Dpto. Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur

Elementos de Bases de Datos

Segundo Cuatrimestre 2015

Clase 23:

Bases de Datos + Orientación a Objetos

Mg. María Mercedes Vitturini
[mvitturi@uns.edu.ar]

Primeras Aplicaciones de BD's

Requerimientos de las aplicaciones de BD “tradicionales”:

- Uniformidad en las estructuras de datos.
- *“Orientación a registros”* datos formados por un conjunto definido de atributos de longitud fija.
- Datos pequeños, acotados a un determinado número de bytes.
- Atributos atómicos.
- Transacciones cortas.
- Esquemas de datos estáticos.

Nuevas aplicaciones

- Bases de datos multimediales: conteniendo imágenes, audio y video.
- Bases de datos con hipertexto.
- Bases de datos para redes sociales.
- Diseño asistido por computadora (CAD: Computer Aided Design).
- Ingeniería de software asistida por computadora (CASE: Computer Aided Software Engineering).
- Entre otras …




Aplicaciones: nuevos requerimientos

Estas nuevas aplicaciones requieren:

- Manipular objetos complejos: los objetos de pueden tener una estructura interna compleja arbitraria.
- Meta-conocimiento: o conocimiento sobre el conocimiento. Cierta información no es dato sino son reglas generales de la aplicación.
- Transacciones largas: herramientas CAD y CASE implican interacción entre el usuario y los datos. Por lo tanto, las transacciones tienen mayor duración por ser interactivas.
- Manipular información sobre comportamiento: distintos objetos pueden responder de manera diferente a la misma orden.

Fortalezas del Modelo Relacional

- ☺ Fuerte basamento teórico.
- ☺ Modelo de datos base simple y confiable.
- ☺ Apropiado para procesamiento de transacciones cortas on-line.
- ☺ Soporte para independencia de datos (modelos conceptual, lógico y físico).
- ☺ Tiene una respuesta adecuada en problemas con:
  - tipos de datos simples (fechas, strings),
  - gran número de instancias (estudiantes, empleados, transacciones comerciales),
  - relaciones bien definidas entre datos,
  - transacciones reducidas,
  - consultas simples

Falencias de las BD Relacionales

☹ La Normalización obliga a crear 'objetos' que no se ajustan a las del mundo real.

☹ Sobrecarga semántica.

- Todos los datos se almacenan como tablas.
- Datos organizados en filas y columnas y no todos los conceptos del mundo real pueden organizarse de esta forma.

☹ Incompatibilidad entre los tipos de datos del SMBD y los tipos de datos los lenguajes propietarios.

El Paradigma Orientado a Objetos

- Fundamentos del paradigma *orientado a objetos:*
  - *Sistema de tipos* poderoso.
    - Tipos atómicos, y
    - Constructores de tipo
  - *Clases* como tipos de datos, contendoras de *objetos*.
  - Las clases incluyen *métodos* o procedimientos aplicables sobre sus objetos.
  - *Herencia* que permite ordenar las clases en una jerarquía.

Modelado Orientado a Objetos

- Las propuestas de modelado orientado a objetos proveen un entorno adecuado para el desarrollo de sistemas complejos.
- Un modelado OO se basa en objetos que encapsulan datos y comportamiento.
- Las herramientas de modelado (por ejemplo UML) capturan el modelado de datos y su comportamiento. Modelar ambos aspectos ofrece beneficios a nivel de herencia y reuso de la implementación.

Bases de Datos + Orientación a Objetos

- **Propuestas** para gestión de datos persistentes orientados a objetos:
  1. Extender el Lenguaje de Programación Orientado a Objetos con capacidades para administrar datos persistentes.
  2. Considerar extender el modelo relacional y definir *sistemas de gestión de bases de datos relacionales-orientados a objetos*.
  3. Nuevas propuestas de Mapeo Objeto-Relacional (ORM's).

Objetivo: proveer soporte para datos persistentes compatibles con el sistema de tipos del lenguaje orientado a objetos.

ORDBMS´s

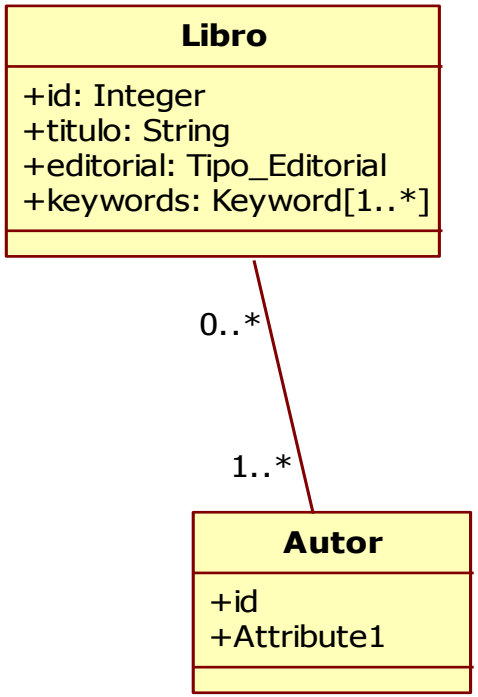
Considera extender el modelo relacional según el paradigma OO:

OBJECT RELATIONAL DATABASE MANAGEMENT SYSTEM (ORDBMS'S)

- Soportan SQL basado en SQL:1999 y SQL:2003. Se extiende SQL para representar:
  - Tipos de datos complejos y colecciones (datos multivalorados).
  - Herencia.
  - Referencia a objetos, referenciar desde un objeto a otros objetos.
- Los datos siguen estando almacenados en tablas respetando el modelo relacional.
- Extienden el poder de modelado.

Ejemplo

- Consideremos la *clase libro*. Cada libro puede pensarse:
  - Su título,
  - Lista de autores,
  - Editorial,
  - Conjunto de palabras claves.



TÍTULO	AUTORES	EDITORIAL (NOMBRE, UBICACIÓN)	PALABRAS CLAVES
Compiladores	[Smith, Jones]	(McGraw-Hill, Nueva York)	{parsing, análisis}
Redes	[Frick, Jones]	(Oxford, Londres)	{Internet, Web}

Alternativa Normalizada en una BD Relacional

Título	Autores	Posición
Compiladores	Smith	1
Compiladores	Jones	2
Redes	Jones	1
Redes	Frick	2

Título	PlbrsClv
Compiladores	Parsing
Compiladores	Análisis
Redes	Internet
Redes	Web

Título	Ed_Nombre	Ed_Ciudad
Compiladores	McGraw-Hill	New York
Redes	Oxford	London

SQL Extendido

- SQL:1999 extiende el sistema de tipos para permitir:
  - **Tipos estructurados:** registros, conjuntos y arreglos.
  - **Herencia**
  - Definir **métodos** asociados a los tipos
  - Se extiende el LMD y LDD para adecuar las definiciones y consultas

SQL: Tipos Estructurados

Tipos y tablas

```
CREATE TYPE TipoApyNom AS
(nombre       varchar(20),
 apellido     varchar(20))

CREATE TYPE TipoDireccion AS
(domicilio      varchar(20),
 ciudad         varchar(20),
 codigo_postal varchar(10));

CREATE TABLE clientes(
nombre          TipoApyNom,
cli_direccion TipoDirección,
fecha_nac       DATE);
```

Otra alternativa

```
CREATE TYPE TipoPersona
(nombre       ApyNom,
 domicilio    Direccion,
 fecha_nac    DATE
)

CREATE TABLE Personas of
TipoPersona;
```

Se usa notación de punto para referenciar los componentes de un tipo complejo: *nombre.apellido*

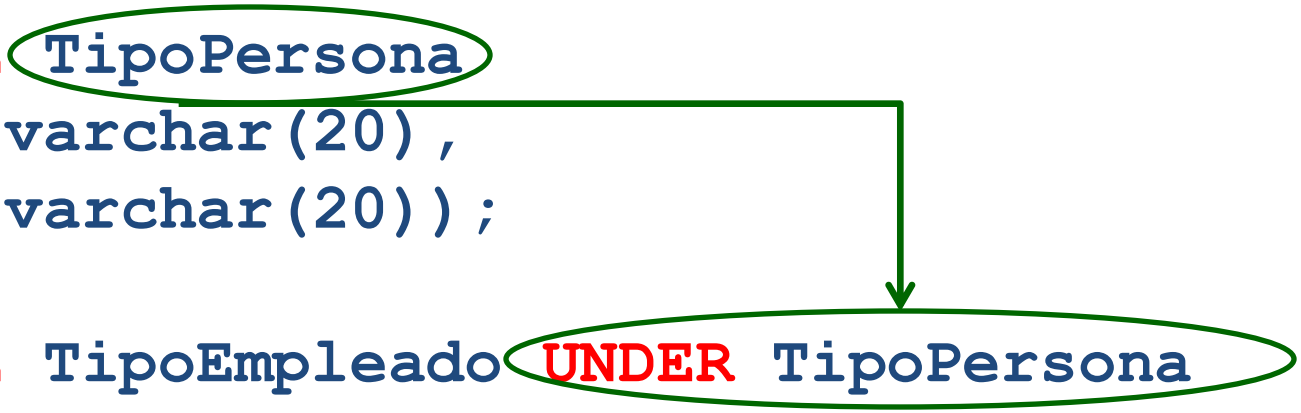
Herencia de Tipos y Tablas

```
CREATE TYPE TipoPersona
(nombre    varchar(20),
 direccion varchar(20));

CREATE TYPE TipoEmpleado UNDER TipoPersona
(salario        integer,
 departamento   varchar(20));

CREATE TABLE personas OF TipoPersona;

CREATE TABLE empleados OF TipoEmpleado
  UNDER personas;
```



Atributos multivaluados: Arreglos y conjuntos

```
CREATE TYPE TipoEditorial AS
(nombre           varchar(20),
 domicilio        varchar(20))

CREATE TYPE TipoLibro AS
(titulo                  varchar(20),
 arreglo_autores         varchar(20) ARRAY [10],
 fecha_edicion           date,
 editorial               TipoEditorial,
 conjunto_kwords         varchar(20) MULTISET )

CREATE TABLE libros OF TipoLibro;
```

Referencia a objetos

```
CREATE TYPE Departamento (
    nombre varchar (20),
    director ref (Persona)scope personas);

CREATE TABLE departamentos of Departmento;
```

- La tabla referenciada debe contar con un atributo identificador, llamado **self-referential attribute**

```
CREATE TABLE personas of Persona
    ref is person_id system generated;
```

LMD

También, se adecua la *sintaxis del LMD* para manejar los nuevos datos.

Ejemplo:

```
INSERT INTO libros
VALUES('Database System Concepts',
 array['Silberchatz', 'Korth'],
 new ('MacGraw-Hill','New York'),
 multiset['database','SQL'];

SELECT titulo
FROM libros
WHERE 'database' IN (UNNEST(conjunto_kwords));
```

Object Relatinal DBMS -

- Básicamente se resuelven como extensiones de los sistemas de bases de datos relacionales existentes.
- Minimizan los cambios en el sistema de almacenamiento (almacenamiento relacional, índices) usados por los DBMS relacionales.
- Soportan ciertos tipos de datos complejos.
- Preservan los fundamentos relacionales, en particular el acceso declarativo a los datos.
- Extienden el poder de los lenguajes de consultas relacionales.
- Favorecen un modelado más intuitivo para aplicaciones con tipos de datos complejos.

Lenguajes de Programación con manejo de Datos Persistentes

- Un *lenguaje de programación persistente* (LPP) es un lenguaje de programación extendido con constructores para manipular directamente datos persistentes.
- Existen importantes diferencias entre un lenguaje de programación persistente y un lenguaje con SQL embebido.
- Ejemplos de LPP's: C++, Java, Smalltalk.

LPP - Ventajas

- El *lenguaje de manipulación de datos está plenamente integrado con el lenguaje huésped* y comparten el mismo sistema de tipos.
- El programador manipula *datos persistentes* trabajando en *alto nivel* (despreocupándose de traer datos a memoria o grabarlos al disco).
- Dirigidos *a aplicaciones que requieren de alto rendimiento*.

LPP - Desventajas

- Puesto que el lenguaje de programación es poderoso, *es relativamente fácil introducir errores que dañen la base de datos* (problemas de integridad de dato).
- Es *difícil realizar optimización* en nivel alto, tal como reducir las operaciones de acceso al disco.
- Los lenguajes de programación persistentes *no soportan consultas declarativas*.
- *No suelen disponer de grandes variaciones de consultas*.

DBMS – OODBMS – RODBMS

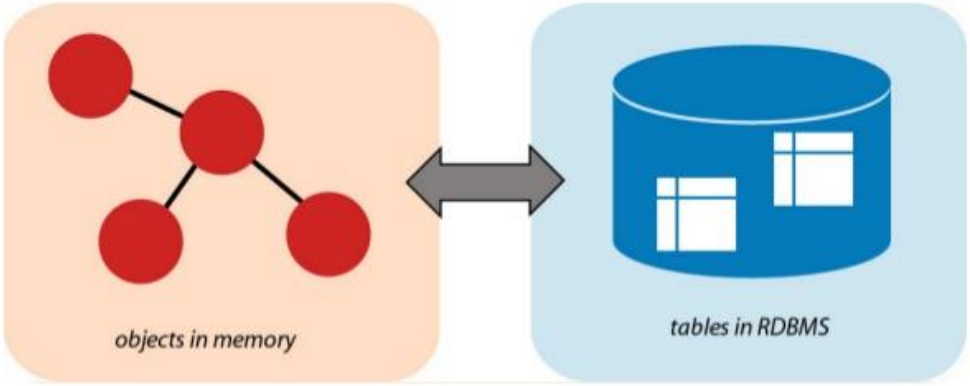
Criterio	RDBMS	OODBMS	RODBMs
Estándar	SQL2	ODMG 2.0	SQL3
Soporte de las características OO	No	Si	Nuevos tipos de datos. Herencia de datos
Tipos de Datos	Básicos	Amplia variedad de tipos de datos y con relaciones complejas	TDA y relaciones
Ventajas	SQL y optimización de consulta. Performance. Habilidad para la integridad de datos.	Aplicaciones complejas. Reusabilidad del código. Coherencia en el sistema de tipos	Datos complejos y consultas sobre ellos. Representación del modelo físico.

DBMS – OODBMS – RODBMS Comparación

Criterio	RDBMS	OODBMS	RODBMs
Desventajas	Limitación de uso para aplicaciones complejas	Baja performance. Dificultades de uso a gran escala.	Baja performance
Mercado	Mercado de consumo y proveedores amplio.	Mercado reducido.	Los proveedores de DBMS tradicionales expanden sus productos hacia RODBMS

Object-Relational Mapping (ORM)

- Proveen la tercer variante para integrar lenguajes de programación OO y los sistemas de bases de datos relacionales.
- Los sistemas ORM ofrecen un nivel de abstracción por encima de las bases de datos relacionales. El programador de define por única vez el mapeo entre el modelo de datos OO y el modelo de datos relacional.



Object-Relational Mapping Frameworks

Persistencia de Objetos usando Bases de Datos Relacionales

- Framework completos ORM: Hibernate, JDO, Java Persistence API, TopLink
- Ocultan el acceso a los datos relacionales desde las aplicaciones orientadas a objetos creando un nivel de persistencia transparente.
- Proveen un esquema de mapeo declarativo que vincula a las clases de dominio que requieren persistencia con las tablas relacionales.

Object-Relational Mapping Frameworks

Persistencia de Objetos usando Bases de Datos Relacionales

- De manera análoga lo hacen para manejar transacciones, seguridad, etc. que están ocultas desde la aplicación.
- Las clases a las que ORM da persistencia desconocen que son persistentes.
- El entorno ofrece lenguajes de consulta que permiten diseñar las consultas directamente sobre el modelo orientado a objetos.

Conceptos estudiados

- Aplicaciones orientadas a registros. Características.
- Nuevos tipos de problemas. Requisitos.
- Modelo Relacional vs. Modelo Orientado a Objeto
- Propuestas en Bases de Datos Orientadas a Objetos
  - Extensiones al modelo relacional (RODBMS)
  - Lenguajes de Programación Orientados a Objetos Persistentes (OODBMS)
- ORDBMS
  - Extensiones sobre el RDBMS.
  - Características
- OODBMS
  - Caracterísiticas
- ORM

Temas de la Clase de Hoy

- Bases de Datos Relacionales OO.
  - Tipos complejos.
- Bases de Datos OO.
  - Clases, Herencia.
  - Lenguajes de Programación Persistente.

- **Bibliografía**
  - “Database System Concepts” – A. Silberschatz. Capítulo 9 (ed. 2005), Capítulo 22 (ed. 2010) .