

Apuntes Complementarios al Programa Docente

Tema V: El Lenguaje SQL de BDR

Estándar SQL:99. Modelo de Datos Objeto-Relacional de 2003

<http://sinbad.dit.upm.es>

● Búsqueda Bibliográfica recomendada

1) Desde hace años, **Andrew Eisenberg y Jim Melton** son promotores destacados de las distintas versiones por las que pasa y avanza el estándar SQL.

Por tanto, se recomienda consultar las referencias bibliográficas que resultan del siguiente acceso:

- ❖ En Google pedir la búsqueda de: "*DBLP Bibliography*"
- ❖ O bien, ir directamente a <http://www.informatik.uni-trier.de/~ley/db/>
- ❖ Una vez allí, ir a la parte *Search* y pinchar en [Author](#)
- ❖ Escribir el nombre "*Jim Melton*"
- ❖ Y también escribir el nombre "*Andrew Eisenberg*"

2) *Desde la Web de Oracle: <http://www.oracle.org/>*

- ❖ Opción "Also try" SQL, ir a <http://encyclopedia.thefreedictionary.com/sql+query>

3) *Desde el sitio Web de Microsoft:*

- ❖ <http://www.microsoft.com/sql/default.mspx>, Also try "Product Information SQL",

4) *Desde los sitios Web de IBM, Ingres, Postgres, Sybase, Informix, etc., podemos ilustrarnos sobre el Lenguaje SQL Objeto –Relacional que hoy posee la actual tecnología de BDO-R.*

Estándar SQL:1999. Su Modelo de Datos Objeto-Relacional de 2003

Algunas Referencias Bibliográficas del actual Estándar SQL

- Andrew Eisenberg, [Jim Melton](#), [Krishna G. Kulkarni](#), [Jan-Eike Michels](#), [Fred Zemke](#): *SQL: 2003*. [SIGMOD Record 33](#) (1): 119-126 (2004)
- Andrew Eisenberg, [Jim Melton](#): *An Early Look at XQuery API for Java (XQJ)*. [SIGMOD Record 33](#)(2): 105-111 (2004)
- Andrew Eisenberg, [Jim Melton](#): *Advancements in SQL/XML*. [SIGMOD Record 33](#)(3): 79-86 (2004)
- [Charles E. Campbell](#), Andrew Eisenberg, [Jim Melton](#): *XML schema*. [SIGMOD Record 32](#)(2): 96-101 (2003)
- [Susan Malaika](#), Andrew Eisenberg, [Jim Melton](#): *Standards for databases on the grid*. [SIGMOD Record 32](#)(3): 92-100 (2003)
- [Jan-Eike Michels](#), [Krishna G. Kulkarni](#), [Christopher M. Farrar](#), Andrew Eisenberg, [Nelson Mendonça Mattos](#), [Hugh Darwen](#): *The SQL Standard*. in - [Information Technology 45](#)(1): 30-38 (2003)
- Andrew Eisenberg, [Jim Melton](#): *SQL/XML is Making Good Progress*. [SIGMOD Record 31](#)(2): 101-108 (2002)
- [Jim Melton](#), Andrew Eisenberg: *An Early Look at XQuery*. [SIGMOD Record 31](#)(4): 113-120 (2002)
- Andrew Eisenberg, [Jim Melton](#): *SQL/XML and the SQLX Informal Group of Companies*. [SIGMOD Record 30](#)(3): 105-108 (2001)
- [Jim Melton](#), Andrew Eisenberg: *SQL Multimedia and Application Packages (SQL/MM)*. [SIGMOD Record 30](#)(4): 97-102 (2001)
- Andrew Eisenberg, [Jim Melton](#): *SQL Standardization: The Next Steps*. [SIGMOD Record 29](#)(1): 63-67 (2000)
- Andrew Eisenberg, [Jim Melton](#): *SQL: 1999, formerly known as SQL 3*. [SIGMOD Record 28](#)(1): 131-138 (1999)
- Andrew Eisenberg, [Jim Melton](#): *SQLJ-Part 1: SQL Routines Using the Java Programming Language*. [SIGMOD Record 28](#)(4): 58-63 (1999)

Estándar SQL:1999. Su Modelo de Datos Objeto-Relacional de 2003

Otras Referencias Bibliográficas del Estándar SQL-BDOR

- *SQL99, SQL/MM, and SQLJ: An Overview of the SQL Standards*, Nelson M. Mattos. IBM Database Common Technology, 1999
- *High Performance Oracle8 Object-Oriented Design*, David A. Anstey. Coriolis Group, 1998

Contenido

1. Introducción a las BDOR
2. LOB, Large Object Data Type (Tipo de Objeto Grande)
3. UDT, User Definition Type (Tipo Definido por el Usuario)
 - 3.1. Distinct Type (Tipo Distinto)
 - 3.2. Structured Type (Tipo estructurado)
4. Métodos
5. Ref Type (Tipo Referencia)
6. Jerarquías de Tablas
7. Collection Type (Tipo Colección): Arrays
8. ROW Type (Tipo fila)
9. Productos: Oracle, DB2, Ingres, Sybase, Informix

1. Introducción a las BDOR

Se extiende la sintaxis de SQL:92 y se amplía su alcance:

- **Extensiones en los Tipos de Datos (dominios) Objeto-Relacionales:**
 - Tipo Definido por el Usuario
 - Tipo fila y referencia
 - Tipo colección (set, bag, list y array)
 - Tipo de Objetos Grandes, LOB (Large Objects)
- **Extensiones en el Control de la Semántica de datos Objeto-Relacionales:**
 - Triggers
 - Procedimientos almacenados y funciones definidas por el Usuario
- **Extensiones en las capacidades Consultivas Objeto-Relacionales:**
 - Consultas recursivas
 - SQL:99 Incorpora al lenguaje XML y viceversa desde 2006
- **Otras Extensiones en las Bases de Datos Objeto-Relacionales:**
 - Extensiones a la ayuda en la toma de Decisiones: OLAP, etc.
 - Intercambio de Objetos (OEM), Datos Multimedia, etc.

1. Introducción a las BDOR

Beneficios de la Extensión Objeto-Relacional

- Extensiones en las capacidades de los SGBDOR vs. SGBDR:
 - Nuevos Tipos de Datos que permiten gestionar aplicaciones más complejas con una gran riqueza de dominios (imagen, voz, sueldo, etc.)
 - Nuevas operaciones que permiten gestionar el comportamiento de los Tipos de Datos
 - Mayor capacidad expresiva para los conceptos y asociaciones complejos
 - Reusabilidad, propio de la Orientación a Objetos. Se pueden compartir diversas bibliotecas de clases ya existentes
 - Integración de lenguajes: relacional, de objetos, XML, etc. En un solo lenguaje
 - Nuevas Consultas con mayor capacidad consultiva (consultas anidadas, recursivas, almacenadas, pre-fabricadas), etc.

1. Introducción a las BDOR

Tipos de Datos Objeto-Relacionales

- Tipo Objeto: Large Object (LOB)
 - ❖ Binary Large Object (BLOB)
 - ❖ Character Large Object (CLOB)
- Tipo Definido por el Usuario (UDT)
 - ❖ Distinct types
 - ❖ Structured types
- Constructores de tipo objeto
 - ❖ Row types
 - ❖ Referenced types
- Tipos colección
 - ❖ Set, Bag, List y Arrays
- Métodos, funciones y procedimiento definidos por el usuario
- Jerarquías de tablas y de vistas

2. Tipo Large Object: LOB

- Tipo **LOB**, se almacena en la BDOR como cualquier atributo. Su gran capacidad puede definirse en DDL como: KB, MB o Gigabytes
- LOB tiene dos subtipos
 - ❖ Binary Large Object, **BLOB**, almacena objetos multimedia: audio, imagen, mapas, vídeo, fotos.
 - ❖ Character Large Object, **CLOB**, almacena texto
- Ejemplo: **CREATE TABLE** documental (
 titulo **VARCHAR2(200)**,
 id_doc **INTEGER**,
 resumen **CLOB (32K)**,
 texto_doc **CLOB (20M)**,
 video_film **BLOB (2G)**);

2. Tipo Large Object: LOB

- El Tipo **LOB** se gestiona como cualquier columna de una BD:
 - Consultar un valor o parte de él (**SELECT...**),
 - Añadir valores (**INSERT...**), con algunas restricciones
 - Borrar (**DELETE...**)
 - Actualizar (**UPDATE...**), con algunas limitaciones

- Adicionalmente, el tipo LOB tiene permitidas las operaciones siguientes:
 - ❖ Predicado **LIKE**
 - ❖ La yuxtaposición o concatenación de objetos
 - ❖ Funciones: **LENGTH, SUBSTRING, POSITION, TRIM**

2. Tipo Large Object: LOB

- Debido al gran tamaño del tipo LOB, el SGBDOR precisa la definición de 'buffers' del tamaño adecuado para su gestión, según sea cada caso de LOB.
- Cuando un LOB es del orden de GB, entonces la cláusula **LOCATOR** ayuda a los aplicativos y al SGBDOR para una mejor gestión de estos LOB muy grandes.
- Sin embargo, los LOB no tienen permitidas las siguientes operaciones :
 - ❖ **GREATER THAN** y **LESS THAN**
 - ❖ Claves primarias **PK**, claves externas **FK** y **UNIQUE**
 - ❖ **GROUP BY** y **ORDER BY**
 - ❖ Operadores: **UNION**, **INTERSECT** y **EXCEPT**
 - ❖ **JOIN** (como columnas del Join)

3. Tipo Definido por el Usuario

- SQL:92 no es fuertemente tipado
- SQL:99 es un lenguaje **Fuertemente Tipado**. Los tipos tienen **Comportamiento**
- Posee el tipo **Distinct Type** definido al renombrar a un tipo existente:
 - ❖ Ambos tipos comparten la misma representación
 - ❖ Tienen distinto comportamiento
 - ❖ El tipo distinto no es comparable con el tipo original
 - ❖ **Son fuertemente tipados**
- **Distinct Type** tiene las siguientes operaciones:
 - ❖ Operadores de comparación
 - ❖ Operadores de CAST
 - ❖ Métodos y funciones
 - ❖ No hay mecanismos de herencia entre tipos distintos

3. Tipo Definido por el Usuario

Ejemplo de Tipo Distinto, primero se define la tabla:

```
● CREATE TABLE Sala (  
    » IdSala CHAR (10),  
    » LongSala INTEGER,  
    » AnchoSala INTEGER,  
    » AreaSala INTEGER,  
    » PerimSala INTEGER));
```

Si hiciéramos:

```
UPDATE Sala  
SET AreaSala=LongSala;
```

Error: tipo distinto, no es comparable con el tipo original

3. Tipo Definido por el Usuario

- Ejemplo de Tipo Distinto

```
CREATE TYPE tipoSala  
AS CHAR(10) FINAL;
```

```
CREATE TYPE tipoMetros  
AS INTEGER FINAL;
```

```
CREATE TYPE tipoMetrosCuad  
AS INTEGER FINAL;
```

```
CREATE TABLE Sala (  
  IdSala tipoSala,  
  LongSala tipoMetros,  
  AnchoSala tipoMetros,  
  AreaSala tipoMetroscuad,  
  PerimSala tipoMetros));
```

```
UPDATE Sala  
SET AreaSala=LongSala;
```

Incorrecto

```
UPDATE Sala  
SET AnchoSala=LongSala;
```

Correcto

3. Tipo Definido por el Usuario

- SQL 99 posee el **Tipo Estructurado** con las siguientes características:
 - ❖ Fuertemente Tipado
 - ❖ Comportamiento, Encapsulación, Sustitución, Polimorfismo
 - ❖ Ligadura dinámica
 - ❖ Verificación de tipos estática
- **Structured Type** se puede usar como:
 - ❖ **Column Type (Tipo Columna):**
 - Para modelar *atributos* de las entidades. Ej: Texto, imagen, vídeo, series de tiempo, punto, línea...
 - Facilita el uso de objetos multimedia (SQL/MM)
 - ❖ **Row Type (Tipo Fila):**
 - Para modelar *entidades* con relaciones y comportamiento. Ej: empleado, departamento, alumno...
 - Facilita el uso de objetos de negocio

3. Tipo Definido por el Usuario

- Tipo Estructurado
- Ejemplo de **Tipo Row (Fila)**:

```
CREATE TYPE empleado AS (  
    id INTEGER,  
    nombre VARCHAR (20));
```

3. UDT, Tipo Definido por el Usuario

- **UDT** es una sentencia DDL que define un tipo del interés del usuario y a la medida de ciertas aplicaciones.
- Su sintaxis se inicia con la cláusula:

CREATE TYPE nombre_UDT AS (...

- Cada UDT se define en términos de:
 - otros tipos del lenguaje SQL y/o
 - otros tipos definidos previamente por el usuario.

3. Tipo Definido por el Usuario

```
CREATE TYPE direccion AS (  
    calle VARCHAR (40),  
    ciudad VARCHAR (20),  
    provincia VARCHAR (40),  
    código_postal INTEGER (5);  
  
CREATE TYPE foto-bitmap AS BLOB FINAL;  
  
CREATE TYPE chalet AS (  
    propietario REF (persona),  
    precio INTEGER,  
    habitaciones INTEGER,  
    tamaño DECIMAL (8,2),  
    ubicación direccion,  
    imagen foto-bitmap) NOT FINAL
```

3. UDT, Tipo Definido por el Usuario

- El UDT se puede utilizar como:
 - ❑ Atributo de otro tipo estructurado
 - ❑ Parámetro de función (es), método(s) y procedimiento(s)
 - ❑ Variable SQL
 - ❑ Dominio o columna
 - ❑ También se pueden usar para definir **tablas y vistas**

Ejemplo: **CREATE TABLE** bien_inmueble **OF** chalet

Estándar SQL:1999. Su Modelo de Datos Objeto-Relacional de 2003

4. Métodos

- El Método: es una función SQL ligada a un Tipo Definido por el Usuario
 - ❖ Se define en el esquema
 - ❖ La signatura se define separada de la especificación del cuerpo

```
CREATE TYPE empleado AS (  
    id INTEGER,  
    nombre VARCHAR (20),  
    sueldo_base DECIMAL (9,2),  
    primas DECIMAL (9,2),  
    INSTANTIABLE NOT FINAL  
    METHOD sueldo() RETURNS DECIMAL (9,2));
```

```
CREATE METHOD sueldo() FOR empleado  
begin  
.....  
end;
```

- Los métodos se invocan utilizando la notación "punto"

```
SELECT mgr.sueldo()  
FROM empleado
```

4. Métodos. Diferencias entre Funciones y Métodos

- El método no queda necesariamente anclado a la definición de un tipo específico. Aunque sí queda vinculado exactamente a un tipo.
- Existen diversas formas de invocación del método y de ligadura a los posibles tipos de datos durante el tiempo de compilación y/o de ejecución (early binding and late binding, polimorfismo, definición de interfaz vs. implementación, herencia, etc.).

Sintaxis de invocación: Notación funcional, Notación punto

- ❖ Esquema de residencia: Cualquier esquema, Esquema de su tipo asociado
- ❖ Resolución de rutina: Totalmente resuelto en tº de compilación
- ❖ La compilación resuelve un conjunto de métodos candidatos, y se termina de resolver durante la ejecución

4. Tipos de Métodos

- Existen dos tipos de Método:
 - ❖ Originales (del supertipo)
 - ❖ Overriding (de los subtipos)

```
CREATE TYPE empleado AS (  
    id INTEGER,  
    nombre VARCHAR (20),  
    sueldo_base DECIMAL (9,2),  
    primas DECIMAL (9,2),  
    INSTANTIABLE NOT FINAL  
    METHOD sueldo() RETURNS DECIMAL (9,2));
```

```
CREATE TYPE gestión UNDER empleado AS (  
    stock_option INTEGER)  
    INSTANTIABLE NOT FINAL  
    OVERRIDING METHOD sueldo() RETURNS DECIMAL (9,2),  
    METHOD original1 () RETURNS INTEGER
```

- **Ligadura dinámica:** los métodos sobrecargados se resuelven en tiempo de ejecución (vinculándose al más específico)

5. Tipo REF (Referencia)

CREATE TYPE chalet AS (

propietario **REF** (persona),
precio **INTEGER**,
habitaciones **INTEGER**,
tamaño **DECIMAL** (8,2),
ubicación **direccion**,
imagen **foto-bitmap**) **NOT FINAL**

CREATE TABLE bien_inmueble OF chalet
(REF IS oid USER GENERATED)

Los atributos del tipo se convierten en columnas de la tabla

Se añade una columna más para definir el valor **REF** de la fila (**OID**)

- Tablas Tipadas

5. Tipo REF (Referencia)

- Un tipo estructurado tiene un tipo referencia correspondiente
- El tipo **REF** se puede usar donde se use cualquier otro tipo

Su sintaxis puede ser:

- User generated (**REF USING** <tipo predefinido>)

```
CREATE TYPE bien_inmueble AS ( ....  
NOT FINAL REF USING INTEGER
```

- System generated (**REF IS SYSTEM GENERATED**)
- Derivada de un conjunto de atributos (**REF (<atributos>)**)

```
CREATE TYPE persona AS ((nss INTEGER, nombre VARCHAR(20), ...)  
NOT FINAL REF (nss)
```

5. Tipo REF (Referencia)

- El tipo **REF** no tiene la misma semántica que la regla de referencias cruzadas o restricción de integridad referencial
 - La integridad referencial implica una dependencia de inclusión y el tipo **REF** no.

```
SELECT ch.precio, ch.propietario->nombre
FROM chalet ch
WHERE ch.propietario->ciudad="Madrid"
```

```
SELECT e.nombre, e.departamento ->num_empleados()
FROM empleado e
Eliminación de JOIN
```

Llamada a métodos

```
SELECT e.nombre, Deref (e.departamento)
FROM empleado e
WHERE e.nombre="J. Pérez"
```

Obtención de datos a los que se referencia

6. Jerarquías de Tablas

- La tablas tipadas pueden tener subtablas
- Las subtablas heredan: atributos, restricciones, disparadores, métodos.

```
CREATE TYPE persona AS .... NOT FINAL
CREATE TYPE bienes AS (dueño REF (persona), .... NOT FINAL
CREATE TYPE tierras UNDER bienes AS .... NOT FINAL
CREATE TYPE casa UNDER bien_inmueble AS ....NOT FINAL
CREATE TYPE chalet UNDER bien_inmueble AS ....NOT FINAL
CREATE TABLE personas OF persona (...)
CREATE TABLE bienes OF bien_inmueble
CREATE TABLE tierras OF tierra UNDER bienes
CREATE TABLE casas OF casa UNDER bienes
```

```
Bienes:: {Bien_Inmueble, Tierras}
Bien_Inmueble:: {Casa, Chalet}
Personas:: {Dueño}
```

6. Jerarquías de Tablas

- Las consultas sobre la supertabla, devuelven también las filas de las subtablas

```
SELECT *  
FROM bien_Inmueble  
WHERE ....
```

- Se puede restringir las filas seleccionadas a las de la supertabla

```
SELECT *  
FROM ONLY bien_Inmueble  
WHERE ....
```

6. Jerarquías de Tablas

- Path expression (referencias con ámbito)

```
SELECT ch.precio, ch.dueño->nombre  
FROM chalet ch  
WHERE ch.dueño->direccion.ciudad="Madrid"
```

- Las referencias pueden servir para invocar métodos

```
SELECT ch.precio, ch.dueño->salario (2000)  
FROM chalet ch
```

- Las referencias pueden servir para obtener el valor del tipo estructurado al que está referenciando (derreferencia)

```
SELECT ch.precio, Deref (ch.dueño)  
FROM chalet ch
```

7. Tipo Collection: Arrays

- Los arrays permiten soportar columnas que contengan colecciones de elementos (NFNF)
- Características:
 - Diferencia entre la longitud máxima y la actual (como el CHAR VARYING)
 - Pueden definirse sobre cualquier tipo (excepto array)
 - Pueden definirse arrays en “cualquier sitio”
- Operaciones:
 - Acceso a los elementos por un número ordinal,
 - Cardinalidad, Comparación, Constructores, Asignación,
 - Concatenación, CAST, Facilidades de selección declarativa sobre arrays (por contenido o por posición),
 - Transformación implícita de un array en una tabla...

7. Tipo Collection: Arrays

● Ejemplo:

```
CREATE TABLE libro (  
    título VARCHAR (200),  
    id_libro INTEGER,  
    autores VARCHAR (15) ARRAY [20],  
    resumen CLOB (32K),  
    texto_libro CLOB (20M),  
    película BLOB (2G));
```

```
INSERT INTO libro (título, id_libro, autores)  
VALUES ("A guide to the SQL Standard", 15, ['Date', 'Darwen'])
```

```
SELECT id, autores[1] AS nombre  
FROM libro
```

8. Tipo ROW

```
CREATE TABLE empleado (  
    id_emp INTEGER,  
    nombre ROW (  
        nombre_de_pila VARCHAR(30),  
        apellido VARCHAR(30) ),  
    dirección ROW (  
        calle VARCHAR(50),  
        ciudad VARCHAR(30),  
        provincia CHAR(2) ),  
    sueldo REAL );  
  
SELECT E.nombre.apellido  
FROM empleado E
```

9. Productos comerciales BDOR

- ❖ DB2 Universal Database de IBM,
<http://www.ibm.com>
- ❖ Universal Server de INFORMIX (ahora de IBM),
<http://www.informix.com>
- ❖ INGRES II, Computer Associates,
<http://www.ingres.com>
- ❖ ORACLE de Oracle Corporation,
<http://www.oracle.com>
- ❖ SYBASE de SYBASE,
<http://www.sybase.com>

9. El Producto BDOR de Oracle

Véanse apuntes de BSDT sobre BDOR de Oracle en <http://sinbad.dit.upm.es>, Docencia/ grado/2005-2006 y (Ansteiy,1998)

- ❖ Tabla de Vista
- ❖ Vistas de Tablas
- ❖ Tabla de Objetos
- ❖ Vista de Objetos
- ❖ Tipos Predefinidos
- ❖ Tablas de Objetos
- ❖ Tipos Def. Usuario