El acceso a la base de datos de WordPress de *Mosaic* desde el *plug-in* VisDa

Carlos Casado

PID_00201064

Universitat Oberta de Catalunya

Índice

1. Introducción

Como hemos visto en el capítulo anterior, WordPress es un gestor de contenidos especializado en la gestión de blogs muy usado por su potencia, flexibilidad, facilidad de uso y por ser GPL. Como la mayoría de gestores de contenidos actuales, WordPress se puede ampliar con facilidad mediante la adición de *plug-ins* que **cualquiera** puede programar y distribuir apoyándose en la documentación que los desarrolladores del programa ponen a disposición de la comunidad.

Esta material pretende ser una ayuda para aquellas personas que desean crear un *plug-in* para WordPress y en particular para aquellos que deseen visualizar la información de un blog realizado con dicho gestor de contenidos.

Si queremos crear un *plug-in* para WordPress es posible que queramos recuperar alguna información de la base de datos. Si queremos listar las etiquetas más usadas, los usuarios que más escriben, los *posts* con más comentarios, deberemos acceder a la base de datos para poder recuperar esa información.

En general, si queremos recuperar cualquier contenido generado por el usuario y almacenado en WordPress, deberemos acceder a la base de datos y recuperar de allí la información. En la base de datos se guarda información sobre las entradas, los usuarios, etiquetas y comentarios, entre otros. Conocer la base de datos, cómo está estructurada (tablas, campos, relaciones), qué información podemos encontrar en ella y de qué manera extraer los datos es imprescindible si queremos crear un *plug-in* que use la información generada por el usuario. Este apartado pretende explicar cómo extraer información de la base de datos.

Antes de entrar en la estructura de la base de datos de WordPress, hay que tener presente que WordPress usa MySQL y está programado en PHP. Se le supone al lector un cierto conocimiento de MySQL y el acceso a los datos desde PHP.

Para explicar algunas características de la base de datos de WordPress, vamos a usar datos de la revista *Mosaic*. *Mosaic* es una revista electrónica que en el momento de redactar este material tiene algo más de diez años de vida y 94 números.

Este material se divide básicamente en dos partes: la primera es un repaso exhaustivo de la base de datos de WordPress y la segunda explica cómo obtener los datos que nos interesan desde un *plug-in*.

Este documento está actualizado a la versión 3.4 de WordPress. Aunque la base de datos de WordPress es bastante estable y no suele modificarse con los cambios de versión, es conveniente tener en cuenta que en futuras versiones la base de datos podría no ser exactamente igual a la que se trata en este material.

Algunos detalles de nomenclatura

WordPress, como gestor de contenidos especializado en blogs, basa toda la estructura de su base de datos en el contenido principal, los *posts*, aunque tiene también otro tipo de contenido, al que llama páginas. La diferencia principal es que las páginas están pensadas para un contenido que siempre debe estar presente y que acostumbra a ser accesible desde un menú en la página principal del blog. Por contra, los *posts* aparecen al principio de la primera página cuando hace poco que están escritos y van hundiéndose en la estructura del blog a medida que van siendo más antiguos y otros más modernos van ocupando su posición.

En castellano, la palabra *post* suele traducirse por entrada o por artículo, aunque sigue siendo bastante común el uso de la palabra inglesa. En este texto, se usan, indiferentemente, *post* y entrada.

En cuanto a las páginas, conviene diferenciar una página de WordPress de una página web. En este documento, todas las veces que se use la palabra *página* será con referencia a las páginas de WordPress, a menos que se especifique *web* a continuación.

Finalmente, como WordPress, usaremos la palabra *término* para referirnos de manera genérica a categorías y etiquetas.

2. Estructura de la base de datos de WordPress

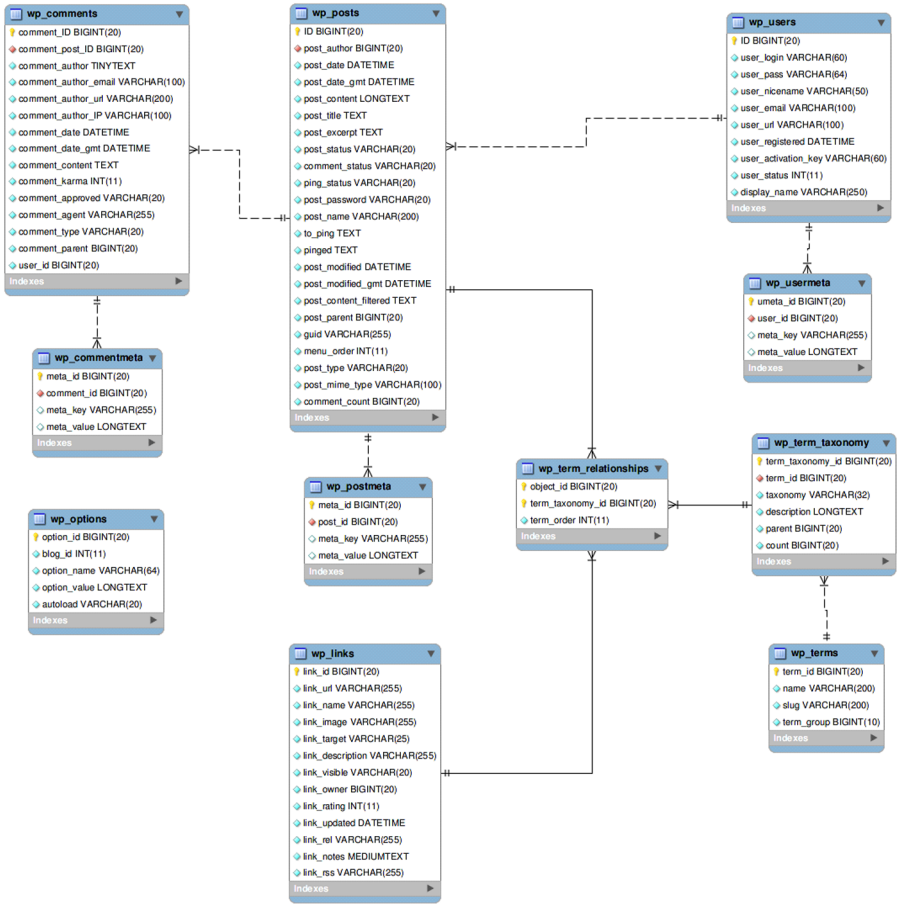
La base de datos de WordPress se fundamenta en estas once tablas:

Tablas WordPress

Nombre de la tabla	Descripción
wp_post	Es la tabla más importante de la base de datos de WordPress. Guarda información sobre las entradas, las páginas, los ficheros adjuntos y los menús. Para cada versión de una entrada se guarda su contenido, el título, el autor, la fecha, el resumen, estatus, etc. Existe un campo (post_type) que indica de qué tipo es.
wp_postmeta	Metadatos de las entradas. Entre otras cosas, guarda información sobre los adjuntos asociados.
wp_comments	En esta tabla se guardan los comentarios con su información asociada: autor, fecha, contenido, si está o no aprobado, etc.
wp_commentsmeta	Metadatos de los comentarios. Tan solo es usada por algunos *plug-ins* como Akismet.
wp_links	Lista de enlaces.
wp_term_relationships	A través de esta tabla se relacionan las entradas y los enlaces con sus respectivas etiquetas y categorías.
wp_term_taxonomy	Tabla de etiquetas y categorías. WordPress guarda juntas etiquetas y categorías tanto de los enlaces como de los posts (aunque no son compartidas).
wp_terms	Nombres de las etiquetas y las categorías.
wp_users	Información de los usuarios. En esta tabla se guardan el nombre de usuario (*nicename*), el correo electrónico, la dirección web que este especifique, la fecha de registro, la clave de activación (que se envía para comprobar la dirección de correo electrónico de los nuevos usuarios) y el nombre que debe mostrarse.
wp_usersmeta	Opciones de los usuarios. Aquí se guarda el resto de información del usuario, como su nombre real, el tipo de editor que usa, su nivel, etc.

Nombre de la tabla	Descripción
wp_options	Tabla de opciones de WordPress.

Figura 1. Diagrama de la base de datos de WordPress



Disponible en http://codex.wordpress.org/Database_Description

En la Figura 1, podemos ver todas las tablas de la base de datos de WordPress y las relaciones entre ellas. Como es normal, la base de datos parece centrada en la tabla de entradas. Tan solo la tabla de opciones no mantiene ninguna relación con wp_posts.

Un detalle importante es que, a pesar de que en la imagen el nombre de todas las tablas empieza por wp_ , esto no siempre es así. WordPress permite cambiar el prefijo de las tablas para facilitar la convivencia de diferentes instalaciones del gestor de contenidos en una misma base de datos. Existe una clase global, llamada $wpbp, que nos facilita el acceso a las tablas independientemente del prefijo que se haya puesto al nombre.

3. Las tablas una a una

3.1. wp_posts

La tabla wp_posts es la que tiene más campos y la de más peso en la mayoría de instalaciones de WordPress.

En *Mosaic*, en el momento de escribir este texto, wp_posts tenía 7.924 filas que ocupaban un total de 60 Mb. Para hacernos una idea, la siguiente tabla en peso es la wp_term_relationships con 436 Kb y 7.455 filas. En wp_posts, se guardan las entradas (*posts*), las páginas, todas las revisiones que se hacen de ambos tipos de contenidos y la información relacionada con todos los archivos que se suben.

Opción de mantener revisiones

Si se desea, se puede desactivar la opción de mantener las revisiones añadiendo la siguiente línea en el archivo wp-config.php:
define('WP_POST_REVISIONS', false);

Los campos que contiene dicha tabla son los siguientes:

Campos tabla wp_posts

Campo	Tipo	Descripción
ID	BIGINT(20)	Identificador.
post_author	BIGINT(20)	Autor. Relaciona la tabla wp_posts con la tabla wp_users.
post_date	DATETIME	Fecha y hora de creación de la entrada/página en el horario local.
post_date_gmt	DATETIME	Fecha y hora de creación de la entrada/página en horario gmt.
post_content	LONGTEXT	El contenido de la entrada o página.
post_title	TEXT	El título de la entrada o página.
post_excerpt	TEXT	Extracto.
post_status	VARCHAR(20)	Situación del post: publicado, pendiente, privado, etc.
comment_status	VARCHAR(20)	Estado de los comentarios: *open*/*closed* (se aceptan o no).
ping_status	VARCHAR(20)	Situación de los pings: *open*/*closed*.

Campo	Tipo	Descripción
post_password	VARCHAR(20)	Si se asigna una contraseña a una entrada, esta sólo puede ser leída si se introduce dicha contraseña. Si su estado es *publish*, aparecerá el título pero solicitará el *password* para poderla leer. La contraseña se guarda en la base de datos sin encriptar.
post_name	VARCHAR(200)	Nombre de la entrada. Si está publicada coincide con el título. Si es una revisión o un autoguardado, recibe como nombre el ID de la entrada, la palabra *revisión* o *autosave* y el número de revisión (si hay más de una).
to_ping	TEXT	Si desde esta entrada se hace un ping a otra página, queda reflejado aquí.
pinged	TEXT	Si esta entrada ha recibido un ping desde otra página, queda reflejado aquí.
post_modified	DATETIME	Fecha de la última modificación de la entrada o página en horario local.
post_modified_gmt	DATETIME	Ídem que la anterior pero en formato GMT.
post_content_filtered	TEXT	No usado directamente por WordPress, usado por algunos *plug-ins* que modifican el contenido del post, para no tener que hacer la modificación cada vez que se muestra la entrada.
post_parent	BIGINT(20)	En caso de que sea una revisión, se refiere al ID del post revisado. En caso de que sea un adjunto, será el ID del post en que se subió. En caso de que sea una página o un ítem de menú, la página o el ítem de menú padre.
guid	VARCHAR(255)	La dirección del post o del archivo (si se trata de un adjunto).
menu_order	INT(11)	Se usa en los ficheros adjuntos para mantener un orden entre ellos. También se usa para listar las páginas en un orden diferente al alfabético, que es el que usa WordPress por defecto.
post_type	VARCHAR(20)	Clase de contenido almacenado (*attachment, page*, post, revision, nav_menu_item).
post_mime_type	VARCHAR(100)	Tipo de contenido cuando se trata de un archivo adjunto (jpg, gif, pdf, etc.).

Campo	Tipo	Descripción
comment_count	BIGINT(20)	Número de comentarios.

El campo **post_status** puede tener los siguientes valores:

- **publish**: La entrada o página está publicada y es visible para todos los visitantes.
- **inherit**: Una revisión.
- **pending**: Pendiente de publicación.
- **private**: Privado, solo accesible por los usuarios registrados.
- **future**: Con fecha de publicación futura.
- **draft**: Borrador, no guardado todavía.
- **trash**: Borrado.

En el campo **post_type** se guarda qué clase de contenido se almacena en ese registro. Las diferentes clases pueden ser las siguientes:

- **attachment**: Indica que en ese registro se guarda información de un archivo subido; el enlace al archivo se encuentra en el campo **guid**.
- **page**: Página.
- **post**: Entrada.
- **revision**: Revisión de una entrada.
- **nav_menu_item**: Ítem de menú.

Un *plug-in* podría añadir nuevas clases de contenido.

Índices

Esta tabla tiene los siguientes índices:

- post_name (primario)

- post_type+post_status+post_date+ID
- post_parent
- post_author

3.2. wp_postmeta

En la tabla wp_posmeta se guarda información referente a los registros guardados en wp_post.

Por ejemplo, se guardan todos los campos personalizados de las entradas, también se usa para bloquear una entrada o página y evitar así que dos usuarios puedan modificarla a la vez. Como las páginas pueden usar diferentes plantillas, también se guarda en esta tabla la plantilla que se usa para una determinada página.

Campos tabla wp_postmeta

Campo	Tipo	Descripción
meta_id	BIGINT(20)	Identificador del registro.
post_id	BIGINT(20)	Identificador del post o entrada al que va referido.
meta_key	VARCHAR(255)	El nombre del campo personalizado.
meta_value	LONGTEXT	El valor del campo personalizado.

Índices

Esta tabla tiene los siguientes índices:

- meta_id
- post_id
- meta_key

3.3. wp_comments

En la tabla wp_comments se guardan los comentarios tanto de las entradas como de las páginas.

Campos tabla wp_postmeta

Campo	Tipo	Descripción
comment_id	BIGINT(20)	Identificador del comentario.
comment_post_id	BIGINT(20)	Identificador de la entrada o página relacionada.
comment_author	TINYTEXT	Nombre del autor. Se le pide antes de escribir el comentario. Es un dato obligatorio.
comment_author_email	VARCHAR(100)	Correo del autor. También se le obliga a ponerlo antes de publicar el comentario.
comment_author_url	VARCHAR(200)	Dirección de la página web del comentarista. No obligatorio.
comment_author_ip	VARCHAR(100)	Dirección IP de la persona que escribe el comentario.
comment_date	DATETIME	Fecha y hora en horario local.
comment_date_gmt	DATETIME	Fecha y hora en formato GMT.
comment_content	TEXT	El contenido del comentario.
comment_karma	INT(11)	Este campo no es usado directamente por WordPress (al menos en apariencia) pero sí usado por algunos *plug-ins*.
comment_approved	VARCHAR(20)	Especifica si el comentario está aprobado (1), no (0) o está marcado como spam (spam).
comment_agent	VARCHAR(255)	En caso de que el comentario sea un *trackback* o un *pingback* guarda la dirección desde la que se ha hecho.
comment_type	VARCHAR(20)	Determina si el comentario es un *trackback*, un *pingback*, o un *comment*.

Campo	Tipo	Descripción
comment_parent	BIGINT(20)	En caso de que el tema ofrezca la posibilidad de anidar comentarios, guarda el ID del comentario padre.
user_id	BIGINT(20)	En caso de que el usuario que hace el comentario sea un usuario registrado, guarda su ID.

Índices

Esta tabla tiene los siguientes índices:

- comment_ID
- comment_approved
- comment_post_ID
- comment_approved + comment_date_gmt
- comment_date_gmt
- comment_parent

3.4. wp_commentmeta

La tabla wp_commentmeta permite añadir metadatos a los comentarios de WordPress.

La aplicación no la usa directamente, sino a través del *plug-in* Akismet, que va incluido por defecto en WordPress.

Campos tabla wp_commentmeta

Campo	Tipo	Descripción
meta_id	BIGINT(20)	Identificador del registro.
comment_id	BIGINT(20)	Identificador del comentario al que va referido.

Campo	Tipo	Descripción
meta_key	VARCHAR(255)	El nombre del campo personalizado.
meta_value	LONGTEXT	El valor del campo personalizado.

Índices

Esta tabla tiene los siguientes índices:

- meta_ID
- comment_id
- meta_key

3.5. wp_links

En la tabla wp_links se guarda información sobre los enlaces almacenados en WordPress.

WordPress ofrece una herramienta de gestión de enlaces bastante potente.

Campos tabla wp_links

Campo	Tipo	Descripción
link_id	BIGINT(20)	Identificador del registro.
link_url	VARCHAR(255)	Dirección del enlace.
link_name	VARCHAR(255)	Nombre de la página enlazada.
link_image	VARCHAR(255)	Imagen representativa de la página enlazada.
link_target	VARCHAR(25)	Tres opciones: _top, _blank o _none.
link_description	VARCHAR(255)	Descripción de la página enlazada.
link_visible	VARCHAR(20)	Si es visible o no en el blog.
link_owner	BIGINT(20)	Identificador del usuario que ha añadido el enlace.

Campo	Tipo	Descripción
link_rating	INT(11)	Puntuación (de 1 a 10) del enlace.
link_updated	DATETIME	WordPress no usa actualmente este campo, pero hay *plug-ins* que ponen en él la fecha en la que se ha modificado el enlace.
link_rel	VARCHAR(255)	Relación de conocimiento del usuario que pone el enlace con el propietario de la página enlazada.
link_notes	MEDIUMTEXT	Comentarios sobre el enlace.
link_rss	VARCHAR(255)	Dirección RSS de la página enlazada.

Índices

Esta tabla tiene los siguientes índices:

- link_ID
- link_category
- link_visible

3.6. wp_users

En la tabla wp_users se guarda información sobre los usuarios registrados.

Campos tabla wp_users

Campo	Tipo	Descripción
ID	BIGINT(20)	Identificador del registro.
user_login	VARCHAR(60)	Nombre de identificador del usuario. Es el nombre que el usuario usa para acceder al escritorio de WordPress. Puede tener mayúsculas, minúsculas y espacios, pero no acentos.

Campo	Tipo	Descripción
user_pass	VARCHAR(64)	Contraseña codificada en MD5.
user_nicename	VARCHAR(50)	El *login* convertido en minúsculas y con los espacios substituidos por guiones.
user_email	VARCHAR(100)	Dirección de correo electrónico del usuario.
user_url	VARCHAR(100)	Dirección de la página web del usuario.
user_registered	DATETIME	Fecha en la que el usuario se registró.
user_activation_key	VARCHAR(60)	Código de comprobación. Se envía al usuario si ha perdido la contraseña, para permitir la creación de una nueva.
user_status	INT(11)	Actualmente no usado por WordPress.
display_name	VARCHAR(250)	Nombre que se muestra en las entradas escritas por el usuario.

Índices

Esta tabla tiene los siguientes índices:

- ID
- user_login
- user_nicename

3.7. wp_usermeta

En la tabla wp_usermeta se guarda información del usuario que no se almacena en la tabla wp_users.

Entre la información que se guarda aquí, está el nombre real del usuario, su nivel de acceso, su alias (*nickname*) o el tipo de editor que usa, entre otros.

Campos tabla wp_usermeta

Campo	Tipo	Descripción
umeta_id	BIGINT(20)	Identificador del registro.
user_id	BIGINT(20)	Identificador del usuario al que va referido.
meta_key	VARCHAR(255)	El nombre del campo personalizado.
meta_value	LONGTEXT	El valor del campo personalizado.

Índices

Esta tabla tiene los siguientes índices:

- umeta_id
- user_id
- meta_key

3.8. wp_terms

En la tabla wp_terms se guardan todas las categorías, etiquetas y formatos creados.

Es importante tener en cuenta que si una categoría y una etiqueta tienen el mismo nombre no estarán duplicadas en esta tabla sino que el nombre solo aparecerá una vez y la información contenida en la tabla wp_term_taxonomy será la que nos sirva para diferenciar uno de otro.

Formatos predeterminados

Desde la versión 3.1 de WordPress, se incluyen también aquí los formatos predeterminados para las entradas. Estos formatos permiten tener diferentes plantillas para diferentes tipos de entradas. Esos formatos están predeterminados y no deberían añadirse nuevos ni por temas ni por *plug-ins*.

Campos tabla wp_terms

Campo	Tipo	Descripción
term_id	BIGINT(20)	Identificador del registro.

Campo	Tipo	Descripción
name	VARCHAR(200)	Nombre.
slug	VARCHAR(200)	Nombre sólo con letras, dígitos y guiones. Usado para facilitar su uso en una dirección http. Este es el nombre que se usa cuando listamos todas las entradas de una determinada categoría o que contienen una determinada etiqueta.
term_group	BIGINT(10)	El valor del campo personalizado.

Índices

Esta tabla tiene los siguientes índices:

- term_ID
- slug
- name

3.9. wp_term_taxonomy

En la tabla wp_term_taxonomy se guarda información sobre el término (si es una categoría, una etiqueta, una categoría de un enlace o un formato) guardado en la tabla wp_terms. También guarda información acerca del número de veces que se ha usado esa categoría o etiqueta y la relación padre-hijo si existe.

Campos tabla wp_term_taxonomy

Campo	Tipo	Descripción
term_taxonomy_id	BIGINT(20)	Identificador del registro.
term_id	BIGINT(20)	Identificador de la tabla wp_terms.

Campo	Tipo	Descripción
taxonomy	VARCHAR(32)	Puede tener cuatro posibles valores: category para las categorías, link_category para las categorías de los enlaces, post_tag para las etiquetas y post_format para los formatos predeterminados. Actualmente WordPress permite crear taxonomías con lo que a estos valores podrían añadírseles los que fuesen necesarios para las nuevas taxonomías.
description	LONGTEXT	Descripción.
parent	BIGINT(20)	En el caso de las categorías puede haber una relación de padre-hijo. En ese caso, esa relación queda aquí especificada.
count	BIGINT(20)	Número de entradas/enlaces que tienen esa categoría, etiqueta o formato. Pueden existir registros que tengan en este campo un valor cero si se han creado pero no se están usando.

Índices

Esta tabla tiene los siguientes índices:

- term_taxonomy_id
- term_id + taxonomy
- taxonomy

3.10. wp_term_relationships

La tabla wp_term_relationships permite relacionar los términos con las entradas y enlaces que las usan o ambos.

Campos tabla wp_term_relationships

Campo	Tipo	Descripción
object_id	BIGINT(20)	Identificador de la entrada o enlace. Dado que puede referirse a cualquiera de las dos tablas, podemos encontrarnos con valores repetidos. Así, un 2 puede referirse tanto a un enlace como a una entrada. Para diferenciarlas deberemos ir a la tabla wp_term_taxonomy.
term_taxonomy_id	BIGINT(20)	Identificador de la tabla wp_term_taxonomy a partir del cual sabremos si es una categoría, una categoría de enlaces o una etiqueta, así como su nombre.
term_order	INT(11)	Sin usar actualmente.

Índices

Esta tabla tiene los siguientes índices:

- object_id + term_taxonomy_id
- term_taxonomy_id

3.11. wp_options

En la tabla wp_options se guardan las diferentes opciones de WordPress. También se guardan aquí las opciones creadas por los *plug-ins* instalados.

WordPress ofrece funciones para añadir, consultar, modificar y borrar filas de esta tabla, por lo que no será necesario acceder a ella directamente. Sin embargo, a modo informativo, se explican los campos de los que está compuesta.

Campos tabla wp_options

Campo	Tipo	Descripción
option_id	BIGINT(20)	Identificador del registro.
blog_id	INT(11)	Identificador del blog (en multiblog).

Campo	Tipo	Descripción
option_name	VARCHAR(64)	Nombre de la opción.
option_value	LONGTEXT	Valor de la opción.
autoload	VARCHAR(20)	Cuando WordPress arranca carga todas las opciones que estén marcadas como *yes* en este campo. *Yes* es, además, el valor por defecto.

Índices

Esta tabla tiene los siguientes índices:

- option_id + blog_id + option_name
- option_name

4. Relaciones entre tablas

A menudo, cuando queramos obtener información de la base de datos de WordPress los datos que deseamos obtener están repartidos entre dos o más tablas relacionadas entre sí. Nos interesa conocer qué tablas están relacionadas entre sí y cuál es esa relación para saber cómo recuperar los datos.

Ved también

En el apartado "Estructura de la base de datos de WordPress", encontraréis la figura 1, donde pueden verse todas las relaciones entre tablas que vamos a explicar en este apartado.

4.1. Las tablas meta

Las tablas wp_comments, wp_posts y wp_users tienen cada una de ellas una tabla relacionada llamadas wp_commentmeta, wp_postmeta y wp_usermeta respectivamente, donde se guarda información opcional.

Tablas "meta"

	Tabla	Tabla de opciones
Nombre **Campo de relación**	wp_posts ID	wp_postmeta meta_id
Nombre **Campo de relación**	wp_comments comment_ID	wp_commentmeta meta_id
Nombre **Campo de relación**	wp_users ID	wp_usermeta umeta_id

Habitualmente, nos va a interesar recuperar los campos opcionales de una determinada entrada, de un comentario o de un usuario. Sabiendo el ID, fácilmente podremos acceder a los datos opcionales.

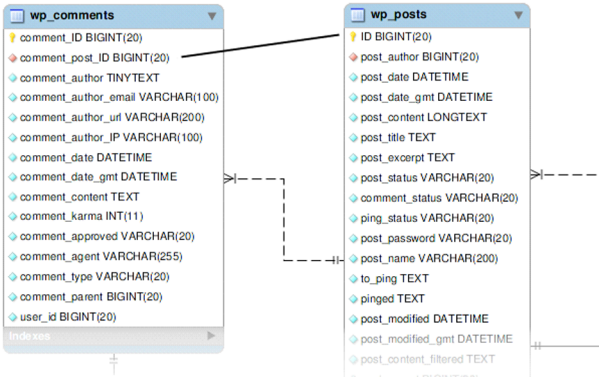
Ved también

En el apartado "Consultas sencillas de este material", se pueden encontrar algunos ejemplos sobre cómo recuperar los datos opcionales de una determinada entrada.

4.2. Comentarios y entradas

La tabla de comentarios tiene una relación *N* a 1 con la tabla de *posts*. La relación se establece a través del campo comment_post_ID de la tabla wp_comments con el campo ID de la tabla wp_posts.

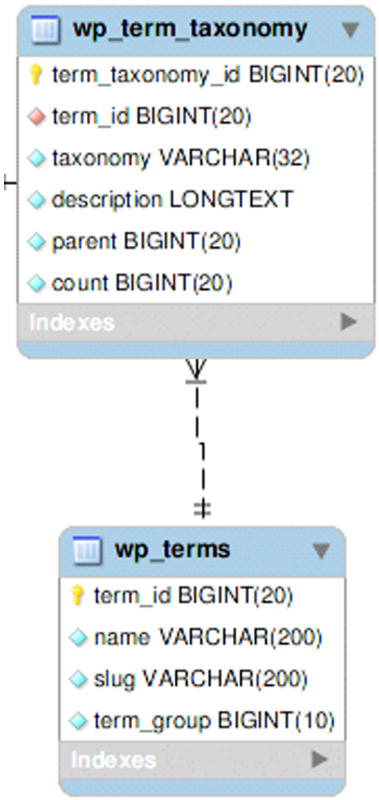
Figura 2. Las tablas wp_comments y wp_posts



4.3. wp_term_taxonomy y wp_terms

Tal como ya se ha explicado antes, la tabla wp_term_taxonomy guarda información acerca del término (categoría, etiqueta, categoría de enlace) mientras que la tabla wp_terms guarda el nombre del elemento. Estas dos tablas tienen poca utilidad por separado así que será habitual consultar las dos conjuntamente.

Figura 3. Etiquetas y categorías



La relación de estas dos tablas es *N* a 1 dado que un término puede ser dos cosas a la vez: podemos tener una categoría llamada **fotos** y también una etiqueta con el mismo nombre. Siguiendo con este ejemplo, en la tabla wp_terms tendríamos un registro donde en el campo *name* tendríamos la palabra **fotos** (y en el campo *term_id* un valor, **n**) y en la tabla wp_term_taxonomy tendríamos dos registros, con un mismo valor (**n**) en el campo *term_id* y dos valores diferentes (uno en cada registro) en el campo *taxonomy*:

Tabla wp_terms

wp_terms	
term_id	5
name	Fotos
slug	Fotos

wp_terms	
term_grup	0

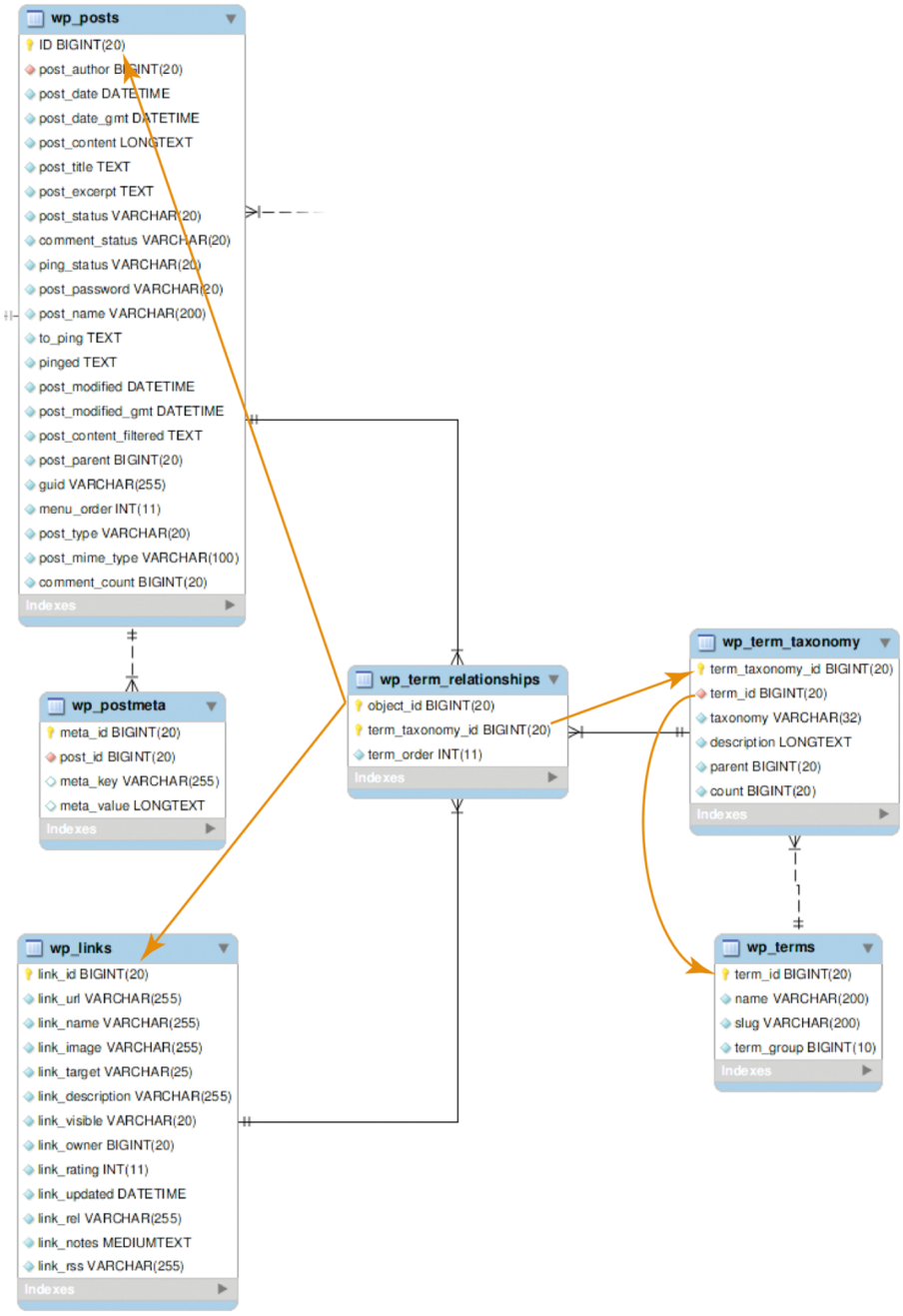
Tabla wp_term_taxonomy

wp_term_taxonomy		
	Registro 1	**Registro 2**
term_taxonomy_id	102	253
term_id	5	5
taxonomy	post_tag	category

4.4. Entradas, enlaces, categorías y etiquetas

El siguiente cuadro de relaciones (figura 4) es el más complejo pero también el más útil a la hora de crear un *plug-in* de visualización de datos de WordPress. Estas relaciones nos permiten saber las entradas de una determinada categoría o que tienen una determinada etiqueta. Aunque la tabla de enlaces seguramente no sería útil a la hora de visualizar información, se incluye aquí porque forma parte del conjunto.

Figura 4. Las relaciones entre las entradas y los enlaces y las categorías y etiquetas



La relación entre las entradas y las categorías o las etiquetas se establece a través de la tabla wp_term_relationships, donde se guardan pares de valores compuestos por el identificador de la entrada y el identificador de la categoría o etiqueta (representado por el identificador de la tabla wp_term_taxonomy).

La dificultad de este conjunto de interrelaciones viene dada por la necesidad, si se quiere obtener las entradas de una determinada categoría por ejemplo, de obtener todas las entradas relacionadas desde la tabla wp_terms, donde estará el nombre de la categoría.

El camino será:

- Buscar la categoría por su nombre en la tabla wp_terms. Obtenemos el identificador del término (campo *term_id*).

- Con el identificador de la categoría, buscar el identificador de la tabla wp_term_taxonomy (term_taxonomy_id). Puesto que podemos tener un mismo nombre asociado a categoría y etiqueta (por ejemplo), de los registros que obtengamos, nos quedaremos aquel que en el campo *taxonomy* tenga *category*. Con esto, obtenemos el identificador term_taxonomy_id.

- En la tabla wp_term_relationships buscar todas las apariciones del identificador de wp_term_taxonomy (term_taxonomy_id). Obtendremos un conjunto de registros correspondiente con todas las entradas de esa categoría.

- Obtener los datos que deseemos de las entradas a partir de los registros encontrados.

A partir de estos sencillos pasos, se puede crear una consulta SQL que obtenga los títulos de todas las entradas de una determinada categoría (o cualquier otro dato que nos interese como la fecha de publicación o el contenido, por ejemplo). En el siguiente apartado, vamos a ver algunos ejemplos sencillos de cómo hacerlo.

5. Consultar la base de datos

Los próximos apartados explican cómo acceder a la base de datos para obtener la información que nos interese. Están planteados de manera que sean fáciles de seguir sin necesidad de tener conocimientos de SQL, el lenguaje que usaremos para acceder a los datos.

Evidentemente, unos buenos conocimientos de SQL pueden hacer innecesario leer esta parte del material. Sin embargo, si queréis crear un *plug-in* para WordPress que incluya accesos de lectura y escritura a la base de datos puede ser interesante tener presentes los siguientes consejos:

- No añadáis campos a las tablas de WordPress. Las tablas pueden cambiar en versiones posteriores, lo que podría hacer que el *plug-in* dejase de funcionar.

- No añadáis registros directamente a la tabla wp_options, usad las funciones que WordPress ofrece para ello.

- Usad siempre la clase wpdb para referiros a las tablas de WordPress. Eso facilitará que el *plug-in* funcione en todas las instalaciones de WordPress.

- Si tenéis que crear una nueva tabla, usad el método *query* de la clase wpdb. Podéis ver un ejemplo en la web de Hungred Dot Com

5.1. Consultas sencillas

Empezamos por un ejemplo sencillo, obtener los títulos de todas las entradas del blog. La consulta SQL sería:

```
SELECT wp_posts.post_title
FROM wp_posts;
```

Prefijo

Cuando configuramos en WordPress se nos permite escoger el prefijo que tendrán las tablas de la base de datos. Por defecto, es wp_ pero se puede escoger cualquier otro. Para la elaboración de este material, en los ejemplos SQL se ha usado el prefijo por defecto.

Sin embargo, en los ejemplos de PHP, como se explica en los consejos iniciales de este apartado, se usa la clase $wpdb, que se encarga de usar el prefijo que corresponda.

Lo que traducido vendría a decir: seleccionamos el campo título de la tabla wp_posts.

Esta consulta debería hacerse de la siguiente manera en WordPress:

```
global $wpdb;
$query = "SELECT $wpdb->posts.post_title
FROM $wpdb->posts";
$resultado = $wpdb->get_results($query,ARRAY_N);
```

Para poder ejecutar la instrucción SQL en WordPress, primero debemos guardarla en una variable para después ejecutarla. La variable global $wpbd facilita la consulta en cualquier instalación del gestor de blogs, ya que la hace independiente del prefijo que se haya escogido para las tablas al instalar WordPress.

En la variable $resultado se guardará un *array* con el contenido de todos los títulos del blog. Sin embargo, si ejecutamos la instrucción tal cual, veremos muchos títulos repetidos, títulos de entradas que no existen, títulos de páginas... Toca un repaso a lo que vimos sobre la tabla wp_posts.

WordPress guarda en la tabla wp_posts los datos necesarios de *posts*, páginas, archivos subidos y menús creados por los usuarios. Para poder diferenciar los diferentes tipos de registros guardados en la tabla, existe un campo llamado post_type, donde se guarda dicha información. Así, en los *posts*, el valor tomado por ese campo será 'post'.

Teniendo eso en cuenta, podemos escribir la instrucción SQL que nos permitirá listar solamente los títulos de los *posts*:

```
SELECT wp_posts.post_title
FROM wp_posts
WHERE wp_posts.post_type = 'post'
```

Sin embargo, esta instrucción incluye borradores y *posts* programados para verse más adelante.

¿Qué debemos hacer para poder limitar los registros listados a solo aquellos que están publicados? Tal como se explica en el apartado correspondiente, los *posts* publicados tienen en el campo post_status el valor de 'publish'. Así pues, elegiremos aquellos *posts* con post_status 'publish' y post_type 'post'. Esta sería la consulta SQL:

```
SELECT wp_posts.post_title
FROM wp_posts
WHERE post_status = 'publish' AND post_type = 'post';
```

Para hacer la consulta con PHP y en WordPress, deberíamos escribir el siguiente código:

```
global $wpdb;
```

```
$query = "SELECT $wpdb->posts.post_title
FROM $wpdb->posts
WHERE $wpdb->posts.post_status = 'publish' AND $wpdb->posts.post_type = 'post'";
$resultado = $wpdb->get_results($query,ARRAY_N);
```

5.2. Acceder a los datos de dos tablas

Listar datos de una sola tabla, como hemos visto, resulta muy fácil. Un poco más complicado resulta hacerlo de dos tablas. Pero no mucho más.

Para acceder al contenido de dos tablas necesitamos saber a través de qué campos se establece la relación entre ellas y para ello usaremos las tablas de relaciones del apartado anterior. Vamos a listar los títulos y la fecha de los *posts* publicados y las fechas de todos los comentarios que se hicieron a cada *post*.

Como vimos, las tablas wp_posts y wp_comments establecen su relación a través de los campos ID y comment_post_ID respectivamente. Para listar los campos que nos interesan usaremos la siguiente instrucción SQL:

```
SELECT wp_comments.comment_date, wp_posts.post_title, wp_posts.ID
FROM wp_comments , wp_posts
WHERE wp_comments.comment_post_ID = wp_posts.ID;
```

Para afinar un poco más, podemos añadir las condiciones que eliminan los *posts* sin publicar y las páginas con comentarios (si existen):

```
SELECT wp_comments.comment_date, wp_posts.post_title, wp_posts.ID
FROM wp_comments , wp_posts
WHERE wp_comments.comment_post_ID = wp_posts.ID
AND wp_posts.post_status = 'publish' AND wp_posts.post_type = 'post';
```

Todavía queda eliminar de esta lista los comentarios no deseados (*spam*) y los no aprobados, con lo que nos quedaremos solo con los aprobados:

```
SELECT wp_comments.comment_date, wp_posts.post_title, wp_posts.ID
FROM wp_comments, wp_posts
WHERE wp_comments.comment_post_ID = wp_posts.ID
AND wp_posts.post_status = 'publish' AND wp_posts.post_type = 'post'
AND wp_cooments.comment_approved = 1;
```

Esta instrucción SQL lista la fecha de todos los comentarios de cada *post*: para un *post* que tenga cuatro comentarios, tendremos cuatro líneas, cada una con la fecha del comentario que toque, el título y el ID del *post*. Por ejemplo:

Ejemplo post con 4 comentarios

comment_date	post_title	ID
2012-08-10 16:35:01	Haciendo pruebas con WordPress	2
2012-08-10 20:40:45	Haciendo pruebas con WordPress	2
2012-08-11 00:17:41	Haciendo pruebas con WordPress	2
2012-08-11 01:05:07	Haciendo pruebas con WordPress	2

5.3. Agrupando datos y contando

SQL permite operaciones mucho más elaboradas de las que hemos visto hasta ahora. A continuación, vamos a ver un par de ejemplos algo más complejos a partir del que elaboramos en el apartado anterior.

Funciones en MySQL

Podemos ver una lista de funciones disponibles en MySQL en la siguiente dirección: http://dev.mysql.com/doc/refman/5.0/en/func-op-summary-ref.html.

Vimos la manera de listar todos los comentarios de todas las entradas, escogiendo incluso solo aquellos que estaban aprobados. Ahora, vamos a contar el número total de comentarios **aprobados** que hay en el blog. Para ello, usaremos la función **count (campo)**, que cuenta el número de registros en los que el campo no es nulo.

```
SELECT count(*) FROM  wp_comments
WHERE wp_comments.comment_approved = 1;
```

Usar la función count() hace que no se listen los registros, sino que solo se presente el resultado final.

Pero saber el número total de comentarios que hay en el blog no nos es de mucha utilidad. Seguramente, sea más útil (daría una especie de clasificación de popularidad de los *posts*) un listado que nos diga cuántos comentarios tiene cada entrada. Esta sería la consulta SQL que nos listaría ID, título y número de comentarios de todos los *posts*:

```
SELECT ID, post_title, count(*) FROM wp_posts, wp_comments
WHERE comment_approved = 1
AND comment_post_ID = ID
AND post_status = 'publish'
AND post_type = 'post'
GROUP BY ID;
```

GROUP BY, como es fácilmente imaginable, agrupa por el campo que le digamos, en este caso ID, con lo cual se listará ID, título y número de comentarios. Al agrupar, no recibimos un solo valor, como en el ejemplo anterior, sino que obtenemos un valor por cada ID.

Siguiendo con el ejemplo, podríamos ordenar los *posts* por número de comentarios, con lo que obtendríamos nuestra clasificación de popularidad:

```
SELECT ID, post_title, count(*) FROM wp_posts, wp_comments
WHERE comment_approved = 1
AND comment_post_ID = ID
AND post_status = 'publish'
AND post_type = 'post'
GROUP BY ID
ORDER BY count(*) DESC;
```

ORDER BY ordena por el campo que digamos (en este caso count()) y DESC indica que ordenará de manera descendente, con el *post* con más comentarios primero. Para acabar, podríamos limitar a cinco el número de registros mostrados añadiendo LIMIT 5 al final.

5.4. Categorías y etiquetas

Posiblemente, las consultas más complicadas que podemos hacer a la base de datos de WordPress son las relacionadas con categorías y etiquetas. Como para listar el título de un *post* con la categoría o categorías a la que está asignado necesitamos pasar por dos tablas intermedias, las consultas son más largas.

Un ejemplo sencillo sería listar los títulos de todas las entradas con sus categorías:

```
SELECT post_title, name
FROM wp_posts, wp_term_taxonomy, wp_term_relationships, wp_terms
WHERE ID = object_id
AND wp_term_relationships.term_taxonomy_id = wp_term_taxonomy.term_taxonomy_id
AND wp_term_taxonomy.term_id = wp_terms.term_id
AND taxonomy = 'category';
```

Esto listaría todos los registros de la tabla *posts* con una categoría asignada. Vale la pena tener en cuenta algunos detalles de esta consulta:

- taxonomy es un campo de la tabla wp_term_taxonomy. Al compararlo con 'category' automáticamente listamos los *posts* y dejamos fuera los enlaces.

- Si no hay confusión posible al especificar el nombre de un campo, no hace falta indicar a qué tabla pertenece. Sin embargo, en esta consulta hay varias

tablas que comparten nombres de campo, con lo cual es imprescindible indicar la tabla.

- Con esta consulta, si hay *posts* con más de una categoría, su título aparecerá repetido tantas veces como categorías tenga.

Podríamos contar cuántas entradas hay de cada categoría:

```
SELECT name, count(*)
FROM wp_posts, wp_term_taxonomy, wp_term_relationships, wp_terms
WHERE ID = object_id
AND wp_term_relationships.term_taxonomy_id = wp_term_taxonomy.term_taxonomy_id
AND wp_term_taxonomy.term_id = wp_terms.term_id
AND taxonomy = 'category';
GROUP BY name;
```

Aquí faltaría escoger los *posts* publicados (ahora mismo lista todos los *posts*).
Por otra parte, hacer lo mismo para las etiquetas sería muy sencillo...