agrupan en la misma partición si tienen la misma especialidad.

Seguimos con

Ventajas y Desventajas

Implementaciones

Bases de Datos Orientadas a Objetos

Ventajas de BDOO

- Soporte para el manejo de tipos de datos complejos (modificaciones con herencia)
- La segunda ventaja, es que manipula datos complejos en forma rápida y ágilmente, debido a las referencias o apuntadores lógicos entre objetos

Bases de Datos Orientadas a Objetos

Desventajas de BDOO

- La inmadurez del mercado de BDOO constituye una posible fuente de problemas
- El segundo problema es la falta de estándares en la industria orientada a objetos, aunque cada vez tiene más fuerza el estándar ODMG

Peor noticia:

Abarcan solamente un 1% del mercado

Bases de Datos Orientadas a Objetos

Implementaciones SGBDO

- ✓ ObjectStore (Adquirida por Progress)
- ✓ Objectivity/DB (Objectivity)
<http://www.objectivity.com/>
- ✓ ONTOS (Ontologic)
- ✓ VERSANT (Versant Object Technology)
<http://www.versant.com/>
- ✓ GemStone (GemStone)
<http://www.gemstone.com/>
- ✓ Jasmine (CA)
<http://www.cai.com/>

Bases de Datos Orientadas a Objetos

Fuentes

Libro: Fundamentos de Bases de Datos

Autor: Elmasri / Navathe

Editorial: Pearson

Libro: The Object Data Standard – ODMG 3.0

Autor: R. Cattell / D. Barry

Editorial: Morgan Kaufmann Publishers