GESTIÓN DE BASES DE DATOS

TIPOS DE DATOS OBJETO

- Un tipo de dato objeto es aquel que define una estructura y un comportamiento comunes para un conjunto de datos.
- La estructura de un tipo de dato objeto consta de dos partes:
  - La especificación: declaración de las estructuras de datos o conjunto de atributos y métodos necesarios para manipular los datos. (públicas)
  - El cuerpo: implementación de la especificación, definición de los métodos. (privadas)

Los tipos de objeto se pueden interpretar como plantillas para los objetos de cada tipo.

COMPONENTES DEL TIPO DE OBJETO

- Atributo
  - Ha de ser único dentro del tipo de objeto. Se declara a partir de un nombre y un tipo de dato.
  - Un tipo de dato para un atributo puede ser otro tipo de objeto.
  - Ejemplo:

Tipo **Direccion**

string: calle,
string: ciudad,
string: codigoPostal

Tipo Cliente

string: nombre,
int: codCliente,
Direccion: direccionCl

COMPONENTES DEL TIPO DE OBJETO

- Método
  - No puede tener el mismo nombre que el tipo de objeto ni el de ninguno de sus atributos.
  - La especificación es el nombre del método junto con una lista opcional de parámetros de entrada y en el caso de las funciones el parámetro de salida.
  - El cuerpo hace referencia al código que se ejecuta para llevar a cabo una operación determinada.

COMPONENTES DEL TIPO DE OBJETO

- Sobrecarga de métodos
  - Para que se dé la situación debemos encontrarnos con el mismo nombre para diversos métodos con diferencias en los parámetros formales ya sea en número de parámetros requeridos en la llamada al métodos como en su orden o tipo de dato.

  - La sobrecarga de métodos no será posible si:
    - Si las funciones se diferencian solo en el tipo de retorno.
    - Si los parámetros solo se diferencian en el modo.

Definición del tipo de objeto

- Teniendo en cuenta que el gestor de bases de datos seleccionado sólo da soporte parcial a la orientación a objetos, hay algunos conceptos que no se podrán desarrollar, como la definición de métodos en la especificación de un tipo.

- Esta parte la veremos de manera teórica.

TABLAS DE OBJETOS

- Una tabla de objetos es una clase espacial de tabla que almacena un objeto por cada fila, y el acceso a los atributos de los objetos se hace mediante las columnas de la tabla.
  - Ejemplo

```
Create type Comprador as(
        nombre varchar(20),
        codCliente varchar(20),
        mail varchar(20)
);
```

```
Create table Cliente of Comprador(
        Primary key (codCliente),
         nombre with options default 10,
        mail with options default 10
)with oids;
```

Cuando creamos el tipo comprador no se aceptan atributos tipo serial. La clave primaria ya se indica al crear la tabla.

TABLAS DE OBJETOS

- Cuando creamos una tabla a partir de un tipo ésta toma la estructura del tipo y hace que las columnas queden determinadas por los atributos del tipo. Aun así al crear la tabla podemos añadir valores y restricciones de tabla.
- La tabla está ligada al tipo de manera que si se elimina el tipo también se elimina la tabla asociada.
  - Ejemplo

    Drop type nombreTipo cascade;

TABLAS DE OBJETOS

- Al hacer una consulta a la tabla del ejemplo anterior podemos ver como los atributos de la tabla son los atributos del tipo que hemos creado con anterioridad.

```
prueba=# select * from cliente;
   nombre    | codcliente |        mail
-------------+------------+--------------------
 jose perez  | az001      | j.perez@gmail.com
(1 fila)
```

HERENCIA

- El tema de la herencia en PostgreSQL se trata a partir de los OIDS.
- Al crear una tabla debemos añadir la clausula with oids para que al crearse la tabla se cree una columna oculta con un identificador de objeto.
  - Ejemplo:

```
Create table Persona(
        codPersona varchar(5),
        nombre varchar (20),
        direccion varchar (20),
        Primary key (codPersona)
)with oids;
```

HERENCIA

- De manera que a partir de la tabla anterior podemos crear otra tabla derivada de Persona.
  - Ejemplo:

```
Create table Estudiante(
        carrera varchar(25),
        gupo varchar (20),
        especialidad varchar (20)
)inherits (persona);
```

HERENCIA

- Si ejecutamos los ejemplos anteriores en PostgreSQL vemos como al hacer la consulta sobre los datos de la tabla estudiante aparecen los atributos de la tabla persona.

```
prueba=# create table persona (codPersona varchar (5),
prueba(# nombre varchar (20),
prueba(# direccion varchar(20),
prueba(# primary key (codPersona))with oids;
CREATE TABLE
prueba=# create table estudante(
prueba(# carrera varchar (25),
prueba(# grupo varchar(20),
prueba(# especialidad varchar(20))inherits (persona);
CREATE TABLE
prueba=# select * from estudiante;
ERROR:  no existe la relación «estudiante»
LÃ?NEA 1: select * from estudiante;
                      ^
prueba=# select * from estudante;
 codpersona | nombre | direccion | carrera | grupo | especialidad
------------+--------+-----------+---------+-------+--------------
(0 filas)
```

HERENCIA

- La herencia no solo permite adquirir atributos sino que va más allá, se establece una relación conceptual.
  - Ejemplo:

```
prueba=# insert into estudante values ('EST01', 'lola', 'c/extremadura 8', 'inge
niero industrial', 'M03', 'mecanica');
INSERT 16508 1
prueba=# select * from estudante;
 codpersona | nombre |    direccion    |       carrera        | grupo | especial
idad
------------+--------+-----------------+----------------------+-------+---------
-----
 EST01      | lola   | c/extremadura 8 | ingeniero industrial | M03   | mecanica
(1 fila)


prueba=# select * from persona;
 codpersona | nombre |    direccion
------------+--------+-----------------
 EST01      | lola   | c/extremadura 8
(1 fila)
```

- La relación conceptual que se muestra es del tipo un estudiante es una persona. Por lo que al consultar cuantas personas hay en la tabla se incluye a los estudiantes.

HERENCIA

- No es posible borrar una tabla “padre” si no se borran primero las tablas “hijo”.
- Como es lógico al borrar una fila de una de las tabla s se borrará simultáneamente una fila de la otra tabla.
- En la herencia no solo se heredan los atributos sinó que también los OIDS.
  - Ejemplo:

```
prueba=# select oid, tableoid, * from estudante;
  oid  | tableoid | codpersona | nombre |    direccion    |       carrera
 | grupo | especialidad
-------+----------+------------+--------+-----------------+---------------------
-+-------+--------------
 16508 |    16505 | EST01      | lola   | c/extremadura 8 | ingeniero industrial
 | M03   | mecanica
(1 fila)

prueba=# select oid,tableoid, * from persona;
  oid  | tableoid | codpersona | nombre |    direccion
-------+----------+------------+--------+-----------------
 16508 |    16505 | EST01      | lola   | c/extremadura 8
(1 fila)
```

HERENCIA

- En bases de datos grandes no se recomienda el uso de OIDS, de manera que es preferible usar una secuencia compartida.
- Para elementos que no sean tipos se recomienda el uso de un serial de manera que tendremos un atributo autoincremental que podrán heredar los hijos de dicha tabla.

HERENCIA EN TABLAS DE OBJETOS

- A partir de un solo tipo podemos crear varias tablas para guardar información de los distintos trabajadores de una empresa.
  - Ejemplo:

```
Create type trabajador as (
        id int,
        nombre varchar(20),
        direccion varchar (20),
        mail varchar(30),
        telefono int
);
```

HERENCIA EN TABLAS DE OBJETOS

- La diferencia entre la primera y la segunda tabla es que la primera almacena objetos del tipo trabajador , y la segunda es una especialización de la tabla anterior.
- Es decir, la segunda tabla hereda las características de la primera y añade las suyas propias.

```
Create table trabajadores of trabajador(
        primary key (id)
);

Create table jefes  (
        departamento varchar(20)
)inherits (trabajadores);
```

HERENCIA EN TABLAS DE OBJETOS

- Vamos a insertar datos para verlo empíricamente:

```
prueba=# insert into trabajadores values (001, 'Rafael Dominguez', 'c/Toledo 10'
, 'rafdom@gmail.com', '936547812');
INSERT 0 1
prueba=# insert into jefes values (002, 'Pablo Drino', 'c/Luna nueva 15', 'padri
no@gmail.com', '912345678', 'ventas');
INSERT 0 1
prueba=# select * from trabajadores;
 id |      nombre      |    direccion     |       mail         | telefono
----+------------------+------------------+--------------------+-----------
  1 | Rafael Dominguez | c/Toledo 10      | rafdom@gmail.com   | 936547812
  2 | Pablo Drino      | c/Luna nueva 15  | padrino@gmail.com  | 912345678
(2 filas)


prueba=# select * from jefes;
 id |   nombre    |    direccion     |       mail         | telefono  | departamen
to
----+-------------+------------------+--------------------+-----------+-----------
---
  2 | Pablo Drino | c/Luna nueva 15  | padrino@gmail.com  | 912345678 | ventas
(1 fila)
```

PARA SABER UN POCO MÁS...

- Base de dades en PostgreSQL. UOC. Marc Gibert Ginestà i Óscar Pérez Mora.
- Bases de dades objecte-relacional. IOC. Aina del Tio Esteve

GRACIAS POR VUESTRA ATENCIÓN