

BDOR de Oracle, 2002

Características Objeto-Relacionales
del Sistema de Gestión de Bases de Datos
de Oracle en 2002

y

ADO.NET de Microsoft, 2007

Características Objeto-Relacionales

Introducción

- Partimos de Oracle 8 *Universal Data Server*, cuando Oracle empezó a implementar características object-oriented (OO) en sus SGBDs.
- Oracle no es un sistema objetual puro, sino un SGBD objeto-relacional.
- Objetivos de Oracle en la OO:
 - Permitir que el usuario pueda modelar objetos a través de tipos.
 - Proveer infraestructura para soportar accesos en la OO.
- El estándar SQL de las BDOR se promulga en 2003

Características y Ventajas de los Objetos en Oracle

Características OO:

- Abstracción
- Encapsulación
- Herencia

Ventajas:

- Reutilización de Objetos
- Uso de Métodos
- Eficiencias
- Modelar Objetos de negocio del mundo-real

Implementación del Tipo Objeto

Creación de Tipos

Similar a la creación de "class" con atributos:

```
CREATE TYPE tipo_direc AS OBJECT
```

```
(calle          varchar2(60),
```

```
ciudad         varchar2(30),
```

```
país           char(2),
```

```
CP             varchar(9));
```

tipo_direc			
Calle	Ciudad	País	CP
varchar2(60)	varchar2(30)	char(2)	varchar(9)

Implementación del Tipo Objeto

Anidando y Embebiendo Objetos

Creamos el tipo persona con el tipo-direc anidado:

```
CREATE TYPE tipo_persona AS OBJECT
    (nombre          varchar2(25),
     dirección      tipo_direc);
```

Creamos el tipo estudiante con el tipo persona anidado:

```
CREATE TYPE tipo_estudiante AS OBJECT
    (id_estudiante  varchar2(9),
     persona        tipo_persona);
```

Implementación del Tipo Objeto

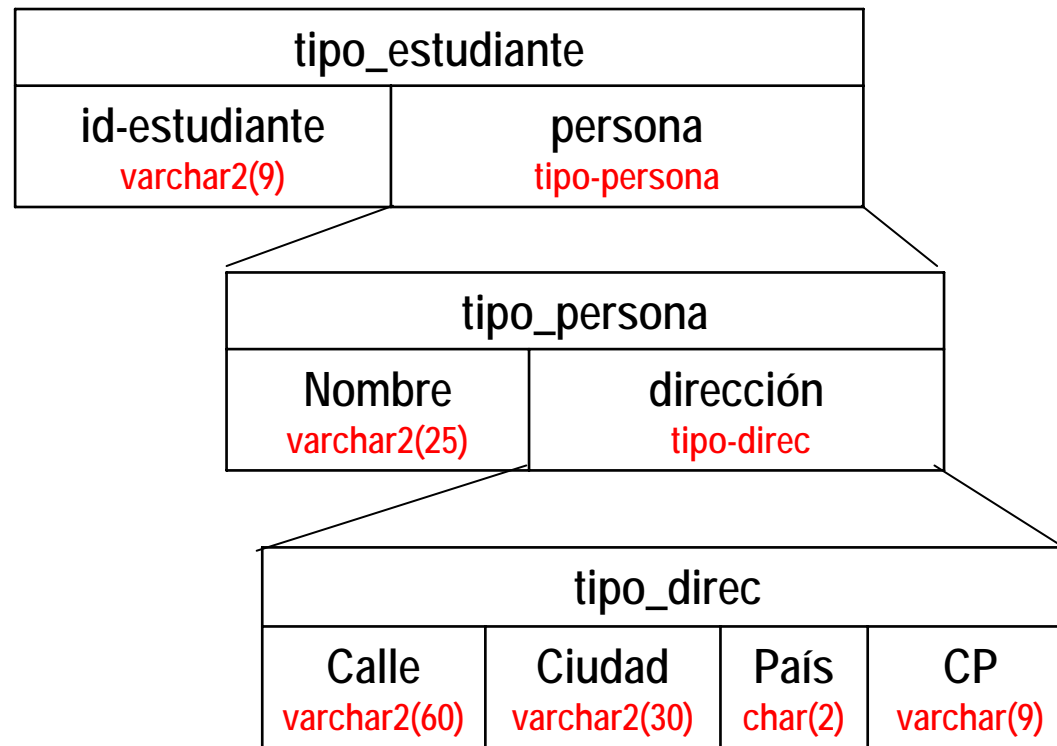


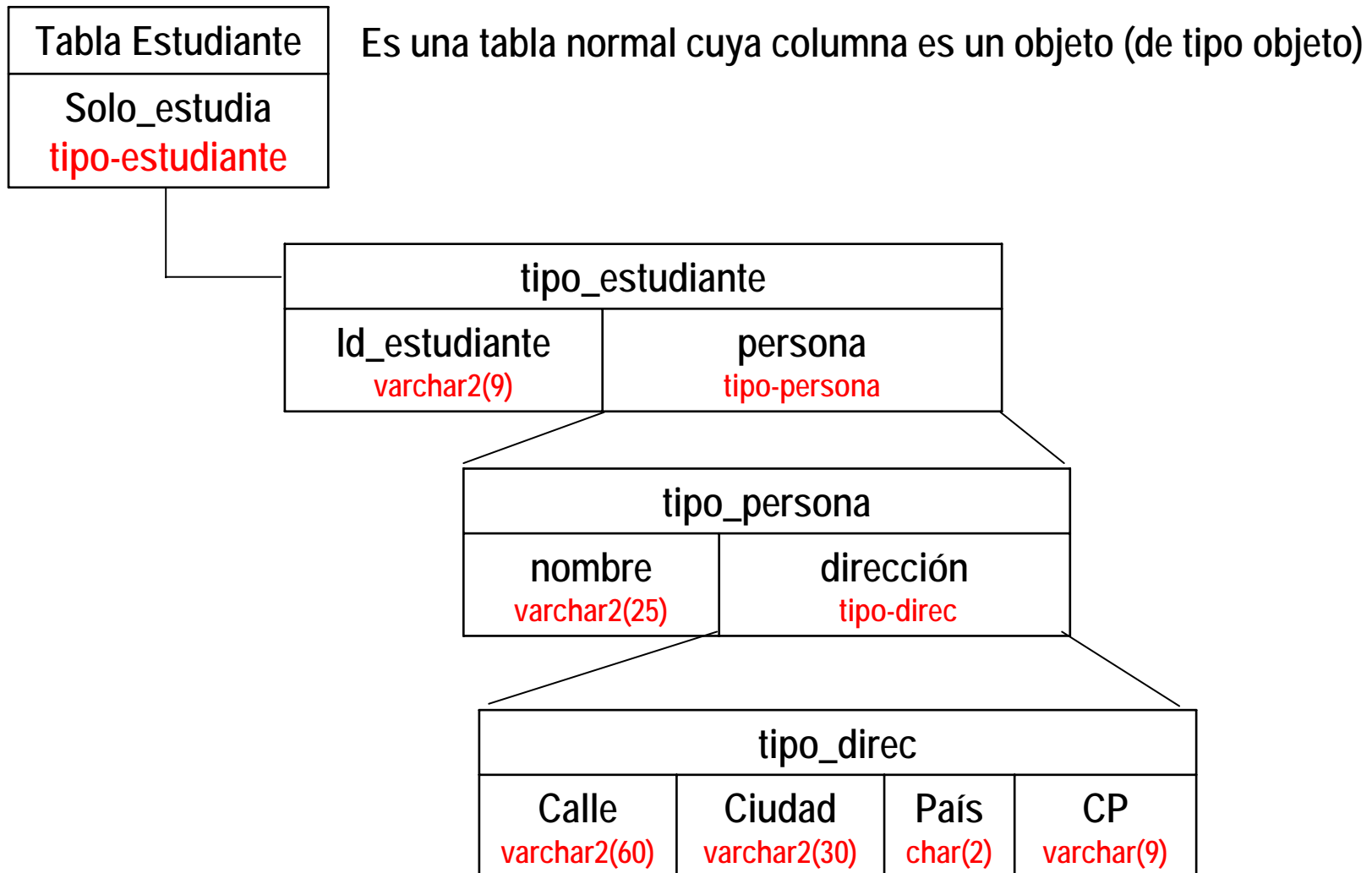
Tabla normal que usa el Tipo Objeto

Creación de una tabla normal cuya columna es de tipo Objeto

Una vez definido el tipo objeto *tipo_estudiante*, ya se puede utilizar al crear una tabla como la siguiente:

```
CREATE TABLE ESTUDIANTE  
    (solo_estudia tipo_estudiante);
```

Implementación del Tipo Objeto



Acceso a la Tabla que usa el Tipo Objeto

Se accede a los datos con la siguiente consulta:

```
SELECT e.solo_estudia. id-estudiante ID, e.solo_estudia.persona.nombre  
NOMBRE, e.solo_estudia.persona.dirección.calle CALLE
```

```
FROM Estudiante e
```

```
WHERE e. solo_estudia. id-estudiante = 100
```

```
Resultado:  ID          NOMBRE          CALLE  
-----  
          100      Juan Pérez. estudiante  Gran Vía, 25
```

Actualizaciones en la Tabla normal que usa el Tipo Objeto

La actualización y borrado es similar a como se hace en el modelo relacional:

UPDATE Estudiante e

SET e.solo_estudia.persona.nombre = 'Juan Sánchez'

WHERE e. solo_estudia. id-estudiante = 100; **Se actualiza una fila**

DELETE FROM Estudiante e

WHERE e. solo_estudia. id-estudiante = 100; **Se borra una fila**

Implementación de Métodos

Sintaxis para implementar un método de un Tipo Objeto:

```
CREATE or REPLACE TYPE tipo_nuevapersona AS object  
  (nombre      varchar2(25),  
   apellido    varchar2(25),  
   fecha_nac   date,  
   member function EDAD (Fecha_Nac in DATE) return NUMBER;
```

member function define la llamada a la función. Ahora definimos la función EDAD con la siguiente sintaxis que define el método en sí mismo. La sintaxis es como sigue:

```
CREATE or REPLACE TYPE BODY tipo_nuevapersona AS  
  member function EDAD (Fecha_Nac in DATE) return NUMBER IS  
    begin  
      return ROUND(SysDate – Fecha_Nac);  
    end;  
  
end;
```

Tabla que es un Objeto

Sintaxis SQL:

CREATE TABLE nombre-tabla-objeto **OF** tipo_objeto-
previamente-definido;

Ejemplo:

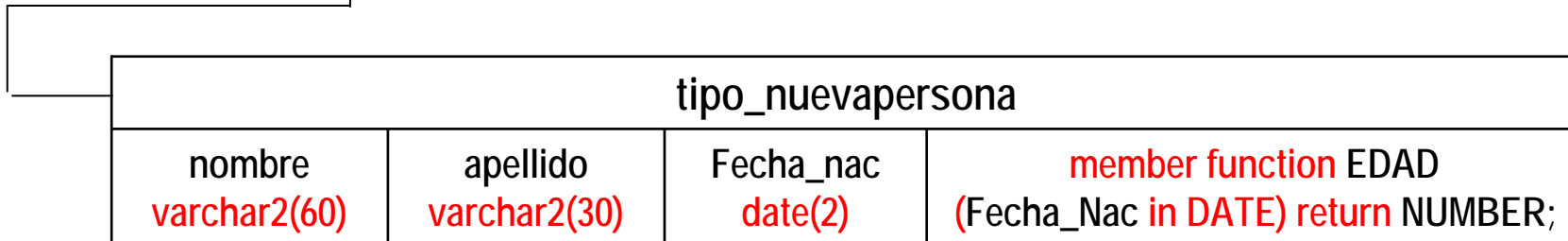
CREATE TABLE PERSONA_NUEVA **OF** tipo_nuevapersona;

Una tabla objeto almacena objetos con identidad propia (**OID**)



3º) **CREATE TABLE** PERSONA_NUEVA **OF** tipo_nuevapersona;

4º) **INSERT INTO** PERSONA_NUEVA **VALUES**
 (tipo_nuevapersona ('Juan', 'Sanchez', TO_DATE ('03-FEB-1970',
 'DD-MM-YYYY')));



1º) **CREATE or REPLACE TYPE** tipo_nuevapersona **AS object**
 (nombre varchar2(25),
 apellido varchar2(25),
 fecha_nac date,
 member function EDAD (Fecha_Nac in DATE) return NUMBER;

2º) **CREATE or REPLACE TYPE BODY** tipo_nuevapersona **AS**
member function EDAD (Fecha_Nac in DATE) return NUMBER
IS
 begin
 return ROUND(SysDate - Fecha_Nac);
 end;
end;

Implementación de Métodos

Ya podemos utilizar la función **EDAD**. Si queremos verificar el método, antes hay que crear una tabla que albergue al menos a una persona del tipo **tipo_nuevapersona**:

```
CREATE TABLE PERSONA_NUEVA OF tipo_nuevapersona; es una tabla de objetos
```

```
INSERT INTO PERSONA_NUEVA VALUES  
(tipo_nuevapersona ('Juan', 'Sanchez', TO_DATE ('03-FEB-1970', 'DD-MM-YYYY')));
```

Podemos verificar la función **EDAD** de la siguiente manera:

```
SELECT P. tipo_nuevapersona.EDAD (P. tipo_nuevapersona.Fecha_Nac)  
FROM PERSONA_NUEVA P;
```

Resultado dado en días:

```
P. tipo_nuevapersona.EDAD (P. tipo_nuevapersona. Fecha_Nac )
```

12005

Referencias

Cada fila objeto tiene un identificador único, llamado **OID**.

OID permite que otros objetos referencien un objeto fila existente en la BDOR.

Para referenciar un **OID** se puede usar la función **REF**:

```
CREATE TABLE NUEVODEPARTAMENTO
```

```
(NomDept VARCHAR(30),
```

```
PersonaEn REF tipo_nuevapersona);
```

La tabla NUEVODEPARTAMENTO tiene una referencia al objeto **tipo_nuevapersona**, que no es el valor de un dato real.

Referencias

Tabla NUEVODEPARTAMENTO	
NomDept VARCHAR(30)	PersonaEn tipo_nuevapersona

```
CREATE TABLE NUEVODEPARTAMENTO  
(NomDept VARCHAR(30),  
PersonaEn REF tipo_nuevapersona);
```

tipo_nuevapersona			
nombre varchar2(60)	apellido varchar2(30)	Fecha_nac date	member function EDAD (Fecha_Nac in DATE) return NUMBER;

Para insertar una tupla en
NUEVODEPARTAMENTO:

```
INSERT INTO NUEVODEPARTAMENTO  
SELECT 'Investigación',  
REF(P)  
FROM PERSONA_NUEVA P  
WHERE Apellido = 'Sanchez';
```

Tabla relacional (normal) vs. nueva Tabla de objetos

1) **CREATE TABLE** Tabla1 (Persona tipo_nuevapersona)

Tabla1 es una tabla relacional (normal) con una sólo columna, que admite añadir más columnas nuevas

2) **CREATE TABLE** Tabla2 **OF** tipo_nuevapersona

Tabla2 es una tabla de objetos, que NO admite añadir más columnas nuevas

3) **CREATE TABLE** Tabla3 (Dept varchar2(5),

DatosPersonales **REF** tipo_nuevapersona
[**SCOPE** Tabla2])

Tabla3 es una tabla normal con una columna **REF** a objeto, donde la opción **SCOPE** restringe a que Tabla 3 sólo tenga OIDs que estén almacenados en Tabla2 (similar a la regla referencial de la clave externa (FKs y PKs))

Referencias

Si queremos tener una descripción completa de la tabla recién creada, la sintaxis sería como sigue:

Set describe depth 2

Desc NUEVODEPARTAMENTO

Nombre	Null?	Type

NomDept	VARCHAR2(30)
PersonaEn	REF OF tipo_nuevapersona
Nombre	VARCHAR2(25)
Apellido	VARCHAR2(25)
Fecha_nac	DATE

Referencias

```
SELECT REF (P)  
FROM PERSONA_NUEVA P  
WHERE Apellido = 'Sánchez';
```

Resultado:

REF(P)

```
000079891B2802295C7AB6AD3B3C4CAD7B080A49C97BACD983561  
9AC8833C49F8DA8CD6E25F067F00700014
```

La función **REF** toma como entrada el alias dado a la tabla de objetos, "P". Y devuelve el **OID** en curso del objeto fila seleccionado (de escasa utilidad).

Referencias

Para insertar una fila en NUEVODEPARTAMENTO, se necesita la función REF para almacenar la referencia a NUEVAPERSONA en la columna PersonaEn:

```
INSERT INTO NUEVODEPARTAMENTO  
SELECT 'Investigación',  
REF(P)  
FROM PERSONA_NUEVA P  
WHERE Apellido = 'Sanchez';
```

- En la tabla PERSONA_NUEVA se inserta el literal "Investigación".
- La función REF retorna el OID a partir de la consulta sobre el objeto PERSONA_NUEVA seleccionado
- Entonces, la tabla NUEVODEPARTAMENTO almacena el OID como un puntero al objeto fila de la tabla PERSONA_NUEVA.
- La tabla NUEVODEPARTAMENTO nunca guarda información de PERSONA_NUEVA, sólo el nombre del departamento (NomDept) y un puntero OID .

Referencias

```
SELECT * FROM NUEVODEPARTAMENTO;
```

Resultado:

NomDept

PersonaEn

Investigación

```
00002202085C3AD6AD3B3C4CD7B080A49C97BACD9835619CD6E2  
5C49F8AC8879791B33DA8F
```

Referencias

Para poder ver el valor del objeto referenciado hay que usar la función **DEREF**. **DEREF** toma el OID y evalúa la referencia para devolver un valor.

```
SELECT DEREf (D.PersonaEn)
FROM NUEVODEPARTAMENTO D
WHERE NomDept = 'Investigación'
```

Resultado:

```
DEREF(D.PersonaEn)(Nombre, Apellido, Fecha_Nac)
-----
tipo_nuevapersona ('Juan', 'Sánchez', '03-FEB-70')
```

Esto demuestra que la tupla 'Juan Sánchez' está referenciada por la tupla 'Investigación' en NUEVODEPARTAMENTO.

Referencias

Para poner junto a la misma estructura del tipo objeto de un objeto tabla, hay que utilizar la función **VALUE** como sigue:

```
SELECT VALUE (P)
FROM PERSONA_NUEVA P
WHERE Apellido = 'Sánchez'
```

Resultado:

VALUE(P) (Nombre, Apellido, Fecha_Nac)

TIPO_NUEVAPERSONA ('Juan', 'Sánchez', '03-FEB-70')

Referencias

Ejemplo PL/SQL:

```
set serveroutput on
declare
v_persona tipo_nuevapersona;
begin
    SELECT VALUE (P) INTO v_persona
    FROM PERSONA_NUEVA P
    WHERE Apellido = 'Sánchez';

    DBMS_OUTPUT.PUT_LINE(v_persona.Nombre);
    DBMS_OUTPUT.PUT_LINE(v_persona.Apellido);
    DBMS_OUTPUT.PUT_LINE(v_persona.Fech_Nac);
end;
```

Resultado:

Juan

Sánchez

03-FEB-70

Herencia

Crear un tipo raíz de una jerarquía de objetos. Sintaxis:

```
CREATE TYPE TIPO_PERSONA AS Object  
  (nombre      varchar2(25),  
   fecha_nac   date  
   member function EDAD() return number,  
   member function PRINTME() return varchar2) not final;
```

Crear un subtipo. Sintaxis:

```
CREATE TYPE TIPO_EMPLEADO UNDER TIPO_PERSONA  
  (sueldo      number,  
   member function PAGA() return number,  
   overriding member function PRINTME() return varchar2);
```

Conclusiones al modelo de las BDO-R

- La capacidad objetual en BDs ha llevado al clásico modelo de datos relacional, MDR, (su tradicional teoría de diseño y la normalización de relaciones) a ser entendido bajo el paradigma O-R en la actual construcción de los SGBDs.
- Se pueden consultar Objetos Anidados y/o Referencias sin necesidad de escribir *joins*.
- Asistimos a un desplazamiento gradual desde el MDR hacia el modelo basado en el paradigma Orientado-a-Objetos. MDR → MDOR, BDR → BDOR
- Oracle no proporciona un modelo Orientado-a-Objetos puro, sino un Modelo Objeto-Relacional.
- Esto provoca cambios docentes y de aprendizaje de Bases de Datos y sus SGBDs.

Tecnología *BD XML-Objeto-Relacional* de Microsoft

Conceptos fundamentales de ADO.NET

Conceptos

Objetos básicos

Trabajo desconectado

Links de utilidad:

MSDN

msdn.microsoft.com

.NET 247

www.dotnet247.com

ADO.net

- Aplicaciones cliente/servidor
 - ~~Conexión permanente~~ **web distribuidas**
 - **datos desconectados**
- Desarrollo rápido
 - Modelo de objetos ~~versátil~~
- Conjunto de datos **muy funcional** **Trabajo extra implícito**
 - Tipo y ubicación de cursor
 - Actualización automática de BD
 - Genérico
 - Personalizable
- **Funcionalidad de acceso a datos separada**

Trabajo con datos

- Desconectados
 - Forma de trabajar de ADO.NET
 - Uso de XML
- Conectados
 - T-SQL: Cursores en el servidor
 - Uso de ADO: afecta el rendimiento

Desarrollo

- Más difícil de aprender
 - Cambio de forma de pensar
 - Más objetos y funcionalidades
- Uso un poco más complejo
 - Modelo de objetos estricto
 - Visual Studio .NET lo simplifica
- Mayor rendimiento
- Mayor potencial

Separación datos-origen

- Funcionalidad de acceso a datos
 - Específica para el origen de datos a utilizar
- Conjunto de datos
 - Sin funcionalidad de acceso a datos
 - Mayor versatilidad
 - Posibilidad de almacenar más datos y metadatos

Proveedores de datos

- Conjunto de objetos creados y optimizados para permitir el acceso a un origen de datos específico
- Interfaces
 - System.Data
- Clases comunes
 - System.Data.Common
- Bases de datos
 - **SQL Server**: System.Data.SqlClient
 - **Oracle**: System.Data.Oracle
- Tecnologías estándar:
 - OLE-DB: System.Data.OleDb
 - ODBC: System.Data.Odbc

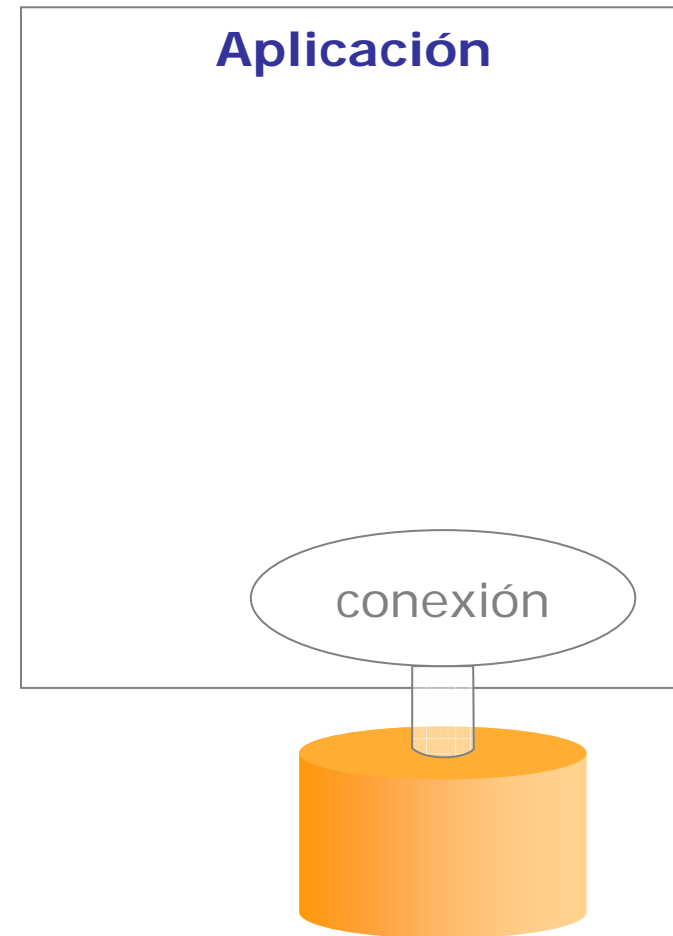
Contenedores de datos

- Dentro de la **biblioteca de clases de la plataforma**
 - System.Data
 - System.Xml
- Utilizan **XML de forma nativa**
 - Potente
 - Fácil persistencia
 - Legible

Objetos de ADO.NET

Conexión

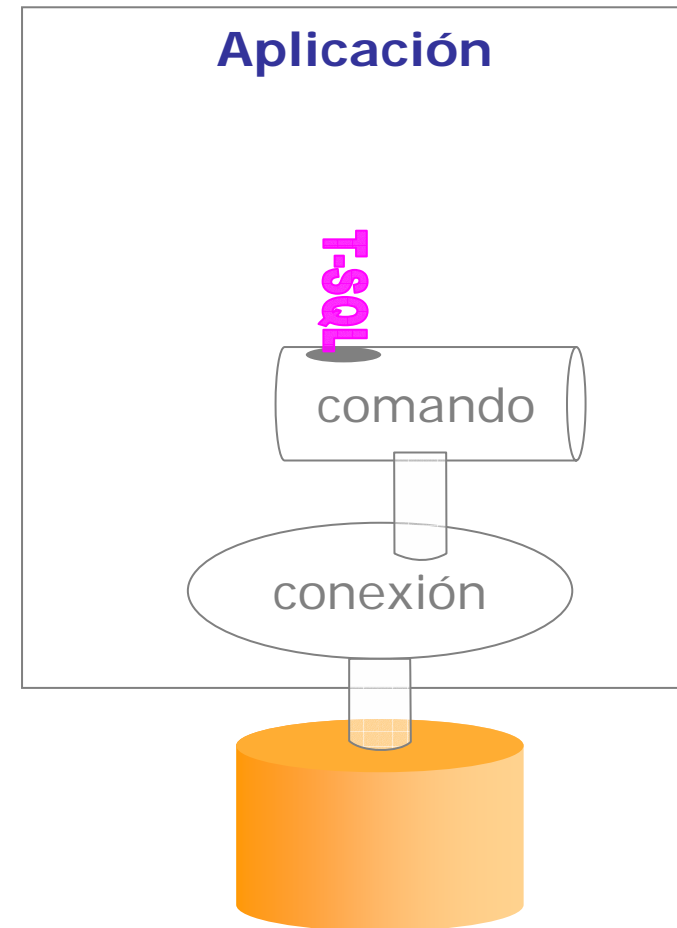
- Vínculo con la BD
- Pooling automático
- Inicia transacciones
- Tipos
 - IDbConnection
 - SqlConnection
 - OleDbConnection



Objetos de ADO.NET

Comando

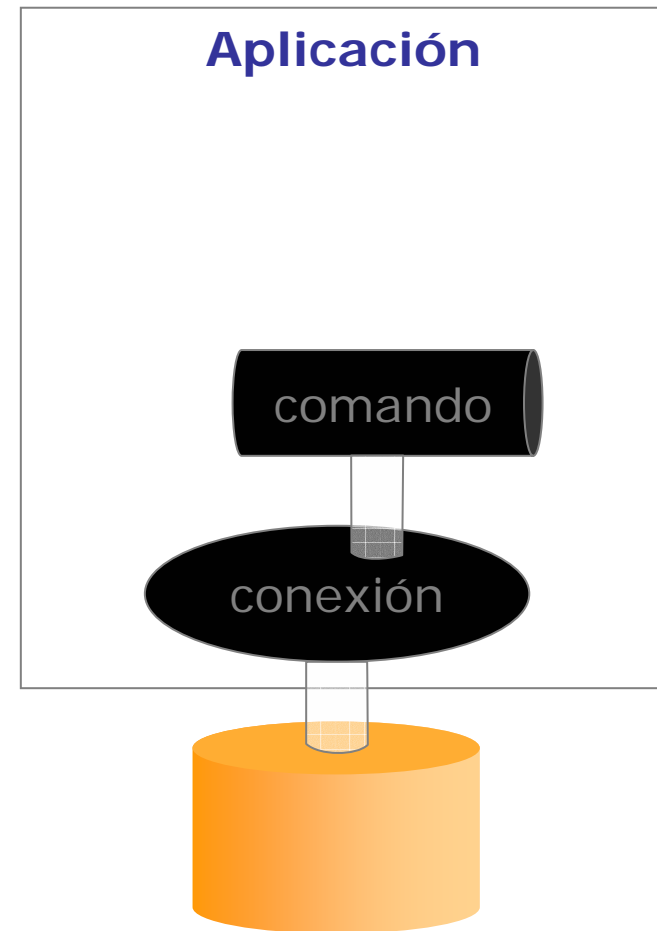
- Envío de sentencias Transact-SQL
 - Parámetros
- Puede devolver
 - Número de filas afectadas
 - Un valor escalar
 - Un lector de datos
- Tipos
 - IDbCommand
 - SqlCommand
 - OleDbCommand



Objetos de ADO.NET

- Lectura secuencial de datos
 - Por fila
 - Por bytes
- Sólo lectura
- Máximo rendimiento
 - Única funcionalidad
 - Forward-only
- Puede manejar varios conjuntos de resultados
- Tipos
 - IDataReader
 - SqlDataReader
 - OleDbDataReader

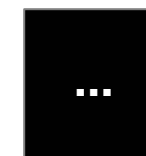
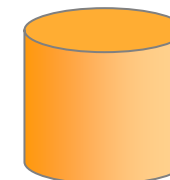
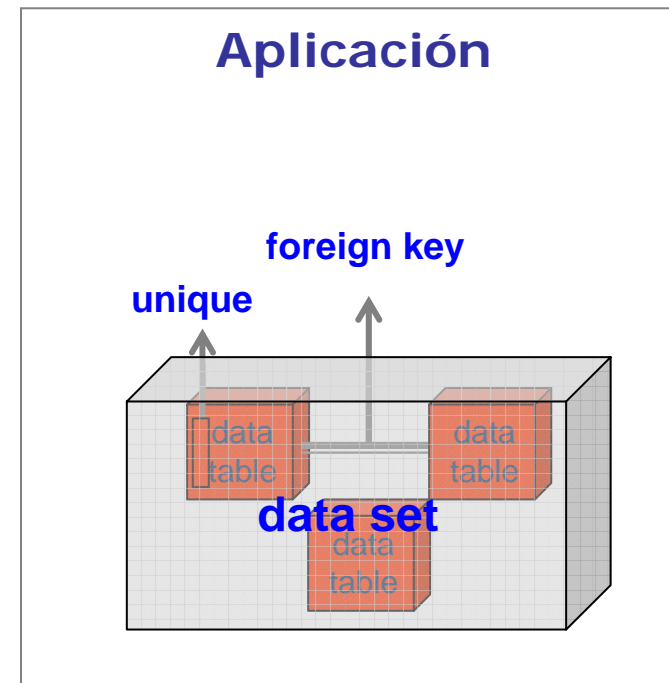
Lector de datos



Objetos de ADO.NET

Haciendo visible lo invisible... Conjunto de datos

- Contenedor de datos
- Cualquier origen de datos
 - Independiente de éste
- Contiene objetos tabla
 - Contienen objetos fila y columna
- Integridad de los datos
 - Restricciones
- Tipos
 - System.Data.DataSet
 - System.Data.DataTable
 - System.Data.DataView



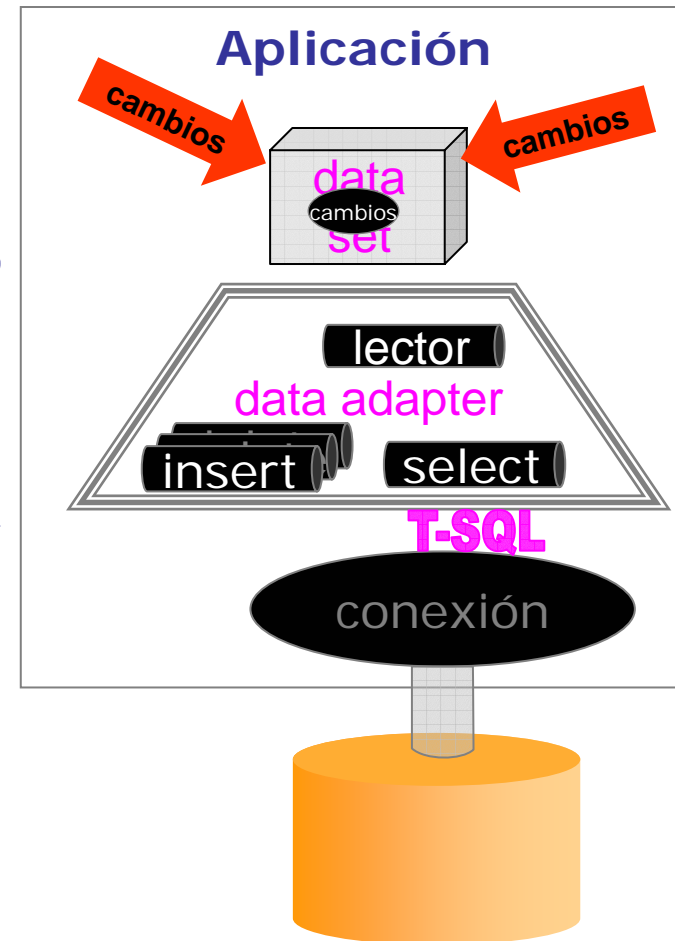
Objetos de ADO.NET **DataSet tipado**

- DataSet preparado para contener datos determinados en tiempo de diseño
 - Agiliza desarrollo
 - Mejora rendimiento
 - Evita errores
- Diseño visual: esquema XSD

Objetos de ADO.NET Adaptador de datos

Haciendo visible lo invisible...

- Orquestador del acceso a datos
- Creado para trabajar con un DataSet
- Contiene un comando de acceso (SELECT) y un lector de datos
- Contiene 3 comandos de actualización y persistencia de la BD (INSERT, UPDATE y DELETE)
- Tipos
 - DbDataAdapter
 - IDbDataAdapter
 - SqlDataAdapter
 - OleDbDataAdapter



Objetos de ADO.NET

Constructor de comandos

- Tipos
 - SqlCommandBuilder
 - OleDbCommandBuilder
- Se vincula con un DataAdapter
- Genera sentencias de actualización de DB basado en el SELECT
- Necesita metadatos de la BD
 - Menor rendimiento
 - Uso no recomendado
 - Usar para sentencias Ad-hoc

Objetos de ADO.NET

- Transacciones **Más objetos...**
 - IDbTransaction
 - SqlTransaction
 - OleDbTransaction
- Manejo de excepciones
 - SqlException y SqlError
 - OleDbException y OleDbError

Práctica para un Grupo de Trabajo

Uso de Conjuntos de Datos

Más información en

- eRevista Danysoft
- Webs Danysoft
- Area Formación y S.Profesionales
- DVD Danysoft
- Libros Danypress
- info@danysoft.com



Referencias Bibliográficas

Consúltese el documento

BDOR 1ª Parte docu de Oracle en 2002 y
ADO.NET en 2007 (Nov 2007).pdf