

## SISTEMAS DE GESTIÓN EMPRESARIAL

### TÉCNICO EN DESARROLLO DE APLICACIONES MULTIPLATAFORMA

# **Bases de datos con PgAdmin**

## **Conexión a base de datos de Odoo**

## **Tablas, vistas, campos y relaciones de tablas en Odoo**

## **Informes y gráficos**

## **Consultas SQL en PgAdmin**

## **Modificación de valores desde PgAdmin**

## **Backups y restauraciones**

*Vamos a ver una herramienta que nos permita gestionar la base de datos de Odoo, Open Source, con una interfaz amigable, pero a la vez potente. Nos permitirá realizar backups, ver informes de la base de datos, modificar datos directamente en la base de datos, etc.*

## La base de datos y PgAdmin

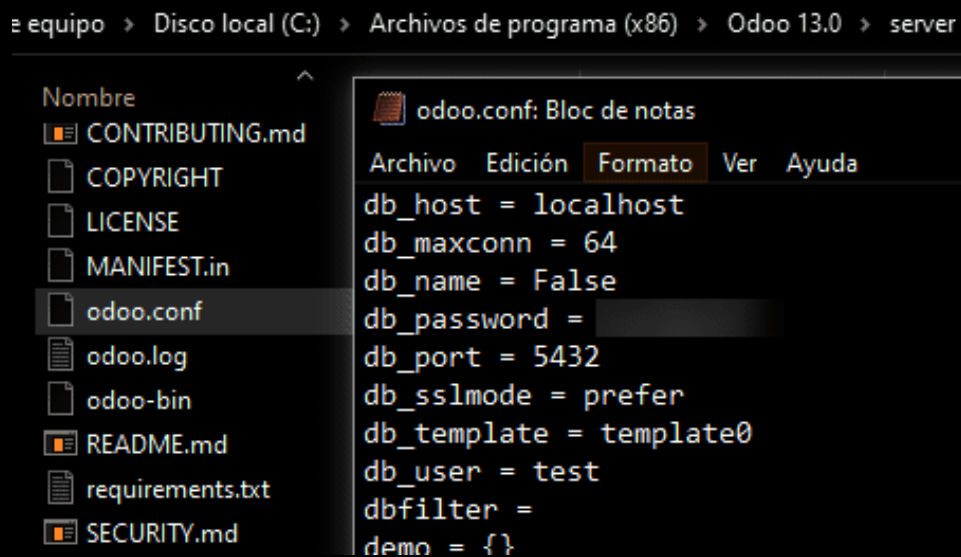
Odoo necesita un sistema de base de datos PostgreSQL para funcionar.

PostgreSQL es un potente sistema de base de datos objeto-relacional de código abierto que usa y amplía el lenguaje SQL.

Al instalar Odoo con el instalador en Windows, éste ya instalaba y configuraba PostgreSQL para su uso. Pero tenemos que aprender a configurar la conexión con la base de datos y saber cómo funciona, por si elegimos otro sistema operativo, otro modo de instalación, necesitamos crear roles, conectarnos a otra base de datos, hacer backups, por si surgen problemas, etc.

Para empezar, en el **archivo de configuración** de Odoo (**odoo.conf**) podemos ver como se está conectando actualmente Odoo a la BD (**host, usuario, contraseña, puerto**, etc.):

fichero odoo.conf

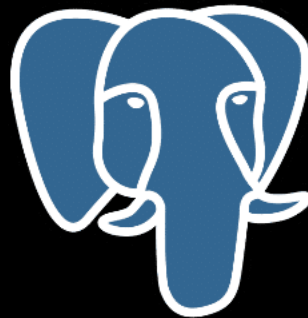


Para seguir conociendo como configurar PostgreSQL, lo haremos a través de la herramienta PgAdmin.

## PgAdmin

PgAdmin es una aplicación muy popular que se usa tanto para la **gestión** como para la **administración** y el **desarrollo de bases de datos en PostgreSQL**.

Esta herramienta es también Open Source y multiplataforma. Probablemente se haya **instalado junto con PostgreSQL** (si instalamos en Windows), pero si no, bastará con descargarla e instalarla desde su web oficial. En nuestro caso usaremos la versión más actual (la 4). En enero de 2024 la actual es la 7.

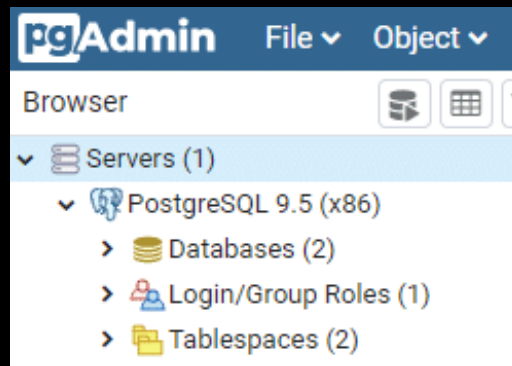


## La conexión en PgAdmin

Al abrir PgAdmin por primera vez, si ya tenemos Odoo instalado, nos pedirá la contraseña para el usuario (en este caso db\_user = test), que es la que vimos en la imagen del fichero odoo.conf mas arriba.

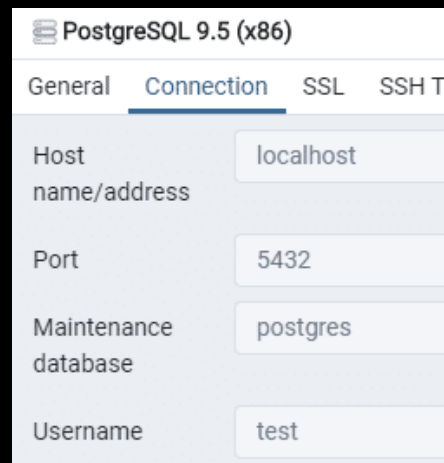
Una vez dentro, podremos apreciar que existe un servidor 'PostgreSQL 9.5 (x86)' (en enero 2024 el servidor será PostgreSQL 12), en cuyo interior hay **2 bases de datos**, un **usuario** de login y **2 tablespaces**:

Panel principal de PgAdmin



Si pulsamos con el **botón derecho** sobre el **servidor** 'PostgreSQL 9.5' y abrimos sus **propiedades**, podremos comprobar en la **pestaña de conexión** que se trata de la **misma configuración del fichero odoo.conf**:

Configuración de la conexión del servidor en PgAdmin



De esta forma comprobamos que **PgAdmin** está **conectado a la base de datos** con la que actualmente estamos trabajando **en Odoo**.

Al haber conectado **PgAdmin** con el servidor que usa Odoo, comentar que los cambios que se realicen sobre esta herramienta **afectarán directamente a Odoo**. Es decir, desde PgAdmin **podremos**, por ejemplo, hacer que un **campo** de una tabla sea obligatorio, **crear/modificar/eliminar registros** de una tabla, crear/modificar/eliminar **tablas**, etc. y todo esto tendrá la **misma repercusión** que si lo hiciésemos **directamente en Odoo**.

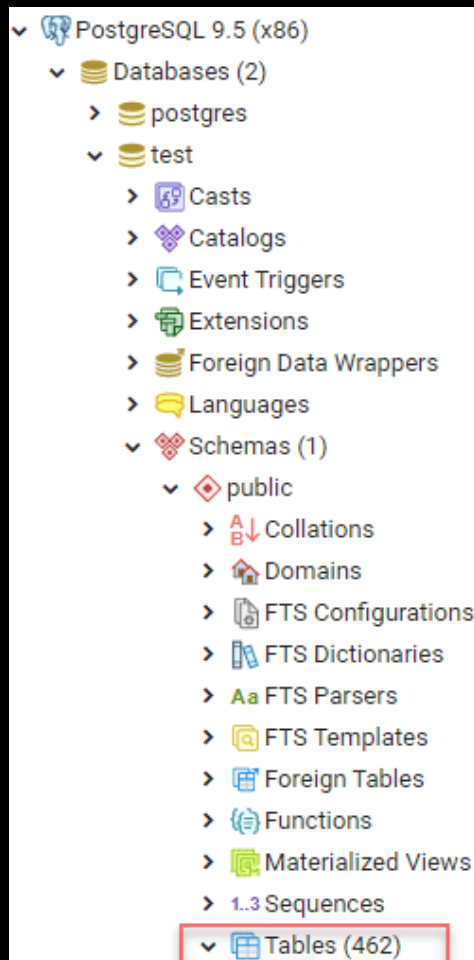
## Tablas, vistas y campos de la base de datos

A continuación, vamos a ver las **tablas y las vistas** (con sus respectivos **campos**) que configuran la **base de datos de Odoo**.

Para **ver las tablas** en **PgAdmin** deberemos desplegar la siguiente **ruta del árbol**:

→ **servidor / base de datos / schemas / public / tables**.

Desde PgAdmin bastará con **desplegar el árbol** que se muestra en la siguiente imagen:



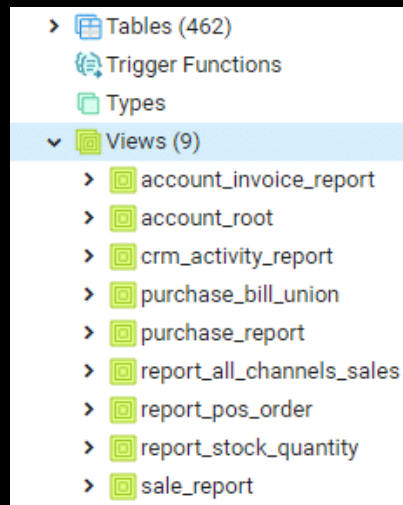
Podemos ver que disponemos de **462 tablas** (454 en la versión 17) en la base de datos.

Seguidamente, si desplegamos la **sección 'tablas'**, podemos ver cada una de ellas con sus **características, propiedades, columnas**, etc.

Por ejemplo, en la tabla `product_template` nos encontramos con 45 columnas, 10 restricciones o constraints y 3 índices.

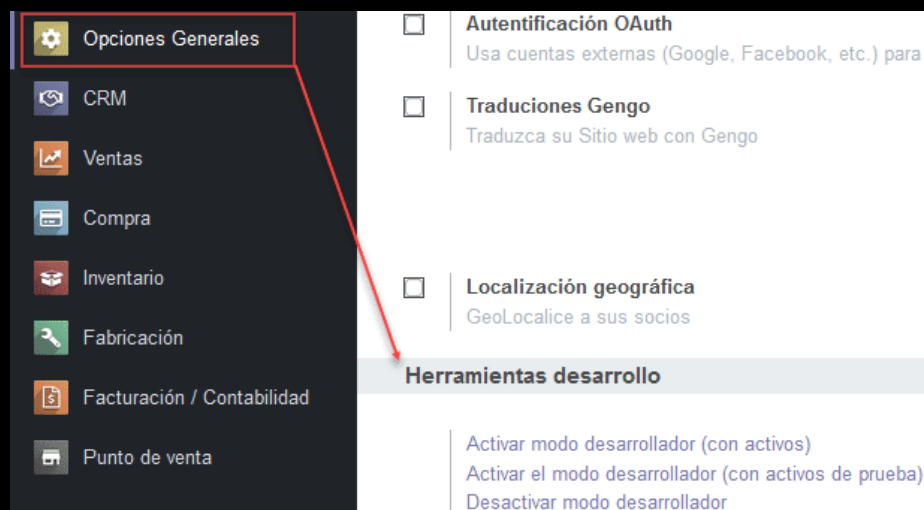
Al **seleccionar la tabla**, alguna de sus columnas o cualquier objeto, en la derecha, podremos ver sus **propiedades**, su **declaración SQL**, sus **estadísticas** (cuantas veces se ha leído, el tamaño de la tabla, cuantas veces se han insertado registros, etc.) y las **dependencias**.

Además de las tablas y sus campos también podremos analizar las **'vistas'** que se van **generando, junto con sus campos**.

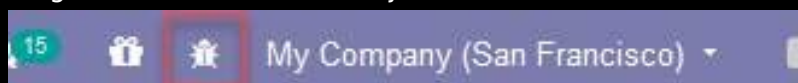


## Tablas, vistas y campos en Odoo

Para poder visualizar información técnica en Odoo, deberemos entrar en el **modo desarrollador** de la herramienta. Para ello tendremos que entrar en las **opciones generales** de los ajustes de Odoo y pulsar sobre '**activar modo desarrollador (con activos)**'.



En ese momento, entre otras cosas, se nos habilitarán las herramientas '**Open Developer Tools**' (icono en forma de bicho, igual al icono "debug" en otros IDEs) y muchas opciones en el menú de configuración del módulo de ajustes.



Y en Ajustes tendremos una nueva pestaña llamada “Técnico”.

## Viendo la base de datos desde Odoo

Una vez habilitado el modo desarrollador, ya podremos **visualizar las tablas, vistas y campos en Odoo**.

En esta pestaña “Técnico”, encontraremos “Estructura de la base de datos”:

- “**Precisión decimal**”, donde podremos modificar el número de decimales de un campo.
- “**Modelos**”, donde podremos ver:
  - los campos de cada tabla,
  - los permisos de acceso (de otras tablas para actualizar, eliminar sobre esta tabla),
  - reglas de registro,
  - notas,
  - vistas,
  - en qué módulos (aplicaciones) de Odoo se usa la tabla, es decir, qué módulos tienen acceso a dicha tabla;
- “**Campos**”, donde vemos todos los campos de todas las tablas. Buscamos el campo y a la derecha vemos el modelo (tabla) en el que se usa, y si pulsamos en el campo, vemos todas sus características:



<input type="checkbox"/>	Nombre de campo	Etiqueta de campo	Modelo	Tipo de campo	Tipo	Indexado	Almacenado	Sólo lectura	Relación del objeto
<input type="checkbox"/>	account_src_id	Cuenta de producto	Asignación de cuentas de Posici...	many2one	Campo ba...	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	account.account
<input type="checkbox"/>	attribute_line_ids	Atributos del producto	Producto	one2many	Campo ba...	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	product.template.attribute.line
<input type="checkbox"/>	attribute_line_ids	Atributos del producto	Plantilla de producto	one2many	Campo ba...	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	product.template.attribute.line
<input type="checkbox"/>	bom_product_template_attribute_...	Aplicar en variantes	Línea de Lista de Materiales	many2many	Campo ba...	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	product.template.attribute.val...
<input type="checkbox"/>	byproduct_id	Subproductos	Movimiento de existencias	many2one	Campo ba...	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	mrp.bom.byproduct
<input type="checkbox"/>	byproduct_ids	Subproductos	Lista de materiales	one2many	Campo ba...	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	mrp.bom.byproduct
<input type="checkbox"/>	categ_id	Categoría de producto	Capa de valoración de stock	many2one	Campo ba...	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	product.category
<input type="checkbox"/>	categ_id	Categoría de producto	Línea de plantilla	many2one	Campo ba...	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	product.category

- “**Selección de campos (Fields selection)**”, donde vemos las opciones de cada campo, por ejemplo si un producto es consumible, servicio o almacenable, y además se pueden volver a configurar desde aquí.
- “**Restricciones del modelo**”, por ejemplo la “f” nos indica que es una clave foránea, y con qué módulos se relacionan estas restricciones.
- “**Relaciones Many to Many**”.
- “**Adjuntos**” de imágenes, facturas... que también se almacenarán en una base de datos. Y podremos abrirlas y ver su contenido, ya que es un adjunto generado anteriormente.



Estas opciones y herramientas ayudarán a los desarrolladores a **trabajar con Odoo**. Si vamos a **trabajar técnica o funcionalmente con Odoo**, deberemos familiarizarnos con el **modo desarrollador**.

**Extraer informes y gráficos sobre la actividad del servidor en tiempo real**

Desde la versión 4 de PgAdmin es posible realizar una **monitorización en tiempo real** del **estado del servidor** de la base de datos haciendo uso del **Dashboard**. Para ello deberemos posicionarnos sobre la base de datos a analizar y pulsar sobre Dashboard. En ese momento aparecerán una serie de gráficas:

- La grafica “**Data sessions**”:  
muestra las sesiones de la base de datos:  
en verde las que **están activas**,  
en rojo las **desactivadas**,  
en azul las **totales**.
- La grafica “**transacciones por segundo**”:  
en azul las **transacciones**,  
en verde los **commits** (transacción para guardar),  
en rojo los **rollbacks** (transacción para deshacer).
- La grafica “**Tuples in**”:  
muestra las acciones de **modificaciones** sobre la base de datos:  
en azul las **inserciones**,  
en verde las **actualizaciones** ,  
en rojo las **eliminaciones**.
- La grafica “**Tuples out**”:  
muestra las lecturas a la base de datos:  
en azul las **lecturas**,  
en verde las **devoluciones de datos**.
- La grafica “**Block I/O**”:  
muestra los bloques de entrada/salida:  
en azul los de **salida**,  
en verde los de **entrada**.

Además, en la parte inferior aparece información sobre la **actividad del servidor**, pudiendo incluso **cerrar sesiones**, **cancelar queries**, etc.

## Actividad del servidor

Server activity									
Sessions		Locks	Prepared Transactions				Q	Search	
			PID	User	Application	Client	Backend start	State	Waiting?
✖	■	▶	3660	test		::1	2020-07-10 18:58:52 CEST	idle	no
✖	■	▶	3996	test		::1	2020-07-10 18:58:56 CEST	idle	no
✖	■	▶	5000	test		::1	2020-07-10 18:58:55 CEST	idle	no

**Usos de PgAdmin:**  
**Conexión Odoo y BBDD**

A continuación, veremos algunos usos que le podremos dar a **PgAdmin** y cómo nos ayudará con la **gestión de la BD de Odoo**.

**Importante:**

Otra de las opciones que hemos habilitado con el **modo desarrollador** es la ayuda que Odoo nos proporciona con la **información de los campos**. Ahora, al posicionarnos durante 1 o 2 segundos sobre, por ejemplo, un campo de un formulario, nos aparecerá un pop-up con información referente a dicho campo. Por ejemplo, si nos vamos al producto 'Mesa' que fabricamos en temas anteriores (módulo de fabricación / datos principales / productos / mesa) y posicionamos el cursor sobre el campo 'Precio de venta', **Odoo** nos indicará que se trata del campo 'list\_price' del **modelo 'product.template'** (en **PgAdmin** será la **tabla 'product\_template'**), que es de tipo 'float' y que se muestra como 'Monetary' (moneda):

Pop-up con información técnica del campo



Pero dicha tabla puede tener muchos registros. Para saber concretamente **qué registro es** concretamente la 'Mesa' cuyo valor 'list\_price' es de 330€ podemos **buscar el 'id'** de varias formas:

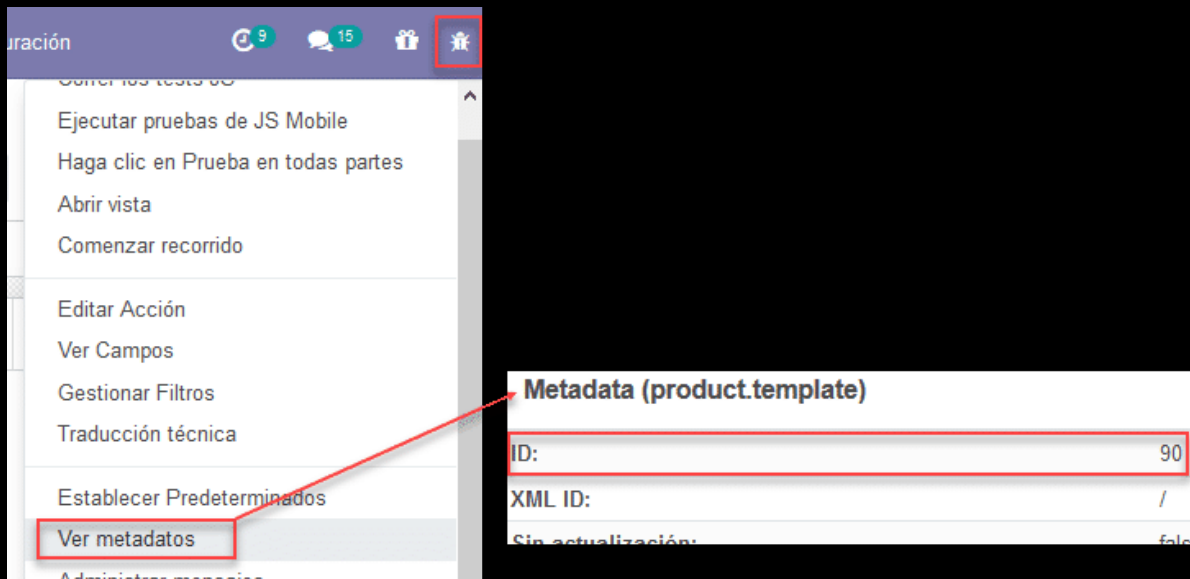
- En la **barra de dirección** (donde además podremos ver la tabla como "*model=*"):

Viendo el 'id' del producto en la barra de direcciones

localhost:8069/web?id=90&action=492&model=product.template&view\_type=form&cids=&menu\_id=313

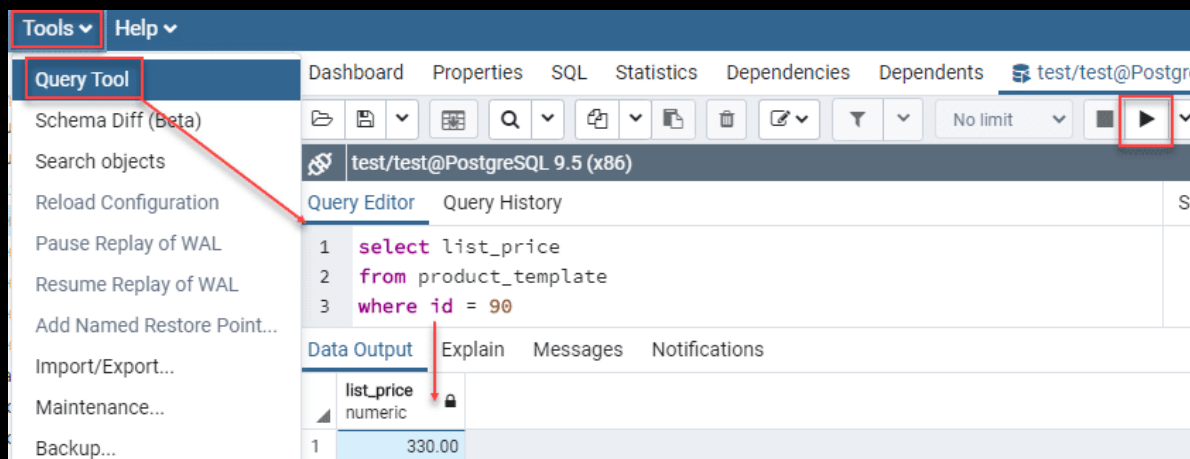
- O accediendo a los **metadatos** (pulsando el icono de **modo desarrollador** y yendo a "Ver metadatos"):

Viendo el 'id' del producto en los metadatos



Así, con la tabla (product\_template), el id (90) del registro y el nombre del campo (list\_price), podríamos realizar una **consulta de acceso a datos en PgAdmin** para buscar dicho valor (330.00):

Consulta de acceso a datos desde PgAdmin



Usos de Pgadmin

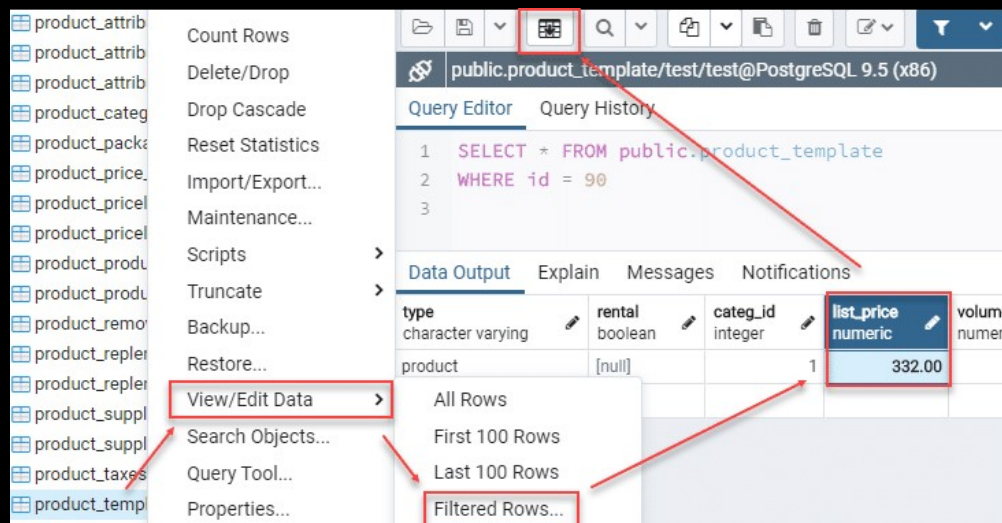
## Modificar valores

Los **cambios** que se realicen **en PgAdmin** **afectarán directamente a Odoo**. Aunque lo más fácil y recomendable sería hacerlo desde Odoo, hay veces que nos ayudará realizar la modificación directamente en la base de datos, por ejemplo, para **modificaciones masivas**.

Así, por ejemplo, si quisiéramos **modificar el valor del precio** del producto 'Mesa' **desde PgAdmin** podríamos hacerlo de varias formas:

### 1. **Actualizando el valor desde la tabla en PgAdmin** (filtrando por id=90):

Seleccionando la tabla, botón derecho se selecciona "View/Edit Data" → "Filtered Rows", y en el cuadro escribimos el filtro, en este caso: id = 90. Damos a "ok" y nos abre el resultado, ejecutando la sentencia completa pgAdmin, como se ve en la imagen. Y ya podemos buscar en la pestaña "Data Output" el campo "list\_price", que se podrá modificar pulsando sobre el cuadro de texto del precio actual. Y guardamos con el icono "Save data changes".



Podemos ver cómo en Odoo los cambios se ven reflejados al instante.

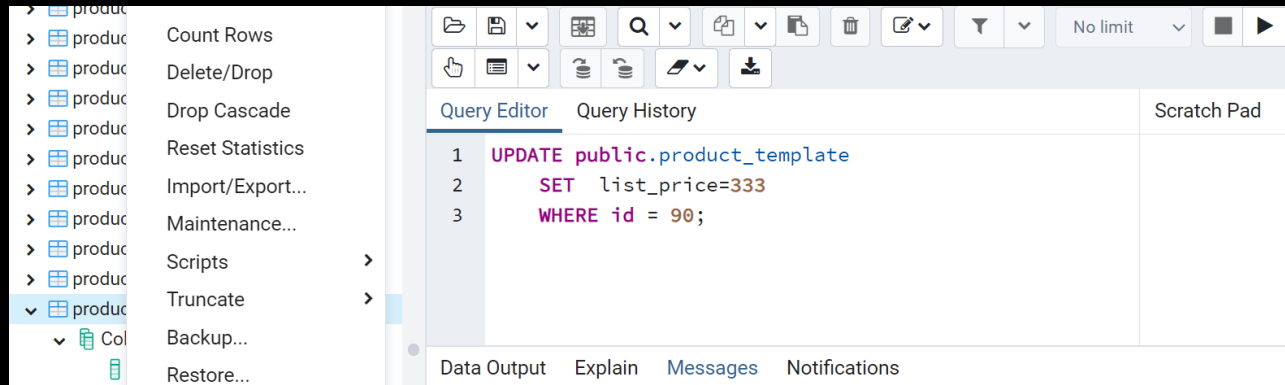
Comprobando en Odoo que se ha modificado el valor



## 2. Actualizando el valor mediante código SQL:

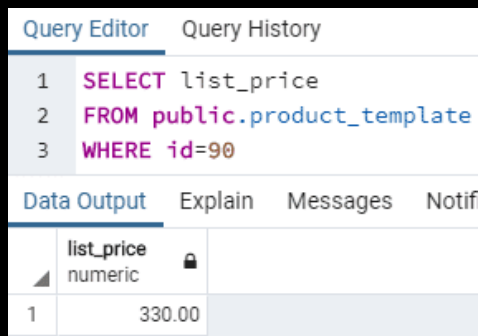
De la misma manera, pero “Scripts” → “UPDATE Script”, y modificando la sentencia SQL:

Actualizando un valor con una sentencia SQL en PgAdmin.



Si ahora, para volver al valor que tenía (330€), lo **modificamos desde Odoo**, podemos ver en **PgAdmin que éste se actualiza**:

Comprobar que ha cambiado en PgAdmin



## Las relaciones

En Odoo pueden darse relaciones:

**many2one:** tipo de campo que almacena una relación de **muchos (n) a uno (1)**),

**one2Many:** (tipo de campo que almacena una relación de **uno (1) a muchos (m)**),

**many2many:** (tipo de campo que almacena una relación de **muchos (m) a muchos (n)**).

Se pueden ver esas **relaciones en Odoo:**

Para ello es necesario entrar en el modo desarrollador. Por ejemplo, una relación one2many podría ser la de un **pedido de ventas** (one) y todos sus productos (many). Si vamos al módulo “Ventas” y elegimos un presupuesto, al entrar en **modo desarrollador** y posicionarnos sobre el **listado de productos** (etiqueta Producto sobre la lista), el **pop-up nos indica que es un tipo one2many de la tabla** (objeto-modelo) **sale.order** con la **tabla** (relación) **sale.order.line**.

The screenshot shows the Odoo Sales module interface. At the top, there's a navigation bar with 'Ventas' and several sub-menus: 'Pedidos', 'A facturar', 'Productos', 'Informes', and 'Configuración'. Below this, the main header indicates 'Presupuestos / S00007'. There are buttons for 'Editar', 'Crear', 'Imprimir', and 'Acción'. The main content area shows the budget details for 'S00007', including the client 'Gemini Furniture' and the template 'Plantilla de presupuesto'. A table titled 'Líneas del pedido' lists items with columns for 'Producto', 'Descripción', 'Cantidad', and 'Entregado'. A developer tool popup is visible over the table, showing the relationship details for the 'order\_line' field, including the object 'sale.order', type 'one2many', and the relationship 'sale.order.line'.

Producto	Descripción	Cantidad	Entregado
[FURN_6666] Pantallas de bloque acústi...	[FURN_6666] Acoustic Bloc Screens	5,000	0,000
[FURN_8999] Sofá de tres asientos	[FURN_8999] Three-Seat Sofa Three Seater Sofa with Lounger in Steel Grey Colour	1,000	0,000
[FURN_8888] Lámpara de oficina	[FURN_8888] Office Lamp	1,000	0,000

**Líneas del pedido**

- Field: order\_line
- Objeto: sale.order
- Tipo: one2many
- Widget: UnoAMuchos (section\_and\_note\_one2many)
- Contexto: {}
- Dominio: []
- Modificadores: {"readonly": ["state", "in", "done", "cancel"]}
- Al cambiar: 1
- Relación: sale.order.line



Esta relación podemos comprobar que existe en **PgAdmin** ya que si entramos en el **campo 'id'** de la tabla '**sale\_order**', vemos que existe una **foreign key** hacia la tabla '**sale\_order\_line**':

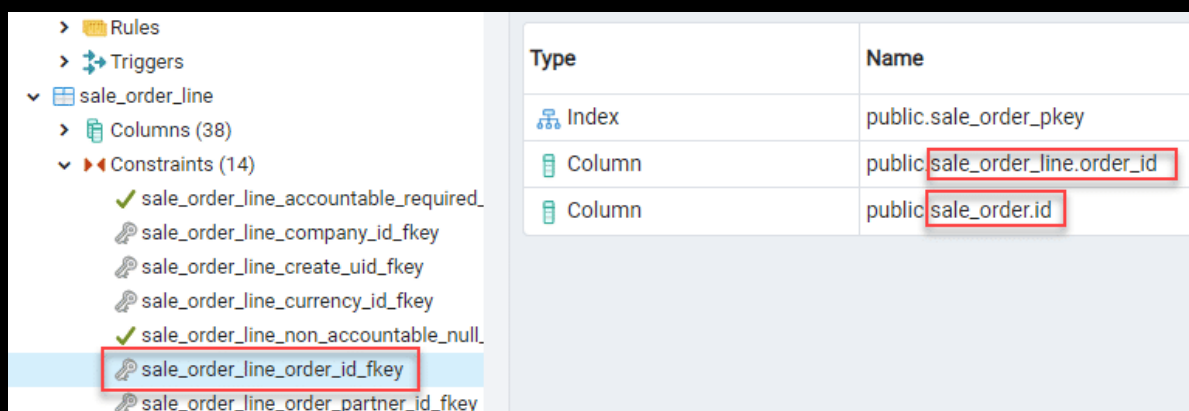
Foreing key en la tabla padre (one)



Type	Name
1..3 Sequence	public.sale_order_id_seq
(=) Function	nextval('sale_order_id_seq'::regclass)
Primary Key	public.sale_order_pkey
Foreign Key	public.procurement_group.procurement_group_sale_id_fkey
Foreign Key	public.purchase_order_line.purchase_order_line_sale_order_id_fkey
Foreign Key	public.sale_order_line.sale_order_line_order_id_fkey
Foreign Key	public.sale_order_option.sale_order_option_order_id_fkey
Foreign Key	public.sale_order_top_rel.sale_order_top_rel_order_id_fkey

Si analizamos dicha **foreign key** en la tabla '**sale\_order\_line**', vemos que relaciona el campo '**order\_id**' de ésta con el campo **id** de la tabla '**sale\_order**':

Foreing key en la tabla hijo (many)



Type	Name
Index	public.sale_order_pkey
Column	public.sale_order_line.order_id
Column	public.sale_order.id

## Analizando las relaciones entre las tablas

Cómo saber desde Odoo las relaciones entre las tablas con claves foráneas, y como PgAdmin nos puede ayudar con ello:

### Actualizando datos de Odoo desde PgAdmin

Vamos a analizar las relaciones entre las tablas de Odoo y cómo la aplicación PgAdmin nos puede ayudar.

Para ello vamos a irnos a “Ajustes” con el modo de desarrollador activado, para que nos aparezca la sección de “Técnicos”. Aquí podemos ver los “Modelos” que son nuestras tablas:

Nuevo Modelos ⚙		Q Buscar...	▼	1-80 / 526	<
<input type="checkbox"/> Modelo	Descripción del modelo	Tipo	Modelo transitorio		
<input type="checkbox"/> project.update	Actualización del proyecto	Objeto base	<input type="checkbox"/>		
<input type="checkbox"/> stock.picking	Albarán	Objeto base	<input type="checkbox"/>		
<input type="checkbox"/> account.move	Asiento contable	Objeto base	<input type="checkbox"/>		
<input type="checkbox"/> rating.mixin	Calificación Mixin	Objeto base	<input type="checkbox"/>		

Si aquí buscamos la tabla “product\_template”, si quisiéramos ver una relación *many2one* de *categ\_id* “categoría del producto”:

Nuevo Modelos		Producto ⚙				
bom_line_ids	Componentes de LdM	one2many	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
can_image_1024_be_zoomed	Puede agrandarse la imagen 1024	booleano	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
categ_id	Categoría de producto	many2one	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
color	Índice de Colores	entero	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Aquí vemos que en la relación, el campo `categ_id` lo está obteniendo de la tabla `product_category`, pero no nos indica la relación en este cuadro, aunque podríamos intuir que será la clave primaria en la tabla `product.category`:

**Abrir: Campos**

Nombre de campo ?	categ_id	Tipo de campo ?	many2one
Etiqueta de campo ?	Categoría de producto	Campo ayuda ?	

Propiedades

Permisos de acceso

Varios

**PROPIEDADES BASE**

Requerido ?	<input checked="" type="checkbox"/>	Modelo relacionado ?	product.category
Sólo lectura ?	<input type="checkbox"/>	Al eliminar ?	Restringir
Almacenado ?	<input checked="" type="checkbox"/>	Dominio ?	

Si buscamos esta tabla (`product.category`), en “Técnico” → “Modelos”, vemos que tiene un campo que tiene el ID, e intuimos que la relación puede ser esa:

<div><div>Nuevo</div><div>Modelos</div><div>Categoría de producto </div></div>				
display_name	Nombre mostrado	Carácter	<input type="checkbox"/>	<input checked="" type="checkbox"/>
filter_for_stock_putaway_rule	stock.putaway.rule	booleano	<input type="checkbox"/>	<input type="checkbox"/>
id	ID	entero	<input type="checkbox"/>	<input checked="" type="checkbox"/>
name	Nombre	Carácter	<input checked="" type="checkbox"/>	<input type="checkbox"/>

Pero una forma más fácil de ver la relación sería haciendo uso de PgAdmin.

Con PgAdmin, podemos en la tabla “`product_template`”, teniendo elegida la pestaña “Dependents” (Dependientes), vemos todas las relaciones que tiene:

Type	Name	Restriction
1..3 Sequence	public.product_category_id_seq	auto
Index	public.product_category__name_index	auto
Index	public.product_category__parent_id_index	auto
Index	public.product_category__parent_path_index	auto
Function	nextval('product_category_id_seq'::regclass)	auto
Foreign Key	public.product_category.product_category_create_uid_fkey	auto
Foreign Key	public.product_category.product_category_parent_id_fkey	auto
Foreign Key	public.product_category.product_category_removal_strategy_i...	auto
Foreign Key	public.product_category.product_category_write_uid_fkey	auto
Primary Key	public.product_category_pkey	auto

Pero si queremos ver concretamente el campo “categ\_id” podemos ver que hay una “Foreign Key” llamada “public.product\_template.product\_template\_categ\_id\_fkey” siendo product\_template la tabla, y “product\_template\_categ\_id\_fkey” el campo ID o clave foránea:

Type	Name	Restriction
Foreign Key	public.product_template.product_template_categ_id_fkey	auto
Rule	_RETURN ON public.vendor_delay_report	normal
Rule	_RETURN ON public.report_pos_order	normal

Si nos vamos a las “Constraints” (Restricciones) de la tabla observamos la clave foránea que acabamos de ver:

product_template
Columns (44)
Constraints (7)
product_template_categ_id_fkey
product_template_company_id_fkey
product_template_create_uid_fkey

Si pulsamos ahora en “Dependencies” (Dependencias), nos dice que la relación es de “public.product\_template.categ\_id” con el “public.product\_category.id”, de modo que aquí se ve más fácilmente lo que antes solo intuíamos:

Dependencies	
Type	Name
Index	public.product_category_pkey
Column	public.product_template.categ_id
Column	public.product_category.id

Si nos vamos a la tabla “product\_category” y abrimos la pestaña “Dependents” (Dependientes), vemos que tiene un campo “id” y con esto vemos que esta es una forma más rápida de analizar cual es la relación de los campos entre tablas, con una Foreign Key:

Type	Name	Restriction
Foreign Key	public.stock_warehouse_orderpoint.stock_war...	normal
Foreign Key	public.stock_valuation_layer.stock_valuation_la...	normal
Foreign Key	public.stock_route_categ.stock_route_categ_ca...	normal
Foreign Key	public.stock_putaway_rule.stock_putaway_rule...	normal
Foreign Key	public.product_template.product_template_cat...	normal
Foreign Key	public.product_pricelist_item.product_pricelist_...	normal
Primary Key	public.product_category_pkey	auto
Sequence	public.product_category_id_seq	auto

## Haciendo backups y restaurando la base de datos

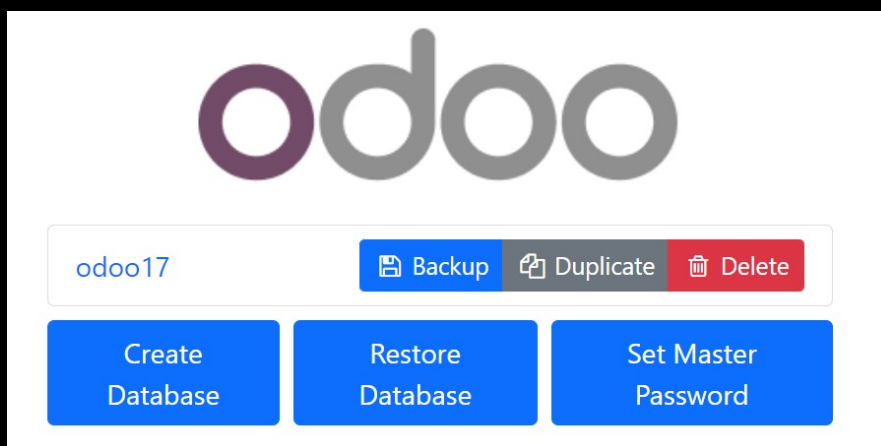
Las copias de seguridad y restauraciones de la base de datos pueden hacerse de **muchas formas** (**incluso** se pueden crear **scripts** para automatizarlas). Aquí explicaremos las **formas más comunes** para nuestro servidor que lo tenemos **en local** (**también** podría hacerse backups si trabajamos en la **nube**):

### 1. Para **bases de datos pequeñas**:

Con el propio **gestor de bases de datos** de la **interface** que nos proporciona **Odoo** para hacer **backups**. Para ello, simplemente tendremos que entrar en el **navegador** en la siguiente web:

<http://localhost:8069/web/database/manager>

Interface que ofrece Odoo para hacer backups/restore



Esta elección es ideal. Nos dará la opción de **crear un .zip**, con la **carpeta filestore** (donde se guardan los adjuntos) o un **pg\_dump sin la carpeta filestore**. La **restauración** podría hacerse desde la **misma ventana**.

### 2. Para **bases de datos más grandes**:

Lo ideal sería hacer uso del comando **pg\_dump** que ofrece PostgreSQL y más concretamente con el uso de un **compresor**.

En su web oficial (<https://www.postgresql.org/docs/9.5/backup-dump.html>) nos indican como hacerlo **con el terminal**:

- Backup: **pg\_dump dbname | gzip > filename.gz**

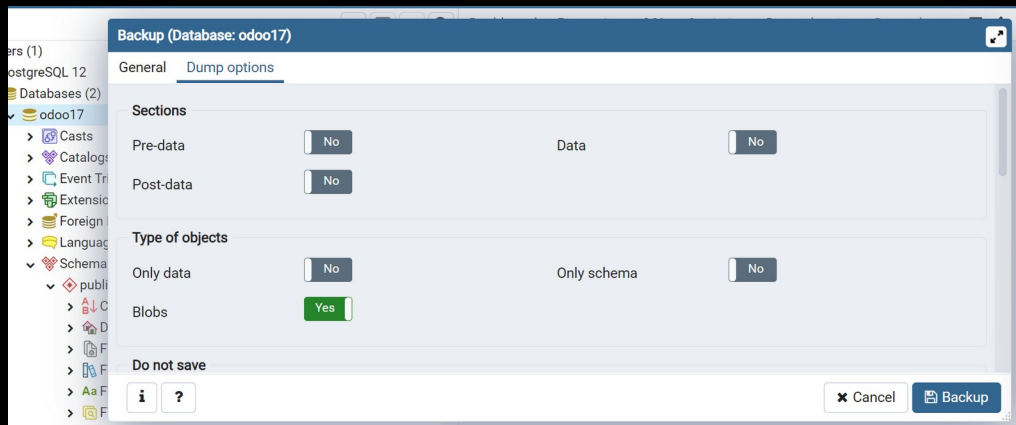
- Restauración: **gunzip -c filename.gz | psql dbname**

### 3. Otras formas de hacer backups e incluso de automatizar el proceso:

Se pueden hacer copias de seguridad de bases de datos desde:

- La aplicación **PgAdmin**:

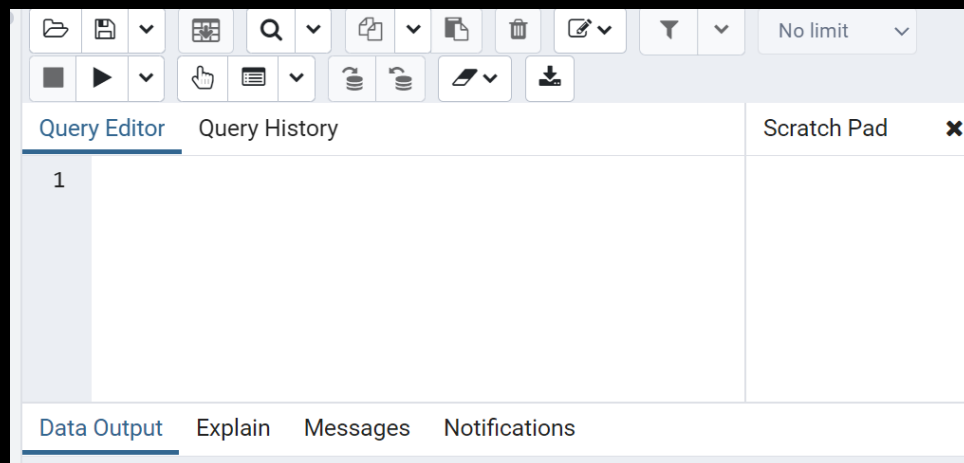
Simplemente tendremos que pulsar con el botón derecho sobre la base de datos a la que queramos crearle una copia de seguridad y después en 'backup'.



- Con **módulos de terceros**, como puede ser '**Database auto-backup**', descargable desde la web oficial de Odoo. Una vez instalado el módulo, cuando entremos en modo desarrollador, aparecerá un **nuevo menú dentro de Técnicos / 'estructura de la base de datos'** que se denomina '**copias de seguridad automatizadas**'. Este nuevo menú nos permitirá configurar **copias de seguridad automáticas** de nuestra base de datos.

Con la herramienta '**Query Tool**' podremos, por ejemplo:

- **Visualizar** un dato en la base de datos: Por ejemplo, realizar una consulta que nos devuelva un listado con las tablas de la base de datos de Odoo ordenadas de mayor a menor tamaño total (excluir las que tengan tamaño 0), seleccionando los datos de pg\_catalog.
- **Modificar** un dato de la base de datos.
- **Insertar** un dato en la base de datos.



[https://www.pgadmin.org/docs/pgadmin4/latest/query\\_tool.html](https://www.pgadmin.org/docs/pgadmin4/latest/query_tool.html)

[https://www.odoo.com/es\\_ES/](https://www.odoo.com/es_ES/)

<https://www.pgadmin.org/>

<https://www.postgresql.org/>