BASES DE DATOS (IG18 Semipresencial)
Otros Modelos de Bases de Datos. El modelo orientado a objetos y objeto-relacional

Lledó Museros / Ismael Sanz
museros@icc.uji.es / isanz@icc.uji.es

- Lo que ya sabemos:
  - **Modelos de datos,** herramientas que permiten definir la realidad.
  - Contenidos de un modelo de datos:
    - **Datos:**
      - Datos o entidades
      - Propiedades de los datos
      - Relaciones entre los datos
      - Restricciones de los datos
  - El modelo representa **entidades genéricas** → Construcción de **esquemas.**
  - Los valores concretos se denominan **ocurrencias o instancias** → **Estado** de la BBDD

- Durante los años 80 y 90 hubo gran interés por las bases de datos orientadas a objetos. Se pensaba que podían ser una serias competidoras de los SGBDR. Aunque tienen su nicho, éste está limitado a casos muy concretos (aunque esto es discutible).
- Modelos tradicionales (relacional, jerárquico y red) han tenido éxito en aplicaciones tradicionales de negocios. Sin embargo presentan **deficiencias** en el modelado de datos de **aplicaciones más complejas** (CAD, CASE, Multimedia, Sistemas Expertos, ...) → Requisitos (objetos complejos, datos de comportamiento, etc) y características diferentes.
- Modelo Orientado a Objetos (OO) ofrece flexibilidad para cumplir estos requisitos.
- Permite **definir** tanto la **estructura** de los objetos como las **operaciones**.
- BBDD tradicionales se relacionan mal con aplicaciones orientadas a objetos.Las BBDD OO se **integran** fácilmente con **aplicaciones OO.**

- Se trata de una técnica para transformar los datos de un lenguaje OO en datos persistentes que se almacenan en una BBDD Relacional.
- En la programación OO, las tareas de manejo de datos son implementadas generalmente por la manipulación de objetos.
- Sin embargo, los SGBDR, solamente pueden almacenar y manipular valores escalares organizados en tablas.
- El programador debe convertir los valores de los objetos en grupos de valores simples para almacenarlos en las BBDD (y volverlos a convertir luego de recuperarlos de la BBDD)
- Es muy complicado traducir estos objetos a formas que se puedan almacenar en la BBDD, y que puedan ser recuperados fácilmente (preservando las propiedades de los objetos y sus relaciones).

- **Prototipos y Sistemas Orientado a Objetos**
  - **Sistemas disponibles en el mercado:**
    - **GEMSTONE/OPAL**
    - **ONTOS**
    - **Versant de Versant Object Technologies**
    - **Jasmine**
    - **....**
  - **Prototipos:**
    - **ORION de MCC, Austin, Texas**
    - **OPENOODB de Texas Instruments**
    - **IRIS de HP**
    - **ODE de ATT Labs**
    - **.....**

- Dado que las necesidades de los nuevos tipos de aplicaciones se han utilizado durante años en la Programación Orientada a Objetos (POO), se ha tratado de **incorporar los mismos conceptos de la POO** en los sistemas de bases de datos.
- Los orígenes de los SGBDOO están en la POO.
- La idea básica es que el usuario no ha de trabajar con estructuras de datos de bajo nivel (bits, bytes o incluso campos y registros) sino que debe trabajar con objetos y sus operaciones.
- En la POO introducir "inteligencia" en los objetos permite reducir código, aumentar la productividad, …. En los SGBDOO puede facilitar, **aunque también puede complicar** ciertas operaciones con la BBDD.

- Necesidad de un estándar
  - Ante la existencia de tantos sistemas, surge la necesidad de un **modelo estándar.**
  - Se formó el ODMG (Object Database Management Group) formado por un consorcio de suministradores y usuarios.
  - Se han definido distintas versiones:
    - ODMG-93 (versión 1.0)
    - ODMG 2.0 (1997)
    - ODMG 3.0 (2000)
  - La arquitectura ODMG está formada por:
    - El Modelo de objetos
    - Un lenguaje de definición de objetos (ODL)
    - Un lenguaje de consulta de objetos (OQL)
    - “Enlaces” con los lenguajes de POO: C++, Smaltalk, Java….

- Programación Orientada a Objetos (POO):
  - **La POO** se basa en los Tipos Abstractos de Datos (**TAD**): Oculta las estructuras internas de datos y especifíca todas las posibles operaciones externas aplicables a un objeto.
  - Un **objeto** tiene dos componentes: su estado y su comportamiento.
  - **Herencia:** Las clases se organizan de forma jerárquica. Puede ser múltiple y selectiva.
  - **Polimorfismo:** sobrecarga de operadores.
  - **Objetos complejos estructurados:** se definen aplicando repetidamente los constructores de tipos proporcionados por el lenguaje de POO (LPOO).
  - Los objetos de un LPOO **existen sólo durante la ejecución del programa.**

- La arquitectura ODMG está formada por:
  - El Modelo de objetos
  - Un lenguaje de definición de objetos (ODL)
  - Un lenguaje de consulta de objetos (OQL)
  - “Enlaces” con los lenguajes de POO: C++, Smaltalk, Java….

- Modelo de Objetos en ODMG:
  - Especifica las características de los objetos, cómo se relacionan, cómo se identifican, construcciones soportadas, …
  - Las primitivas básicas son:
    - **Los objetos,** caracterizados por un identificador, un nombre un tiempo de vida y una estructura.
    - **Los literales,** son objetos que no tienen identificador.
  - Los objetos puede ser:
    - Atómicos, incluyendo los estructurados.
    - Colección:
      - **Conjunto:** grupo de objetos del mismo tipo desordenado. No permite duplicados.
      - **Bolsa:** grupo de objetos del mismo tipo desordenado. Permite duplicados.
      - **Lista:** grupo de objetos del mismo tipo ordenado. Permite duplicados.
      - **Vector:** grupo de objetos del mismo tipo ordenado. Se accede por posición. Tamaño dinámico
      - **Diccionario:** grupo de objetos del mismo tipo. Cada valor está asociado a una clave.

- Modelo de Objetos en ODMG:
  - Clases
    - Una clase es la especificación del estado y el comportamiento de un tipo de objetos.
    - Una clase es un tipo de objetos asociado a un “extent”
    - Puede incluir métodos
  - Cada objeto en ODMG (como en el mundo real) es único.

- ODL:
  - **Object Definition Language (ODL)** es el lenguaje estándar que sirve para especificar la estructura de una BBDDOO.
  - Se trata del DLL de una BBDDOO.

```
Class Persona
(extent personas key DNI)
{attribute struct nombreP {string apellidos,
                           string nombre} nombre;
attribute string DNI;
attribute date fnac;
attribute enum{H,M} sexo;
attribute set<string> telefonos;
short edad();}
```

```
Class Cliente extends Persona
(extent clientes)
{attribute struct nc {short nb,short ns,
                      short cc,short nc} cuenta;
relationship Set<Factura> tiene
inverse Factura::pertenece_a}
```

```
Class Factura
(extent facturas key codfac)
{attribute string codfac;
attribute date fecha;
attribute list<linea_fac> lineas;
relationship Cliente pertenece_a
inverse Cliente::tiene}
```

```
Class linea_fac
(extent lineas key numero)
{
attribute short numero;
attribute Articulo artic;
attribute
...
```

- OQL:
  - Object Query Language (OQL) es el lenguaje estándar de consultas de BBDDOO.
  - Es un lenguaje declarativo del tipo SQL. Es sólo de consulta.
  - La sintaxis básica es del tipo `SELECT… FROM… WHERE…`

```
SELECT c.nombre
FROM c in clientes
WHERE sexo=M
```

  - Permite la especificación de vistas
  - Dispone de operadores sobre colecciones: funciones de columna (`max, min, count, …`) y cuantificadores (`for all` y `exists`).

- Permiten crear **objetos permanentes**: persisten
- Objetos **compartidos** por distintos programas
- Incluyen mecanismos de indexación, control de concurrencia y recuperación
- Permiten la **comunicación con LPOO**
- Proporcionan un **identificador de objetos único** a cada objeto
- **(OID)**
- La estructura del objeto puede ser de **complejidad arbitraria**
- **Independencia entre datos y operaciones.**
- **Herencia**: Definición de nuevos tipos en base a otros ya definidos
- **Polimorfismo de Operadores**
- Definición de **diferentes versiones** de un mismo objeto
- Definición de relaciones entre objetos mediante **referencias inversas** utilizando los OIDs

- Surge como consecuencia de las incorporación de las características del modelo OO a los ya existentes sistemas Relacionales: Oracle, Informix, PostgreSQL, …
  - Soporte de tipos básicos y complejos (vídeo, audio, documentos de texto, …)
  - Soporte para datos adicionales o extensibles
  - Soporte para rutinas definidas por el usuario
  - Herencia
  - Representación de atributos multievaluados
  - Relaciones (tablas) anidadas

- En función del tipo de datos que utilizan y el tipo de consultas que realizan, podemos clasificarlas en cuatro grupos:
  - Utilizan datos simples, realizan consultas simples (Lenguajes de Programación tradicionales)
  - Utilizan datos simples, realizan consultas complejas. (Modelo relacional, Red yJerárquico)
  - Utilizan datos complejos, realizan consultas simples. (OO, capacidades de consulta limitadas)
  - Utilizan datos complejos, realizan consultas complejas. (Objeto-relacional)

```
CREATE TABLE ciudades(
      codciudad SERIAL,
      nombre VARCHAR(30),
      habitantes INTEGER,
      altitud INTEGER,
CONSTRAINT cp_ciudades PRIMARY KEY(codciudad));
```

```
CREATE TABLE capitales(
      provincia VARCHAR(20))
INHERITS (ciudades);
```

- Características introducidas en SQL3:
  - Nuevos tipos de operaciones como `SIMILAR` que permite comparar cadenas de caracteres utilizando expresiones regulares

```
WHERE NAME SIMILAR TO '(SQL-(86|89|92|99))|
                       (SQL(1|2|3))'
```

  - El tipo booleano se ha ampliado con el valor `UNKNOWN` (valor desconocido, recuerda la lógica trivaluada).
  - Consultas recursivas (recursión lineal).
  - Mejora de la seguridad: roles.
  - Disparadores como reglas activas: inserción, borrado y actualización (antes y después)
  - Facilidades para BBDD distribuidas.
  - Estructuras de control.

Soporte Objeto Relacional de SQL3:

- SQL3 soporta:
  - Objetos binarios grandes (BLOB) dentro del propio SGBD
  - Tipo fila:

```
...
DIRECCION ROW(
    CALLE VARCHAR(50),
    CIUDAD VARCHAR(30),
    PAIS VARCHAR(5))
...
```

  - TADs: CREATE TYPE, incluyendo herencia y sobrecarga de operadores
  - Constructores de colecciones

BASES DE DATOS (IG18 Semipresencial)
Otros Modelos de Bases de Datos. El modelo orientado a objetos y objeto-relacional

Lledó Museros / Ismael Sanz
museros@icc.uji.es / isanz@icc.uji.es