**1. Many2one (Many-to-One)**

* Represents a **many-to-one** relationship.
* Many records of the current model can be linked to one record of another model.
* Uses the comodel\_name parameter to specify the related model and inverse\_name to define the reverse relation.

**Full Parameters:**

* comodel\_name: The model to which the current model is related.
* string: The label that will appear in the UI.
* ondelete: Defines what happens when the related record is deleted. Possible values:
  + cascade: Deletes the current record if the related record is deleted.
  + set null: Sets the relation to NULL when the related record is deleted.
  + restrict: Prevents deletion of the related record if there are existing relations.
* domain: Optional filter to limit available records in the drop-down.
* help: A tooltip to describe the field in the UI.
* required: Marks the field as mandatory.

**Example:**

from odoo import models, fields

class SaleOrder(models.Model):

\_name = "sale.order"

\_description = "Sale Order"

customer\_id = fields.Many2one(

comodel\_name="res.partner",

string="Customer",

ondelete="cascade",

help="The customer for this order",

required=True

)

* Here, the field customer\_id references a record in the res.partner model.

**2. One2many (One-to-Many)**

* Represents a **one-to-many** relationship.
* One record of the current model can be linked to multiple records of another model.
* This is a **virtual field**, and the reverse relation is defined using the Many2one field in the related model.

**Full Parameters:**

* comodel\_name: The related model (the model on the "many" side of the relation).
* inverse\_name: The field in the related model that points back to the current model. This should be a Many2one field.
* string: The label that will appear in the UI.
* domain: Optional filter to limit available records in the list.
* limit: Limits the number of records displayed.

**Example:**

class SaleOrder(models.Model):

\_name = "sale.order"

order\_lines = fields.One2many(

comodel\_name="sale.order.line",

inverse\_name="order\_id",

string="Order Lines"

)

class SaleOrderLine(models.Model):

\_name = "sale.order.line"

order\_id = fields.Many2one(

comodel\_name="sale.order",

string="Sale Order"

)

product\_id = fields.Many2one(

comodel\_name="product.product",

string="Product"

)

* order\_lines in sale.order refers to multiple sale.order.line records through the order\_id field.

**3. Many2many (Many-to-Many)**

* Represents a **many-to-many** relationship.
* Many records of the current model can be linked to many records of another model.
* Odoo automatically manages an intermediate table to store the relationships.

**Full Parameters:**

* comodel\_name: The model to which the current model is related.
* relation: The name of the intermediate table used to store the many-to-many relations (Odoo creates this automatically if not specified).
* column1: The name of the field that links to the current model.
* column2: The name of the field that links to the related model.
* string: The label that will appear in the UI.
* domain: Optional filter to limit available records in the list.
* context: Context that applies to the field (e.g., to filter based on user role).
* required: Marks the field as mandatory.

**Example:**

class SaleOrder(models.Model):

\_name = "sale.order"

tags = fields.Many2many(

comodel\_name="sale.order.tag",

relation="sale\_order\_tag\_rel",

column1="order\_id",

column2="tag\_id",

string="Tags"

)

class SaleOrderTag(models.Model):

\_name = "sale.order.tag"

name = fields.Char(string="Tag Name")

* The many-to-many relationship is stored in the sale\_order\_tag\_rel intermediate table, with order\_id and tag\_id linking the two models.

**Summary Table with Full Parameters:**

| **Field Type** | **Relationship** | **Full Parameters** |
| --- | --- | --- |
| Many2one | Many-to-One | comodel\_name, string, ondelete, domain, help, required |
| One2many | One-to-Many | comodel\_name, inverse\_name, string, domain, limit |
| Many2many | Many-to-Many | comodel\_name, relation, column1, column2, string, domain, context, required |

**How foreign keys are stored and used in Odoo's relational fields.**

**1. Many2one (Foreign Key Stored in the Many2one Model)**

In this relationship, the **Many2one** field stores the foreign key in the model that defines the Many2one field. In your case, the sale.order model stores the customer\_id, which points to the res.partner (Customer) model.

**Example:**

class SaleOrder(models.Model):

\_name = "sale.order"

customer\_id = fields.Many2one("res.partner", string="Customer")

**Data Structure:**

| **sale\_order.id** | **order\_number** | **customer\_id (FK to res\_partner.id)** |
| --- | --- | --- |
| 1 | SO001 | 1 |
| 2 | SO002 | 2 |

* The sale\_order table will have a column customer\_id storing the foreign key referencing res.partner.id.

**2. One2many (Foreign Key Stored in the Related Model)**

The **One2many** field in Odoo is the inverse of the Many2one field. The foreign key is stored in the **related model**, not in the model that holds the One2many field. Here, the sale.order model doesn't directly store the foreign key, but the res.partner model (which holds the Many2one field) indirectly links to multiple sale.order records.

**Example:**

class SaleOrder(models.Model):

\_name = "sale.order"

customer\_id = fields.Many2one("res.partner", string="Customer")

class ResPartner(models.Model):

\_name = "res.partner"

sale\_order\_ids = fields.One2many("sale.order", "customer\_id", string="Sales Orders")

**Data Structure:**

| **sale\_order.id** | **order\_number** | **customer\_id (FK to res\_partner.id)** |
| --- | --- | --- |
| 1 | SO001 | 1 |
| 2 | SO002 | 1 |
| 3 | SO003 | 2 |

* The sale\_order table stores the customer\_id, but the relationship is one-way: a res.partner (Customer) record can have many associated sale.order records.

**3. Many2many (Foreign Keys Stored in the Pivot Table)**

For the **Many2many** relationship, Odoo automatically creates a **pivot table** to manage the relationship. The foreign keys for both models are stored in this pivot table, which links records from both models. In your case, the product\_tag\_rel table stores the product\_id and tag\_id, which are foreign keys pointing to the product.product and product.tag models, respectively.

**Example:**

class ProductProduct(models.Model):

\_name = "product.product"

tag\_ids = fields.Many2many("product.tag", string="Tags")

class ProductTag(models.Model):

\_name = "product.tag"

product\_ids = fields.Many2many("product.product", string="Products")

**Data Structure:** **Pivot Table (product\_tag\_rel):**

| **product\_id (FK to product\_product.id)** | **tag\_id (FK to product\_tag.id)** |
| --- | --- |
| 1 | 1 |
| 2 | 1 |
| 2 | 2 |

* The product\_tag\_rel table is the pivot table where the foreign keys are stored, linking products to tags. For example, product with ID 2 is associated with both tag 1 and tag 2.

**Summary:**

* **Many2one**: The foreign key is stored in the model where the Many2one field is defined (e.g., sale.order stores customer\_id as a foreign key to res.partner).
* **One2many**: The foreign key is stored in the related model (e.g., sale.order has a Many2one to res.partner, so the foreign key is stored in sale.order).
* **Many2many**: A pivot table stores foreign keys to link the two models (e.g., product\_tag\_rel links product.product and product.tag).