Unidad 5

Base de datos de objetos y objeto-relacionales

Acceso a datos

5

Índice

Introducción

Con el crecimiento de aplicaciones basadas en lenguajes de programación orientados a objetos y el auge de base de datos relacionales, empezaron a surgir distintos problemas para la persistencia de datos, debido a que cada lenguaje de programación tenía que acoplarse fuertemente a la base de datos.

Se busco una de las soluciones aplicando las técnicas de ORM, haciendo uso de la correspondencia entre los objetos del lenguaje de programación y la base de datos y surgiendo un nuevo problema, el desface objeto-relacional.

A su vez, surgieron dos tipos de base de datos, **Base de datos objeto (BDO)** y **Base de datos objeto-relacional (BDOR)**, buscando el mismo objetivo, manejar objetos directamente desde la base de datos. En esta unidad se profundizará en estos dos tipos de base de datos.

Al finalizar esta unidad

- Entenderemos las bases de datos objeto (BDO).
- Conoceremos el estándar ODMG.
- Aprenderemos los lenguajes ODL y OQL.
- Entender eremos las bases de datos objeto-relacional (BDOR).
- Conoceremos los tipos de uso de objetos de BDOR en Oracle.
- Aprenderemos a definir los objetos en Oracle.

5.1. Bases de datos de objetos y objeto-relacionales, y correspondencia objeto-relacional

Con las **bases de datos relacionales** no es posible almacenar objetos directamente, hay que adaptar los objetos al modelo formal relacional. Una de las soluciones fue aplicar la **correspondencia objeto-relacional**, una técnica basada en especificar o mapear en un fichero como se representa un objeto en el modelo relacional. La correspondencia objeto-relacional presenta un nuevo problema, el desfase objeto-relacional o desajuste. Un objeto puede estar desincronizado de la base de datos y no reflejar los cambios en el lado de la base de datos o viceversa.

Como solución surgió un nuevo tipo, las **bases de datos de objeto (BDO)** que permiten guardar directamente objetos en la base de datos. A raíz de este tipo de base de datos e intentar estandarizar el proceso, se creó la ODMG, llegando a la versión 3.0 en 2000. A partir de su estandarización, se creó el estándar ODL para definir los datos y el lenguaje OQL para consultas. En la actualidad las BDO tienen un uso muy limitado y existen muy pocas bases de datos que implementen este tipo, por ejemplo, Objectivity/DB, db4o y Matisse.

A partir de la revisión de SQL:99, surgió otro tipo, las **bases de datos objeto-relacional (BDOR)**. Son base de datos relacionales que implementan una capa intermedia para usar objetos directamente en la base de datos. Hoy en día, existen dos grandes bases de datos que usan ese tipo, Oracle y Postgre-SQL.

5.2. Características de las bases de datos de objetos

El concepto aprendido del modelo relacional ya no se aplica para este tipo de base de datos, ya se trabaja directamente con objetos, tal y como se pueden representar en un lenguaje de programación orientado a objetos.

Los sistemas gestores de base de datos de objetos (SGBDO o ODBMS) deben de tener las siguientes características:

- **Soportar objetos complejos.** Los objetos complejos están compuestos por un conjunto de atributos, referencias a otros objetos e incluso colecciones.
- **Identidad de objetos.** Poder diferenciar entre un objeto a otro a partir de dar una identidad única de objetos. Se puede distinguir entre el concepto de igualdad, dos objetos son iguales si comparten los mismos valores de sus atributos, y el concepto de identidad, dos objetos son el mismo si tienen el mismo identificador.
- **Tipos o clases.** Tener disponibilidad de tipos primitivos y clases.
- **Extensibilidad.** Ya existe un conjunto de tipos y clases predefinidas. Debe tener la posibilidad de definir nuevos tipos.
- **Completitud computacional.** Hacer cualquier cálculo usando lenguaje de manipulación de datos (DML).
- **Métodos de consulta sencillos.** Que sea eficientes e independientes de cualquier base de datos y lenguaje de programación.

5.3. El estándar ODMG

Object Data Management (ODMG) define el estándar de persistencia de objetos en la base de datos. La última versión ODMG 3.0 del año 2000 especifica:

- El modelo de objetos basado en OMG (Object Management Group).
- Lenguaje de definición de objetos (ODL).
- Lenguaje de consulta de objetos (OQL).
- Vinculaciones con los lenguajes de programación (language bindings).

En las bases de datos relacionales, se basan en SQL como lenguaje de definición de objetos y consulta y es independiente al lenguaje de programación de la aplicación. En las bases de datos de objetos es el propio lenguaje de programación que ofrece los mecanismos para crear, modificar y borrar objetos persistentes.

5.4. ODL

Lenguaje de definición de objetos (ODL) equivale a (DDL) en las bases de datos relacionales.

ODL	DDL
Objetos	Relacional
Clase	Tabla
Instancia (Objeto)	Fila

5.4.1. Modelo de objetos de ODL

El modelo de objetos presenta las siguientes características:

1. El estado de un objeto es cambiante en el tiempo según el valor de sus propiedades.
2. Las propiedades pueden ser atributos de un objeto o relaciones, una a uno, uno a muchos o muchos a muchos.
3. Un objeto es una instancia de una clase, de otro modo, es una representación con valores de la estructura de una clase. El objeto instanciado debe tener un identificador único de objeto (OID).
4. Un objeto puede contener un conjunto de operaciones, definida como conducta (behavior).
5. Un tipo, estructura que asume un objeto tiene una especificación externa. La parte visible que se puede invocar desde objeto, que especifica las operaciones, propiedades (get o set) y excepciones que se pueden producir.

5.4.2. Clases e interfaces

Las clases e interfaces son tipos de un objeto que cumplen similitud de características que en el lenguaje Java.

Clase

Representación abstracta de una entidad real. Contiene atributos y operaciones. Las clases pueden aplicar el concepto de herencia, es decir, la forma de relacionarse en forma de jerarquía con otras clases. Las clases también puede heredar de una interfaz.

En la herencia, se aplica un concepto nuevo, la extensión (haciendo uso de la palabra clave *extent*), que permite acceder a todas las instancias de una clase, además se puede hacer uso de una clave (*key*) para identificar el objeto inequívocamente. En el SGBDO se aplica una restricción de integridad en el que en una relación de herencia no puede haber dos objetos con el mismo valor de clave.

Interfaz

Representación abstracta de operaciones. No se puede crear instancias. Las interfaces pueden aplicar el concepto de herencia, sobre otras interfaces o clases.

5.4.3. Relaciones

Cuando dos objetos tienen un punto de unión, comparten un concepto o información, se dice que tienen una relación. Las relaciones se representan como propiedades de una clase y puede ser uno a uno (1 – 1), uno a mucho (1-N) o muchos a muchos (N-M).

Los *tranversal path* es la representación de las relaciones en los objetos. Siempre aparece un atributo en cada objeto que identifica la relación. Y se tiene que cumplir lo siguiente:

- Si la cardinalidad es 1, el atributo es único. Tendrá operaciones para consultar o modificar el atributo directamente.
- Si la cardinalidad es N, el atributo será una colección. Tendrá operaciones para añadir, modificar o eliminar objetos de la colección.
- Las operaciones que intervienen en la relación tienen que ser públicas.

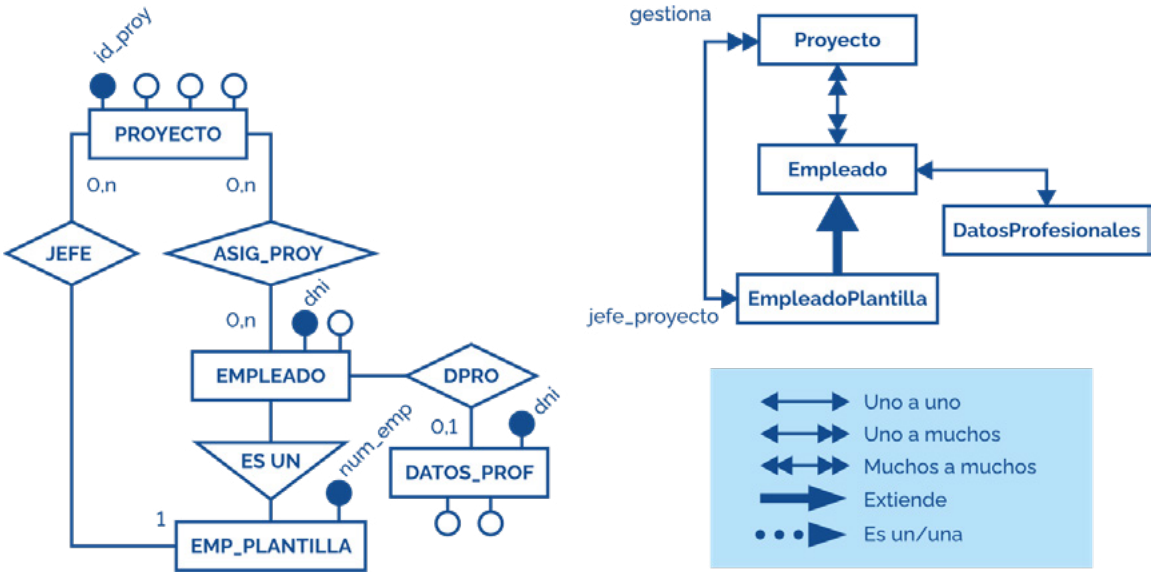

Imagen 1. Ejemplo de diagrama E-R para ODL

5.5. OQL

OQL (Object Query Language) es el lenguaje de consulta del estándar ODMG 3.0. La sintaxis es prácticamente idéntica a las sentencias SELECT de SQL pero se manejas objetos en vez de registros.

La sintaxis básica de OQL es una estructura SELECT... FROM... WHERE.

En las consultas se define los puntos de entrada, que suele ser el nombre de una clase y da acceso a un objeto o colección de objeto. El resultado de cualquier consulta puede ser:

- **Tipo `<bag>`.** En un bag hay una colección de objetos, pudiendo haber repetidos.
- **Tipo `<set>`.** Hay que usar la sentencia `SELECT DISTINCT`. En el set hay una colección de objetos sin elementos repetidos.

Se pueden utilizar expresiones de caminos para especificar un camino a una propiedad y objetos relacionados. Una expresión de camino empieza normalmente con un nombre del punto de entrada o una variable iterador, seguida de ninguno o varios nombres de relaciones o de atributos conectados mediante un punto.

```
class Empleado (extent empleados key dni)

SELECT emp.nombre
FROM empleados emp
WHERE emp.fecha_alta > date '2020-12-01';
```

Imagen 2. Consulta con punto de entrada Empleado

Es posible definir nuevas estructura que agrupen varios atributos, haciendo uso de `struct(nombre: atributo, ...)`.

```
class Empleado (extent empleados key dni)

SELECT struct(nombre: emp.nombre, alta: emp.fecha_alta)
FROM empleados emp
WHERE emp.fecha_alta > date '2020-12-01';
```

Imagen 3. Consulta con uso de struct()

Otras cláusulas que se pueden usar son `GROUP BY`, `HAVING` y `ORDER BY`, como algunos operadores de agregación min, max, count.

OQL solo permite hacer sentencias `SELECT`, no existe sentencias para `INSERT`, `UPDATE` o `DELETE`, estas operaciones se realizan directamente desde el lenguaje de programación.

5.6. SQL:99 y bases de datos objeto-relacionales

SQL ha sufrido diferentes revisiones para incluir mejoras y añadir funcionalidades nuevas. Una de ellas es la versión SQL:99 o SQL3 que introducen los tipos estructurados definidos por el usuario. Esta revisión rompe el esquema de las bases de datos relacional, si podemos hacer estructuras de tipo de datos, por ejemplo, una estructura que contenga más de un atributo, ya no se cumple la atomicidad. Es decir, en una columna puede haber objetos, array, referencias o tablas anidadas.

A partir de esta actualización podemos indicar que:

- Si no se usa esta característica, la base de datos es puramente relacional.
- Si se usa las características de estructuras, la base de datos es de objeto-relacional (BDOR).

Un ejemplo claro de base de datos, que funciona como relacional u objeto-relacional, es **Oracle.**

5.7. Características objeto-relacionales de Oracle

Oracle es una base de datos muy extendida, debido a su facilidad de integración, seguridad y versatilidad, existe diferentes versiones para adaptarse a cualquier sistema. Se basa en el modelo relacional, pero aplica una capa intermedia permitiendo definir las características objeto-relacional en estructuras como:

- Tipos de objetos (clases).
- Relación entre objetos (herencia).
- Tablas de objetos. Cada fila representa un objeto.
- Tablas con columnas de objetos. La columna guarda un objeto.
- Tablas con columnas de arrays.
- Tablas con columnas de tablas. En la columna hay una tabla anidada.

Usando SQL:99 vamos a ver cómo crear cada estructura en Oracle.

5.7.1. Tipos de objetos

Es una estructura definida por el nombre del tipo y puede contener atributos y métodos.

La sintaxis es:

```
CREATE TYPE nombre AS OBJECT (
atributo Tipo,
…
)
```

Los atributos que se definen pueden ser tipos primitivos de la base de datos u objetos,:

```
CREATE OR REPLACE TYPE DatosAcademicos AS OBJECT (
    titulo VARCHAR2(100),
    fecha DATE,
    centro VARCHAR2(100)
);
/
CREATE OR REPLACE TYPE Persona AS OBJECT (
    dni CHAR(9),
    nombre  VARCHAR2(60),
    datos_academicos DatosAcademicos
);
/
```

Imagen 4. Estructura Persona y DatosAcademicos

5.7.2. Herencia

Es posible definir relaciones de herencia simple. Es necesario declarar la clase padre con la palabra reservada `NOT FINAL` y la subclase usar la palabra reservada `UNDER` nombre en la cabecera.

```
CREATE OR REPLACE TYPE Persona AS OBJECT (
    dni CHAR(9),
    nombre  VARCHAR2(60)
) NOT FINAL;
/
CREATE TYPE PersonaPlantilla UNDER Persona (
    num_persona INT
);
/
```

Imagen 5. Estructura con herencia

5.7.3. Objetos de fila y objetos de columna

En la estructura de una tabla podemos definir: `row object`, un objeto por fila u `object column`, un objeto por columna.

Objecto por fila (row object)

Al crear la tabla especificamos con `OF` el objeto, y podemos especificar las restricciones de los atributos (`PK`, `NOT NULL`, etc.)

```
CREATE TABLE personas OF Persona (
    dni PRIMARY KEY,
    nombre NOT NULL
);

INSERT INTO personas VALUES (Persona('11223344X','Pepe'));
INSERT INTO personas VALUES (PersonaPlantilla('22334455Q','María',1));
```

Imagen 6. Tabla con row object y ejemplo de inserción de datos

Solo cuando se crea una tabla por row object podemos obtener las referencias de los objetos (`OID`). Podemos consultar las referencias usaremos la función `REF()`.

```
SELECT REF(P) FROM personas P;
```

Resultado de la Consulta

SQL | Todas las Filas Recuperadas: 2 en 0,08 segundos

	REF(P)
1	SYSTEM.PERSONA('11223344X', 'Pepe')
2	[SYSTEM.PERSONAPLANTILLA]

Imagen 7. Obtener las referencias de objetos persona

Objecto por columna (object column)

Esta forma es más sencilla, al crear la tabla, se especifica en los atributos los objetos que representan la columna.

```
CREATE TABLE personas (
    referencia INT PRIMARY KEY,
    persona Persona not null
);

INSERT INTO personas VALUES (1, Persona('11223344X','Pepe'));
INSERT INTO personas VALUES (2, PersonaPlantilla('22334455Q','María',1));
```

Aquí está el atributo referencia

Aquí estamos introduciendo los rows objects creados antes.

Imagen 7. Tabla con object column y ejemplo de inserción de datos

5.7.4. Tipos de objetos con referencia a otros tipos de objetos

Cuando existan objetos que van a usarse en diferentes tablas, no es recomendable que se cree una copia de cada uno de ellos para cada tabla, ¿Y si se modifica el objeto? Se tiene que cambiar por igual en todas las tablas. Lo habitual es almacenar la referencia del objeto (`OID`) y que cada tabla guarde la referencia al objeto, de esta manera, si se modifica el objeto, los cambios se verán reflejado en todas las tablas que hace referencia.

Para indicar a un atributo de un objeto que se va a almacenar una referencia, se usa la palabra reservada `REF` seguido del nombre del objeto. Cuando se vaya a crear la tabla a partir del objeto, se indicará al atributo que guarda las referencias el ámbito donde obtenerlos, es decir, la tabla donde hace referencia, usando `SCOPE IS`.

```
CREATE OR REPLACE TYPE Actividad AS OBJECT (
    nombre VARCHAR2(100),
    descripcion VARCHAR2(100),
    supervisor REF Persona
);
/
CREATE TABLE actividades OF Actividad (
    nombre NOT NULL,
    descripcion NOT NULL,
    supervisor SCOPE IS personas
);

INSERT INTO actividades
SELECT Actividad('Buceo','Iniciacion al buceo en piscina', TREAT(REF(p) AS REF Persona))
FROM personas p
WHERE p.dni='11223344X';
```

Imagen 8. Ejemplo de manejo de referencias

En el ejemplo de la imagen, se puede ver que al insertar un registro se está usando la función `TREAT`, indicando que la referencia pertenece al objeto indicado.

Si consultamos el registro creado se podrá observar que, en vez de guardar una copia del objeto, ha guardado una referencia a él.

```
SELECT REF(a)
FROM actividades a;
```

sultado de la Consulta

SQL | Todas las Filas Recuperadas: 1 en 0,001 segundos

	REF(A)
1	SYSTEM.ACTIVIDAD('Buceo', 'Iniciacion al buceo en piscina', 'oracle.sql.REF@d0df48e2')

Imagen 9. Consulta de datos con referencias

Por defecto, si se consulta directamente el campo que guarda la referencia, automáticamente muestra los datos del objeto y no la referencia. Aun así existe la función **DEREF que permite obtener los datos del objeto a partir de una referencia.**

```
SELECT DEREF(a.supervisor)
FROM actividades a;
```

sultado de la Consulta

SQL | Todas las Filas Recuperadas: 1 en 0,004

	DEREF(A.SUPERVISOR)
1	SYSTEM.PERSONA('11223344X', 'Pepe')

Imagen 10. Obtener los valores de un objeto con DEREF

5.7.5. Tipos de datos de colección: VARRAY y tablas anidadas

Igual que existen en los lenguajes de programación los arrays, en los objetos de la base de datos también es posible definirlos con la palabra reservada `VARRAY`. Su uso es idéntico, se basa en índices para indicar la posición del array.

```
CREATE OR REPLACE TYPE Telefono AS VARRAY(2) OF CHAR(9);
/
CREATE OR REPLACE TYPE Persona AS OBJECT (
    dni CHAR(9),
    nombre  VARCHAR2(60),
    telefonos Telefono
) NOT FINAL;
/
CREATE TABLE personas OF Persona (
    dni PRIMARY KEY,
    nombre NOT NULL
);
/
INSERT INTO personas VALUES(Persona('11223344X','Pepe',Telefono('666777888','900888000')));
```

Imagen 11. Crear un objeto array

Otra posibilidad es incrustar o anidar una tabla en una columna. Para ello, se creará un objeto referenciando un objeto tabla con `AS TABLE OF REF`.

```
CREATE OR REPLACE TYPE Alumno AS TABLE OF REF Persona;
/
CREATE OR REPLACE TYPE Curso AS OBJECT (
    nombre VARCHAR2(50),
    fecha  DATE,
    alumnos Alumno
);
/
CREATE TABLE cursos OF Curso (
    nombre PRIMARY KEY
) NESTED TABLE alumnos STORE AS alumnos_tab;
/
INSERT INTO cursos VALUES (Curso('Informatica',null,Alumno((SELECT REF(e) FROM personas e))));
```

Imagen 12. Uso de tablas anidadas

www.universae.com