|  |  |  |  |
| --- | --- | --- | --- |
| **logo_udlap**    **EDEI** | **Manejadores de Bases de Datos**  **Trabajo Práctico IV** | | |
| **DEPARTAMENTO** | Computación, Electrónica y Mecatrónica | **MATERIA** | LIS – 3071 |
| **PROFESOR** | José Luis Zechinelli Martini | **PERIODO** | Primavera 2019 |

## Bases de datos objeto-relacional

Queremos modelar e implementar el siguiente escenario. Consideremos empleados que pueden ser Trabajadores con una fecha de contrato y Administrativos con un atributo salario. Los trabajadores trabajan en muchos Proyectos identificados por un nombre de proyecto. Cada proyecto puede ser desarrollado por varios trabajadores. Además, un proyecto es administrado por un administrador (de tipo Administrativos). Un administrador que tiene varios números de teléfono puede administrar proyectos. A continuación describiremos el esquema correspondiente.

**Esquema objeto-relacional**

Primero creamos tipos objeto para cada una de las entidades dadas. Los tipos que pueden ser especializados deben ser declarados como NOT FINAL.

CREATE TYPE employee\_t AS OBJECT (name varchar2(100)) NOT FINAL;

CREATE TYPE worker\_t UNDER employee\_t (contractDate date);

CREATE TYPE telephone\_t AS TABLE OF varchar2(30);

CREATE TYPE manager\_t UNDER employee\_t (salary integer, telephone telephone\_t);

CREATE TYPE workers\_t AS TABLE OF REF worker\_t;

CREATE TYPE project\_t AS OBJECT (

name varchar2(100),

managedBy REF manager\_t,

workers workers\_t

);

Como pueden ver, decidimos modelar la relación  manages por la referencia managedBy en la clase Project\_t e implementar la relación worksOn listando una colección (realizada como una tabla anidada) de workers en un objeto Project\_t. Noten que – como Oracle no soporta la actualización automática de relaciones inversas – decidimos realizar ligas entre objetos como una lista de patrones ligados sobre un lado solamente. Ahora creamos tablas para almacenar objetos de los tipos presentados.

CREATE TABLE worker OF worker\_t;

CREATE TABLE manager OF manager\_t NESTED TABLE telephone STORE AS telephone\_nt;

CREATE TABLE project OF project\_t NESTED TABLE workers STORE AS workers\_nt;

En Oracle debe darse explícitamente el nombre de la tabla anidada (telephone\_nt) que almacena datos de tipos colección.

ALTER TABLE project ADD (SCOPE FOR (managedBy) IS manager);

ALTER TABLE workers\_nt ADD (SCOPE FOR (COLUMN\_VALUE) IS worker);

En suma, para la definición de una tabla, el rango de la referencia válida está restringido. Entonces workers\_t es justamente una tabla de objetos worker\_t y el contendor del nombre del atributo debe ser declarado como COLUMN\_VALUE.

Noten que Oracle no garantiza la integridad referencial entre las referencias de objetos y sus instancias. Por lo tanto, podrían generarse referencias vacías, por ejemplo si borran a un administrador usado en un objeto de tipo Project.

**Inserción de datos**

* Dos empleados y un administrador:

INSERT INTO worker VALUES ('John Brown', cast ('19-02-2002' as date));

INSERT INTO worker VALUES (

worker\_t(' Jim Smith', cast ('11-02-2002' as date)));

INSERT INTO manager VALUES (

manager\_t('Big Boss', 60000,telephone\_t('324-32', '321-32')));

* Un proyecto donde nadie trabaja, pero que es administrado por *Big Boss*:

INSERT INTO project VALUES (

'The big nothing',

(SELECT REF(m) FROM manager m WHERE m.name='Big Boss'),

Workers\_t() );

* Asignar a los dos trabajadores creados a este proyecto:

INSERT INTO TABLE

(SELECT workers FROM project WHERE name='The big nothing')

(SELECT REF(w) FROM worker w

WHERE w.name='John Brown' OR w.name='Jim Smith');

* Otro proyecto administrado por *Big Boss* donde sólo trabaja *John Brown*:

INSERT INTO project VALUES (

project\_t('Small project',

(SELECT REF(m) FROM manager m WHERE m.name='Big Boss'),

Workers\_t( (SELECT REF(w) FROM worker w WHERE w.name='John Brown') ) ));

**Consulta de datos**

* SELECT \* FROM worker

Despliega todos los atributos de tipo worker\_t de todos los objetos de la tabla worker. El resultado parece una tabla clásica.

* SELECT value(w) FROM worker w;

Despliega lo mismo pero presenta los objetos en el formato worker\_t(name, date) (formato constructor).

* SELECT p.name, DEREF(p.managedBy) FROM project p;

Presenta el nombre de todos los proyectos y sus administradores en formato constructor.

* SELECT p.name, DEREF(p.managedBy).name FROM project p;

Presenta todos los nombres de los proyectos junto con el nombre del administrador.

* SELECT p.name, DEREF(w.COLUMN\_VALUE) FROM project p, TABLE(p.workers) w;

Desanida a los trabajadores asociados a cada proyecto y presenta su nombre con los objetos trabajador en formato constructor.

**Algunos tips útiles para su trabajo práctico con Oracle**

* SELECT \* FROM USER\_TYPES;

Da la lista de todos tipos de datos definidos por el usuario (diccionario de datos).

* SELECT \* FROM USER\_ALL\_TABLES WHERE TABLE\_TYPE IS NOT NULL;

Muestra todos los objetos de tipo tabla.

* DROP TYPE employee;

Suprime el tipo Empleado del esquema. El mensaje “Warning: El tipo fue compilado con error” indica un error durante la creación del tipo. Elimina el tipo y corrige el error.