PRACTICA 10
OBJETIVO: Conocer y practicar las cardinalidades
DESCRIPCIÓN:

Crea un modelo

Atributos del modelo: las clases del modelo pueden usar algunos atributos para controlar parte de su comportamiento:

_name: El identificador interno para crear el modelo odoo, que es obligatorio.

_description: cuando la interfaz de usuario muestra el modelo, un título de registro de modelo fácil de usar.

_order: establece el orden predeterminado al examinar los registros del modelo o mostrarlos en la vista de lista.

_rec_name: se utiliza para señalar los campos que hacen referencia al registro de descripción de la palabra relacionada, como una relación de varios a uno. De forma predeterminada, utiliza el campo de nombre, que es un campo común en el modelo. Pero este atributo nos permite utilizar cualquier otro campo para este propósito.

_table: el nombre de la tabla de la base de datos utilizada para respaldar el modelo. Por lo general, se calcula automáticamente a la izquierda y el nombre del modelo se reemplaza por un guión bajo, pero también se puede establecer un nombre de tabla específico.

_herit: herencia.

_inherits: herencia incrustada.

Relación entre modelos

Al nombrar campos, existe la siguiente convención: Los campos nombrados con _id y _ids al final corresponden a la relación entre 2one y 2many respectivamente.

Relación de varios a uno: la relación de Many2one requiere dos parámetros: módulo asociado y carácter de visualización. Creará un campo con una clave externa en la tabla de la base de datos para asociar la tabla de la base de datos. Otros parámetros:

ondelete se activará cuando se elimine el registro asociado. El valor predeterminado es nulo, es decir, cuando se elimina el registro asociado, estará en blanco. Otros valores son restringidos, cuando se asocian registros, se prohíbe eliminar, en cascada, eliminar el registro actual mientras se elimina el registro asociado.

context es un diccionario de datos. En la vista del cliente web, cuando se accede a través de relaciones de asociación, se pasa el contexto. Por ejemplo, establezca el valor predeterminado.

El dominio es una expresión de dominio de una lista de múltiples tuplas, que se utiliza para eliminar registros válidos en el campo asociado.

auto_join = True Después de usar este parámetro, permitirá que ORM use la función de unión SQL (empalme, cascada) para la consulta de datos.

LAS RELACIONES

Relación de muchos a muchos: Many2many debe proporcionar al menos un parámetro, que es el módulo asociado, y se recomienda utilizar el parámetro de cadena para una mejor visualización del título. Almacenamiento de relaciones de varios a varios: en el nivel de la base de datos, no se agregan campos a la tabla de la base de datos. Automáticamente creará una nueva tabla intermedia, esta tabla tiene solo dos campos de ID de clave externa, y estos dos campos están asociados respectivamente con la tabla de base de datos correspondiente.

Relación inversa de uno a muchos: One2many recibe 3 parámetros en orden: el módulo asociado, el nombre del campo del módulo asociado y el texto del título. Los dos primeros parámetros suelen ser el nombre del módulo y el nombre del campo de clave externa correspondiente a la relación opuesta.

PASO 1: Las diferentes cardinalidades que existen en ODOO

one2many (de uno a muchos).

Una relación de uno a muchos sería por ejemplo la relación entre autores y libros. Un autor puede tener escritos varios libros, y varios libros pueden pertenecer a un autor. La relación sería de 1 a N.

Ejemplo de definición de la clase autor:

```
class mybookstore_author(osv.osv):
   _name ='mybookstore.author'
   _description ='Autores'
   _columns= {
      'name': fields.char('Name', size=64, required="True"),
      'active': fields.boolean('Active'),
      'book_ids': fields.one2many('mybookstore.book', 'author_id', 'Books'),
   }

mybookstore_author()
```

• **'book_ids'**: Nombre de campo que usamos para relacionarlo con la tabla libro.
Como en este campo va a haber muchos Ids de libros, el nombre del campo acaba en "_ids". Como norma el nombre del campo será <objeto>_ids.

• **'mybookstore.book'**: Nombre del objeto con el cuál queremos relacionar la tabla.

• **'autor_id':** IMPORTANTE, es el nombre del campo que hemos puesto en la tabla libro para relacionarla con la tabla autor. Para ver esto, se ha puesto en color rojo en los dos ejemplos de definición de las tablas autores, y libros.

• **'Books':** Literal que mostrará el campo.

many2one (de muchos a uno).

Sería la contrapartida del punto anterior. En este caso estamos hablando de que muchos libros, pertenecen a un autor.
Ejemplo de definición de la clase libro:

```
class mybookstore_book(osv.osv):
  _name ='mybookstore.book'
  _description ='Libros'
  _columns= {
    'name': fields.char('Name', size=64, required="True"),
    'pagenumber': fields.integer('Page Number'),
    'author_id': fields.many2one('mybookstore.author', 'Author'),
  }
mybookstore_book()
```

• **'autor_id':** Nombre de campo que usamos para relacionarlo con la tabla autor.
Como en este campo va a haber un id de autor, el nombre de campo acaba en "_id". Como norma el nombre del campo será <objeto>_id.

• **'Author'**: Literal que mostrará el campo.

many2many (de muchos a muchos).

Una relación de muchos a muchos sería por ejemplo la relación entre cantantes y canciones. Un cantante puede cantar varias canciones, y una canción puede ser cantada por varios cantantes. La relación sería de N a N.

Ejemplo de definición de la clase cantante:

```
class singer(osv.osv):
    _name = 'singer'
    _description = 'Singers'

    _columns = {
            'name':fields.char('Name', size=64, required=True),

#############################################################
#            EXPLICACIÓN DE LOS PARAMETROS DE many2many
#            'songs': Objeto con el cual relaciono el objeto cantante
#            'auth_songs_rel': Nombre que pongo a la relación de las 2 tablas
#            'author_id': ID del campo de la tabla actual.
```

```
#            'songs_id': ID del campo de la tabla que quiero relacionar.
#            'Songs': Literal que mostrará el campo
#################################################################

           'songs_ids' : fields.many2many('songs','auth_songs_rel','author_id','songs_id','Songs'),

            'active': fields.boolean('Active'),
            'song_ids' : fields.one2many('song', 'author', 'Singing songs'),
        }

author()
```

PASO 2: Agregar una tabla con relación one2many

```
from odoo import models, fields

class Libro(models.Model):
    _name = 'biblioteca.libro'
    _description= 'Modelo para representar los libros de una biblioteca'
    name=fields.Char('Título', required=True)
    date_release = fields.Date('Fecha de publicación')
    sumario = fields.Text(string='Resumen')
    pages = fields.Integer('Páginas')
    imagen = fields.Binary(string='Imagen')
    discontinued = fields.Boolean(string='Descatalogado', readonly=True)
    language = fields.Selection([('Es', 'Español'),('In', 'Inglés'), ('De',
'Alemán'), ('Fr', 'Francés')], string='Idioma', default='Es')
    autor_ids =
fields.Many2many('biblioteca.autores','biblioteca.editores',string='Autores')
#    editorial_ids = fields.one2many('biblioteca.editores','editorial'
,string='Editores')

class Autores(models.Model):
    _name = 'biblioteca.autores'
    _description = 'modelo para representar los autores de cada libro'
    name = fields.Char('Nombre', required=True)

class Editores(models.Model):
    _name = 'biblioteca.editores'
    _description = 'modelo para representar los editores de cada libro'
    editorial = fields.Char('Nombre', size=65, required=True)
    direccion = fields.Char('Dirección', required=True)
    codigopostal = fields.Integer('Codigo Postal')
    poblacion = fields.Char('Poblacion')
#    libro_ids = fields.many2one('biblioteca.libro',string='libro')
```