HOW WORDPRESS USES MYSQL

Broadly speaking, WordPress can be divided into two segments:

1. The logic and templates that generate the look and feel of a WordPress site.
2. The MySQL database that stores all of the content and powers it all.

But what is MySQL?

MySQL is incredibly powerful, but the stock database that powers WordPress also happens to be lightweight and robust. One of the best parts of a powerful CMS like WordPress is that it handles all of the database management for the user and abstracts it away from the end user. WordPress users, and even designers, don't have to be intimately acquainted with how the database works. However, if you're interested in really understanding how WordPress works, this post is for you.

First, let's talk about what MySQL is. Specifically, MySQL is a relational database management system, or RDBMS for short. An RDBMS is a container designed to handle and run Structured Query Language (SQL). SQL is what powers a database — it controls the database's structure and form, and it also records insertions, deletions, modifications, and so on that are performed

How does MySQL work?

That’s all a bit technical, though, so let’s take a step back and consider what an SQL database would look like.

SQL is organized into tables of information. Broadly speaking, think of a table as a spreadsheet in Excel. Rows and columns store information according to a predetermined structure. However, unlike an Excel spreadsheet, the columns (called keys or sometimes fields) of an SQL database are predefined and categorized in advance.

The defined structure, or columns, of tables in WordPress are set in advance — they won’t change as a site grows unless a plugin or core update specifically changes them. A column is not only defined in advance, but its type is also defined before records are put into it. Types such as numbers (INT for integers) word boxes, (TEXT or VARCHAR for text fields), and others (DATETIME for date and time) are preset in the database. If WordPress (or a hacker) attempts to add data to the database that doesn’t match the pre-specified pattern, the database won’t accept the data.

Another huge feature of SQL databases is found in the RDBMS acronym: the relational feature.

SQL tables and even specific keys in a table have the ability to relate to other tables and keys. This allows users to build databases that relate tightly to one another, which means better organization and greater efficiency.

Let's say you want to build a database of tweets to look at later. You could build a table that contains information about specific tweets that you've saved, such as the tweet's content, how many favorites it has, and so on. You'd also like to save data on the person who tweeted it — what their name is, how many followers they have, things like that.

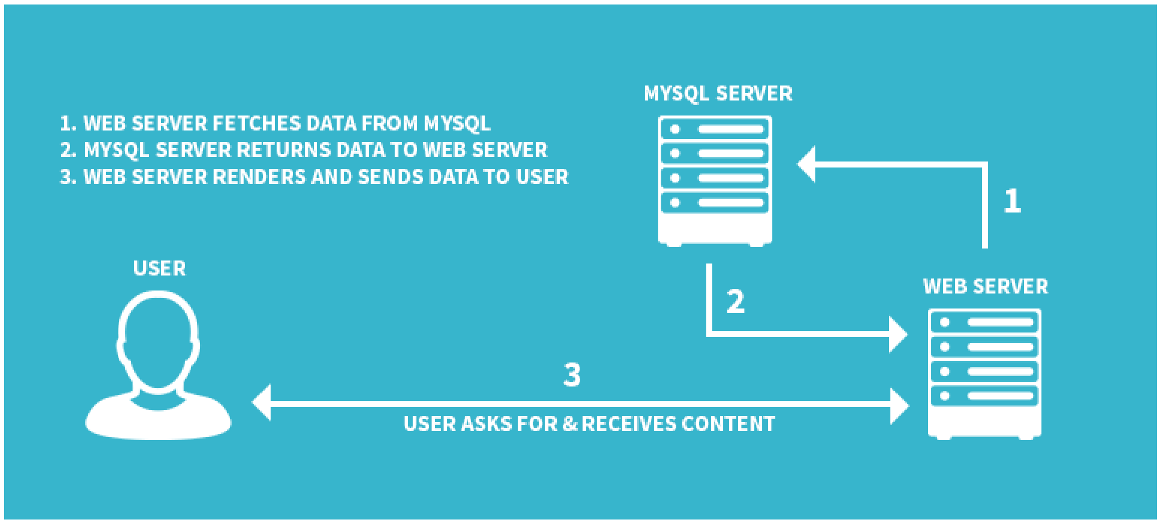
Using relational databases, you can create a tweet table that contains all data about specific tweets, and then you can create a user table that contains all of the information about the person who sent the tweet in question. You could then relate the keys in the two tables, so the database (and therefore you) knows which user is associated with which tweet.

WordPress utilizes relational tables to relate a lot of data. For example, the table wp_posts contains all of the relevant data about a single post on WordPress. The table wp_comments contains every comment anyone has left on a WordPress post, but wp_posts and wp_comments are two different tables on the same database. WordPress builds a relationship between certain keys in the wp_comments table and the wp_posts table so that WordPress can figure out which comment belongs on which blog post.

This database and table structure is built within a MySQL database when you initially set up WordPress. From there, WordPress just manipulates the values, or rows, in each table to make your site operate.

For example, when you create a new user, a row containing all of the data about the user (such as username, password, and permission level) is inserted into the wp_users table. When a user logs in, WordPress accesses the database and checks the information provided at the login screen against the database. If they don’t match, the login is rejected.

How MySQL uses caching


1. WEB SERVER FETCHES DATA FROM MYSQL
2. MYSQL SERVER RETURNS DATA TO WEB SERVER
3. WEB SERVER RENDERS AND SENDS DATA TO USER
MYSQL SERVER
1
2
WEB SERVER
USER
3
USER ASKS FOR & RECEIVES CONTENT

FLYWHEEL

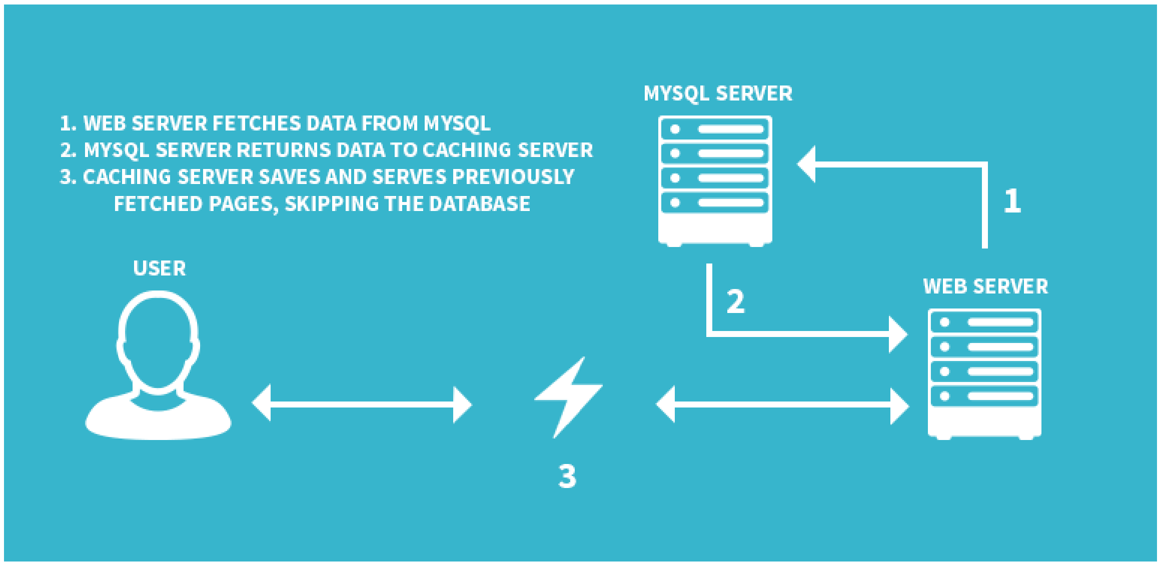
WordPress & MySQL without caching.

This same concept applies to posts: When you write and save a new post, WordPress saves it as a row in the wp_posts database. It's important to note that the inverse of this is also true: When a person viewing a website accesses a page with content on it, WordPress has to access the database, get the data out of the tables, and then render it on their screen. For a few visitors, a normal server can handle this load just fine. But if a lot of people are requesting content from the website, WordPress and the database won't be able to keep up with all of the requests and the site can fold. That's why using a caching engine or plugin is so important.

To get a bit technical again, it's important to understand computational cost. In this case, cost doesn't mean spending money or buying things. It means evaluating several different factors that are important to the speed of computing, such as time, available memory, and number of disk operations that need to be performed.

MySQL stores all of this information on a database on the hard drive of the computer it's stored on. This is a really great system because it's resistant to failure, has a lot of storage space, and doesn't destroy the memory of the computer it's running on. Compared to a database like Redis which stores everything in the RAM of the computer it's being run on, MySQL offers a lot of stability and takes away the worry of data loss.

However, the stability of MySQL comes with a cost: speed. Having WordPress continually access the same keys in a database over and over should be looked at as "expensive." It costs a lot to access the database, perform a disk operation, bring that data back, then render it.

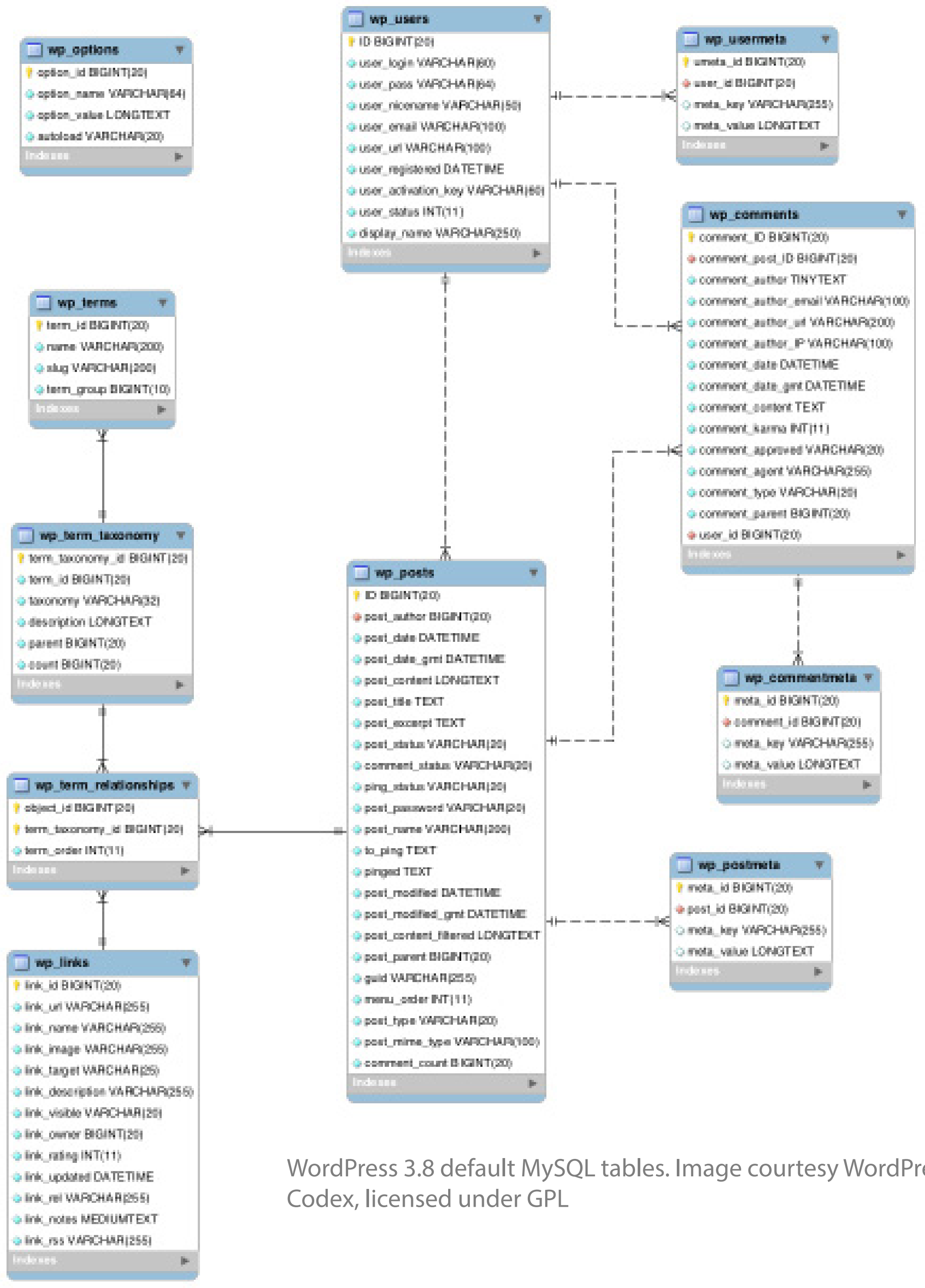

1. WEB SERVER FETCHES DATA FROM MYSQL
2. MYSQL SERVER RETURNS DATA TO CACHING SERVER
3. CACHING SERVER SAVES AND SERVES PREVIOUSLY FETCHED PAGES, SKIPPING THE DATABASE
MYSQL SERVER
1
2
WEB SERVER
USER
3

FLYWHEEL

WordPress & MySQL with caching. Note: the web, MySQL, and caching servers are generally run concurrently on the same server. This diagram separates them out for illustration purposes.

Caching is a great solution to this problem. It takes that final rendered HTML and holds on to it for a short period of time, maybe 30 seconds or so. If someone requests that same page, the caching engine will serve that already rendered page to them, skipping over the entire process of interacting with the database. That's why it's crucial to either install WordPress caching plugins or use a managed hosting service that handles caching for you when building websites. If a post ends up on the front page of Reddit, for example, your site will fold under the pressure of all the new traffic without caching installed.

The eleven MySQL tables of WordPress



WordPress 3.8 default MySQL tables. Image courtesy WordPress Codex, licensed under GPL

Now that we understand how some of these databases and tables actually work, let's talk about the eleven specific tables that WordPress establishes and uses by default:

wp_commentmeta – Any metadata associated with comments, such as keys and values, are stored in this database. It does not hold metadata such as author and date submitted, but rather miscellaneous data that WordPress occasionally uses. This data is sometimes optional or not used. It relates directly to wp_comments.

wp_comments – This table contains all comments made on WordPress posts and pages, as well as all associated data like the author, their email address, the date submitted, and the post where the comment was left.

wp_links – This database is used to contain link data in WordPress posts, but it's been retired as of a few WordPress versions ago. Although it's still there, it's doubtful you'll see this one being used.

wp_options – Any options that have been set in the settings panel are stored in this database.

wp_postmeta – Like wp_commentmeta, wp_postmeta includes optional data about posts. Unless there is a specific use case, this one isn't likely to be heavily utilized.

wp_posts – wp_posts contains all of the data about posts and their associated data. This table is very heavily used and contains all of a site's content.

wp_terms – This table stores content such as tags and categories that posts are classified with.

wp_term_relationships – This table is responsible for maintaining the relationships between posts and their associated categories and tags. There is a MySQL relationship set up here as well: the table is actually connected to both posts and terms via the wp_ term_taxonomy table.

wp_term_taxonomy – This handles tracking what type of taxonomies are associated with posts. It links back to wp_terms as well as wp_term_relationships and just logs associations with categories, tags, and so on.

wp_usermeta – Another meta table, wp_usermeta handles any optional metadata associated with wp_users.

wp_users – wp_users contains all of the data pertaining to WordPress users, including info such as usernames, admin rights, and encrypted passwords.

MySQL can seem intimidating at first, but with WordPress, the tables are laid out in a way that not only makes a lot of sense but also promotes speed and efficiency. The next time you're working on a WordPress site, think about the structure of the MySQL database powering the site. A thorough understanding of the site's database can make sure your site is secure, rock solid, and extremely efficient.

What is Flywheel?

Flywheel is a delightful platform that empowers designers, developers, and digital agencies to focus on what they do best — building beautiful, functional sites for their clients. We make it a breeze to create and develop WordPress sites, handle hosting, manage projects, and ultimately scale your business.

Stop wasting time on server management, security plugins, caching, and all those other boring repetitive tasks that take your focus away from growing your business and jeopardize your relationship with clients. Get Flywheel and get back to doing what you love.

LEARN MORE

FLYWHEEL