ACCESO A DATOS

CAPÍTULO 3

Bases de datos objeto-relacionales y orientadas a objetos

José Eduardo Córcoles
Francisco Montero

Ra-Ma®

ÍNDICE DE CONTENIDOS

OBJETIVOS

a) Identificar las ventajas e inconvenientes de las bases de datos que almacenan objetos.
b) Establecer y cerrar conexiones.
c) Gestionar la persistencia de objetos simples.
d) Gestionar la persistencia de objetos estructurados.
e) Desarrollar aplicaciones que realizan consultas.
f) Modificar los objetos almacenados.
g) Gestionar las transacciones.
h) Probar y documentar las aplicaciones desarrolladas.

3.1 CARACTERÍSTICAS DE LAS BASES DE DATOS ORIENTADAS A OBJETOS

- Los **sistemas gestores de bases de datos relacionales** (SGBDR, en inglés RDBMS), son las aplicaciones software que gestionan los datos según el modelo relacional (no confundir éste con el software que lo implementa y gestiona).
- Pese a su eficacia, el modelo relacional se vio abocado en la década de los 90 a una actualización, a una mejora para adaptarse a los nuevos tiempos.
- Los **SGBD-OO permiten almacenar objetos** (persistencia) y recuperarlos según el modelo orientado a objetos, lo cual simplifica enormemente el desarrollo de software.

3.1.1 ODMG – OBJECT DATA MANAGEMENT GROUP

- Usar un SGBDR para almacenar objetos (persistencia) incrementa significativamente la complejidad de un desarrollo software ya que obliga a una conversión de objetos a tablas relacionales.
- Las características que todo SGBD-OO debe implementar son:

Almacén de objetos complejos	Identidad de los objetos
Encapsulación	Tipos o clases
Herencia	Ligadura dinámica
Completitud de cálculos	El conjunto de tipos de datos debe ser extensible
Persistencia de datos	Manejar gran cantidad de datos
Concurrencia	Recuperación
Método de consulta sencillo	

3.1.2 EL MODELO DE DATOS ODMG

- El ODMG – Object Data Management Group es un consorcio industrial de vendedores de SGBD-OO que después de su creación se afilió al OMG (Object Management Group).
- Entre muchas otras especificaciones el estándar ODMG define el Modelo de Objetos que debe ser soportado por el SGBD-OO.
- El lenguaje de base de datos es especificado mediante:
  - un *Lenguaje de Definición de Objetos* (ODL) que se corresponde con el DDL de los SGBD relacionales,
  - un *Lenguaje de Manipulación de Objetos* (OML) y
  - un *Lenguaje de Consulta* (OQL), que equivale al archiconocido SQL de ANSI-ISO.

3.1.3 ODL (LENGUAJE DE DEFINICIÓN DE OBJETOS)

- ODL (Lenguaje de Definición de Objetos) es un lenguaje para definir la especificación de los tipos de objetos en sistemas compatibles con ODMG.
- Una característica importante que debe cumplir un ODL es ofrecer al diseñador de bases de datos un sistema de tipos semejantes a los de programación OO.

EJemplos:

```
interface Libro {
/* Definición de atributos
attribute string título;
attribute integer año;
attribute integer paginas;
attribute enum PosiblesEncuadernaciones (Dura,Bolsillo) tipo;
/* Definición de relaciones */
relationship Set<Autor> escrito_por inverse Autor::escribe;
}
```

```
interface Autor {
/* Definición de atributos */
attribute string apellidos;
attribute string nombre;
attribute string nacionalidad;
attribute short edad;
/* Definición de relaciones */
relationship Set<Libro> escribe inverse Libros::escrito_por;
}
```

Actividad 3.1

3.1.4 OML (LENGUAJE DE MANIPULACIÓN DE OBJETOS)

- ODMG no define ningún lenguaje de manipulación de objetos (OML), deja descansar esta tarea en los propios lenguaje de programación.
- Para mostrar cómo se pueden modificar objetos directamente con un lenguaje de programación, en la sección 3.3 trabajaremos con Java para realizar modificaciones en una base de datos gestionada con *Matisse*.

Actividad 3.2

3.1.5 OQL (LENGUAJE DE CONSULTAS DE OBJETOS)

- **OQL** -Lenguaje de Consultas de Objetos- es un lenguaje declarativo del tipo de SQL que permite realizar consultas sobre bases de datos orientadas a objetos, incluyendo primitivas de alto nivel para conjuntos de objetos y estructuras.

OQL_1	OQL_2	OQL_3	OQL_4
select a.titulo **from** articulo a **where** a.paginas>30;	**select** distinct a.nombre, a.apellidos **from** autor a **where** apellidos like 'Mura%‘;	**select** l.titulo, a.nombre, a.apellidos **from** autor a, Libro l **where** a.escribe=l.OID and apellidos like 'Mura%‘;	**select** l.titulo, a.dameNombreyApellidos() **from** autor a, Libro l **where** a.escribe=l.OID;

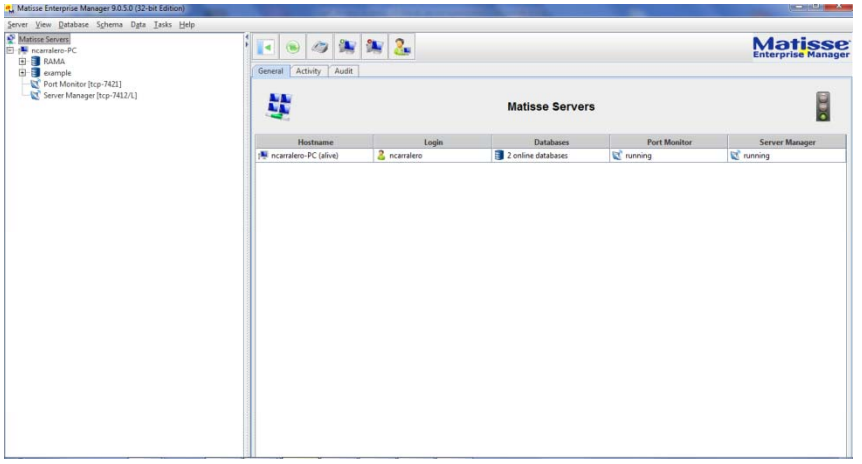
Actividad 3.3

3.2 SISTEMAS GESTORES DE BASES DE DATOS ORIENTADAS A OBJETOS.

- Matisse es una alternativa que respeta en gran medida el estándar ODMG.
- Matisse es una buena referencia para probar los diferentes lenguajes de definición, manipulación y consulta de objetos.
- Matisse tiene ventajas para la gestión propias de la OO. Algunas de ellas son:
  - Técnicas para fragmentar objetos grandes.
  - Una ubicación optimizada de los objetos.
  - Un mecanismo automático de duplicación.
  - Un mecanismo de versiones de objetos incorporado.
  - Soporte para una arquitectura cliente-servidor.
  - Optimización de acceso a objetos relacionados.

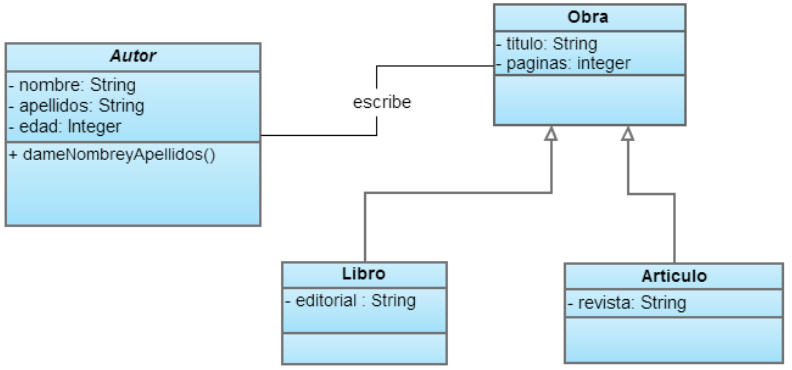
3.2.1 INSTALACIÓN DE MATISSE

- *Matisse* puede ser descargado de www.fresher.com
- Todo lo que ofrece *Matisse* está bien documentado.
- La ventana principal tiene dos partes:
  - El árbol de la izquierda muestra las bases de datos creadas.
  - La parte de la derecha sirve para ejecutar los lenguajes ODL y OQL.

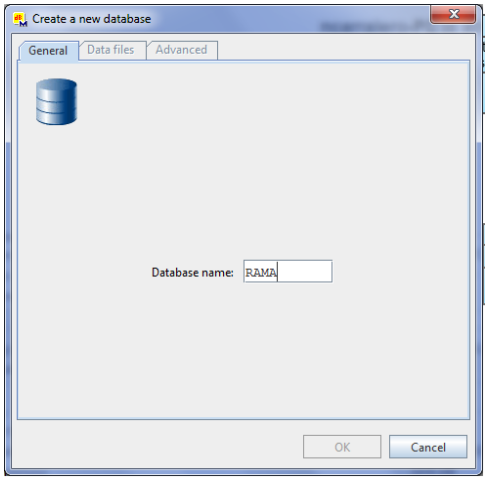


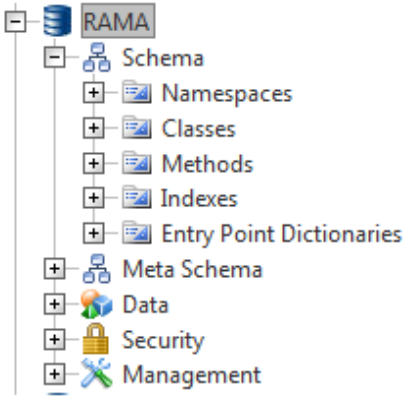
3.2.2 CREANDO UN ESQUEMA CON MATISSE

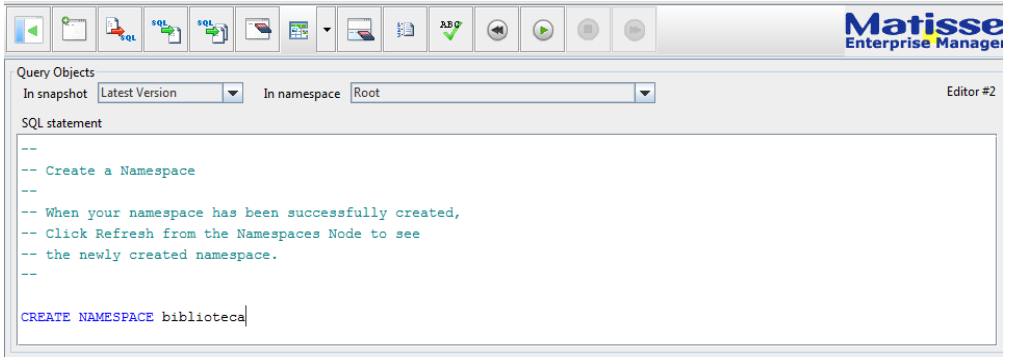
- Considerar el siguiente modelo:
  - Una clase Autor.
  - Una clase Obra.
  - Una clase Libro.
  - Una clase Artículo.

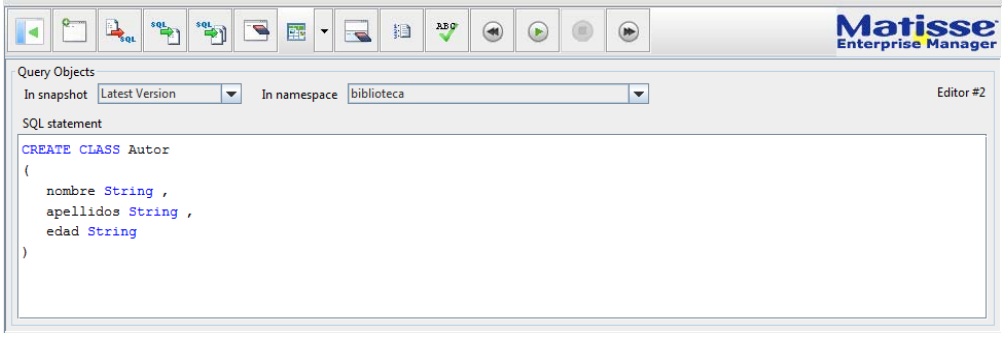


3.2.2 CREANDO UN ESQUEMA CON MATISSE









3.2.2 CREANDO UN ESQUEMA CON MATISSE

```
CREATE CLASS Obra
(
   titulo string ,
   paginas Integer
)

CREATE CLASS Libro
  INHERIT Obra
(
   editorial String
)

CREATE CLASS Articulo
  INHERIT Obra
(
   revista String
)
```

```
ALTER CLASS Autor
  ADD RELATIONSHIP escribe
     RELATIONSHIP SET( Obra)

     INVERSE Obra.escrito_por;

ALTER CLASS Obra
  ADD RELATIONSHIP escrito_por
     RELATIONSHIP SET( Autor)

     INVERSE Autor.escribe;
```

```
CREATE  METHOD
dameNombreyApellidos ()
RETURNS String
FOR Autor
--
-- Describe tu método aquí
--
BEGIN
  return CONCAT (nombre, apellidos);
END;
```

Actividad 3.4

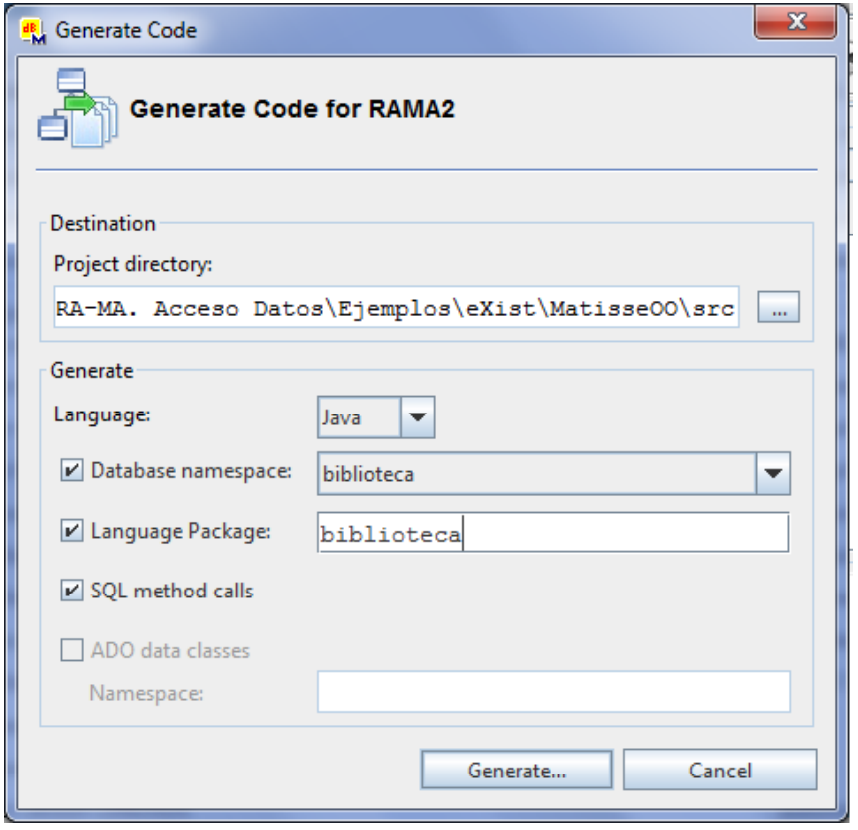
3.3 INTERFAZ DE PROGRAMACIÓN DE APLICACIONES DE LA BASE DE DATOS

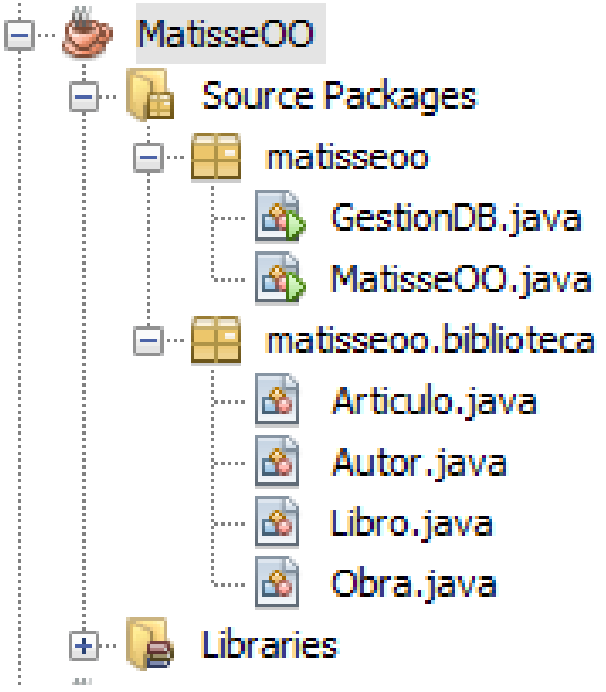
- Una vez el esquema de la base de datos se ha creado, el siguiente paso es preparar el sistema para que pueda ser accedido desde código Java.
- ODMG no define un lenguaje de manipulación de objetos (OML) y *Matisse* tampoco.
- La única manera de añadir, eliminar modificar y consultar objetos en el esquema creado en la sección anterior es usando código fuente.
- En esta sección se tratará esta forma de acceso mediante código fuente Java.

3.3.1 PREPARANDO EL CÓDIGO JAVA

- Para poder entender el proceso para acceder a los objetos almacenados en las bases de datos OO hay que tener siempre presente que lo que se busca es tener la sensación de *trabajar solo con objetos.*
- El mecanismo por el cual el contenido de un objeto se almacena en la base de datos es totalmente transparente al programador.

3.3.1 PREPARANDO EL CÓDIGO JAVA





3.3.2 AÑADIENDO OBJETOS

- Una vez creadas las clases la manera de añadir objetos a la base de datos es sencilla, solo hay que crearse en Java objetos y luego llamar para almacenarlos.

- El objeto necesario para la persistencia es:
  - *MtDatabase*
    - *MtDatabase()*
    - *Open()*
    - *startTransaction()*
    - *Commit()*
    - *Close()*
  - *MtException*

3.3.2 AÑADIENDO OBJETOS

```
public static void creaObjetos(String hostname, String dbname){
    try {
    //Abre la base de datos con el Hostname (localhost), dbname (RAMA) y es namespace  "biblioteca".
      MtDatabase db = new MtDatabase(hostname, dbname, new MtPackageObjectFactory("","biblioteca"));
      //Abre la base de datos y empieza una transacción.
       db.open();
       db.startTransaction();

       // Crea un objeto Autor
       Autor a1 = new Autor(db);
       a1.setNombre("Haruki");
      a1.setApellidos("Murakami");
      a1.setEdad("53");

       // Crea un objeto Libro
       Libro l1 = new Libro(db);
       l1.setTitulo("Baila Baila Baila");
       l1.setEditorial("TusQuests");
       l1.setPaginas(512);

       // Crea otro objeto Autor
      Libro l2 = new Libro(db);
      l2.setTitulo("Tokio Blues");
      l2.setEditorial("TusQuests");
       l2.setPaginas(498);

       //Crea un array de Obras para
      //guardar los libros y hacer las relaciones
         Obra o1[]= new Obra[2];
          o1[0]=l1;
          o1[1]=l2;

      //Guarda las relaciones del autor con los libros que ha escrito.
        a1.setEscribe(o1);
       //Ejecuta un commit para materializar las peticiones.
        db.commit();
       //Cierra la base de datos.
         db.close();
     } catch (MtException mte) {
        System.out.println("MtException : "  + mte.getMessage());
       }
     }
```

Actividad 3.5

3.3.3 ELIMINANDO OBJETOS

- El borrado de objetos se hace utilizando el método deepRemove() definido en todos los objetos de Matisse.

3.3.3 ELIMINANDO OBJETOS (I)

```
public static void borraObjetos(String hostname, String dbname) {
    try {
     MtDatabase db = new MtDatabase(hostname, dbname, new MtPackageObjectFactory("","biblioteca"));
      db.open();
     db.startTransaction();
     // Lista todos los objetos Obra con el método getInstanceNumber
      System.out.println("\n" +  Obra.getInstanceNumber(db) “+ “Obra(s) en la DB.");
      //Crea un Iterador (propio de Java)
      MtObjectIterator<Obra> iter = Obra.<Obra>instanceIterator(db);
       System.out.println("Borra dos Obras");
       while (iter.hasNext()) {
          Obra[] obras = iter.next(2);
           System.out.println("Borrando " +  obras.length +  " Obra(s)...");
            for (int i=0; i < obras.length; i  ) {
           //borra definitivamente el objeto
           obras[i].deepRemove();
        }
```

3.3.3 ELIMINANDO OBJETOS (II)

```
        // (cont.)
         // Solo borra dos y lo deja
                  break;
         }
         iter.close();
         //materializa los cambios y cierra la BD
         db.commit();
         db.close();

         System.out.println("\nHEcho.");
      } catch (MtException mte) {
         System.out.println("MtException : " +  mte.getMessage());
      }
}
```

Actividad 3.6

3.3.4 MODIFICANDO OBJETOS (I)

```
public static void ModificaObjeto(String hostname, String dbname, String nombre, String nuevaEdad)  {
      int nAutores=0;
        try {
           MtDatabase db = new MtDatabase(hostname, dbname, new MtPackageObjectFactory("","biblioteca"));
           db.open();
           db.startTransaction();
           // Lista cuántos objetos Obra con el método getInstanceNumber
           System.out.println("\n“+   Autor.getInstanceNumber(db)+" “Autores en la DB.");
          nAutores=(int)Autor.getInstanceNumber(db);
          //Crea un Iterador (propio de Java)
           MtObjectIterator<Autor> iter = Autor.<Autor>instanceIterator(db);
           System.out.println("recorro el iterador de uno en uno y cambio cuando encuentro 'nombre'");
           while (iter.hasNext()) {
                    Autor[] autores = iter.next(nAutores);
                    for (int i=0; i < autores.length; i  ) {
                       //Busca una autor con nombre 'nombre'
                       if (autores[i].getNombre().compareTo(nombre)==0) {  autores[i].setEdad(nuevaEdad);   }
                    }
           }
// (continuación…)
```

3.3.4 MODIFICANDO OBJETOS (II)

```
        //(cont.)
         iter.close();
        //materializa los cambios y cierra la BD
         db.commit();
         db.close();

         System.out.println("\nHecho.");
      } catch (MtException mte) {
         System.out.println("MtException : "  + mte.getMessage());
      }
}
```

Actividad 3.7

3.3.5 CONSULTANDO OBJETOS CON OQL

- La última de las operaciones básicas sobre una base de datos es la ejecución de consultas.
- Las consultas OQL pueden ejecutarse directamente sobre el propio entorno de *Matisse* sin embargo esta sección se centra en la utilización de Java .

3.3.5 CONSULTANDO OBJETOS CON OQL (I)

```
public static void ejecturaOQL(String hostname, String dbname)    {
    MtDatabase dbcon = new MtDatabase(hostname, dbname);
    //Abre una conexión a la base de datos
    dbcon.open();
    try {
    //Se hace una conexión JDBC
    Connection jdbcon = dbcon.getJDBCConnection();
    // Crea una instancia de Statement
    Statement stmt = dbcon.createStatement();
    // Asigna una consulta OQL. Esta consulta lo que hace es utilizar REF() para obtener el objeto
    //directamente en vez de obtener valores concretos (que también podría ser).
    String commandText = "SELECT REF(a) from biblioteca.Autor a;";
    // Ejecuta la consulta y obtiene un ResultSet
    ResultSet rset = stmt.executeQuery(commandText);
    Autor a1;
     //(continuación)
```

3.3.5 CONSULTANDO OBJETOS CON OQL (II)

```
    // Recorre el rset uno a uno
    while (rset.next()) {
    // Obtiene el objeto Autor
    a1 = (Autor)rset.getObject(1);
    // Imprime los atributos de cada objeto en forma de tabla.
    System.out.println("Autor: "
    String.format("%16s",a1.getNombre())
    String.format("%16s",a1.getApellidos())
    " Páginas: "
    String.format("%16s", a1.getEdad()));
    }
    // Cierra las conexiones.
    rset.close();
    stmt.close();
    dbcon.close();
    }catch (SQLException e) {
       System.out.println("SQLException:  "  + e.getMessage());
    }
}
```

Actividad 3.8 y 3.9

3.4 CARACTERÍSTICAS DE LAS BASES DE DATOS OBJETO-RELACIONALES.

- El término base de datos Objeto-Relacional se usa para describir una base de datos que ha evolucionado desde el modelo relacional hasta una base de datos híbrida, que contiene la tecnología relacional y la orientada a objetos.
- Un tipo de objeto representa una entidad del mundo real y se compone de los siguientes elementos:
  - Su nombre.
  - Sus atributos.
  - Sus métodos.

3.4.1 GESTIÓN DE OBJETOS CON SQL; ANSI SQL 1999

- ¿Cómo se pueden crear objetos en un SGBD-OR?
  - Para crear un objeto con **Oracle** se utiliza CREATE TYPE.

Ejemplo:

CREATE TYPE persona AS OBJECT (
nombre VARCHAR2(30),
telefono VARCHAR2(20)
);

  - En **ANSI** SQL99 sería:

define type persona:
tuple [nombre:string,
telefono:string]

  - Para crear objetos más complejos en **ANSI** SQL99 :

CREATE TYPE estudiante AS OBJECT
(id_estudiante varchar2(9),
datos_personales persona);

  - **Oracle** permite para crear tablas que alberguen objetos:

CREATE TABLE contactos OF persona;
CREATE TABLE admitidos (fecha: date, solo_estudia estudiante);

3.4.1 GESTIÓN DE OBJETOS CON SQL; ANSI SQL 1999.

- Referencias:
  - Las relaciones entre objetos se establecen mediante columnas o atributos de tipo REF.

Ejemplo:

CREATE TABLE Departamento
(NomDept VARCHAR(30),
Jefe REF persona);

- Métodos:
  - los métodos son procedimientos que se declaran en la definición de un tipo de objeto para dotarlo de comportamiento.

```
CREATE TYPE racional AS OBJECT(
numerador INTEGER,
denominador INTEGER,
MEMBER PROCEDURE normaliza, …);
```

```
CREATE TYPE BODY racional AS
  MEMBER PROCEDURE normaliza IS
   g INTEGER;
   BEGIN
    g := gcd(num, den);
    num := num / g;
    den := den / g;
   END normaliza;
END;
```

3.5 CONCLUSIONES Y PROPUESTAS PARA AMPLIAR

- Los SGBDOO siguen sin ser considerados una alternativa fuerte y robusta para dar soporte a los grandes bancos de datos que soportan sistemas relacionales.
- A continuación se sugieren distintas propuestas para ampliar:
  - Hay muchas otros sistemas gestores, además de Matisse.
  - Un ejemplo muy interesante de bases de datos OO es *db4o*.
  - El lenguaje OQL mostrado en *Matisse* es mucho más extenso en sintaxis y semántica que lo mostrado.
  - El acceso a *Matisse* mediante Java para ejecutar consultas OQL mediante JDBC también puede ser ampliada.
  - ¿es posible trabajar directamente con objeto Oracle y objetos Java como permite *Matisse* o es más aconsejable manejar los objetos desde procedimientos almacenados en Oracle?

3.6 RESUMEN DEL CAPÍTULO

- Este capítulo ha mostrado el acceso a SGBD Orientados a Objetos y Objetos Relacionales, siendo los primeros los que han ocupado casi todo el capítulo.
- El capítulo se ha centrado en tres partes bien diferenciadas:
  - Una primera parte de introducción a las bases de datos OO y los estándares que les dan soporte.
  - Una segunda parte con alternativas de acceso y manipulación de datos almacenados en Matisse, un sistema gestor de bases de datos OO. El acceso se ha realizado desde Java, explotando así todas las ventajas de la POO.
  - Una tercera parte que resume los aspectos más destacables de las bases de datos SGBD-OR y sus posibilidades para crear objetos.

ACCESO A DATOS

CAPÍTULO 3

Bases de datos objeto-relacionales y orientadas a objetos

José Eduardo Córcoles
Francisco Montero

Ra-Ma®