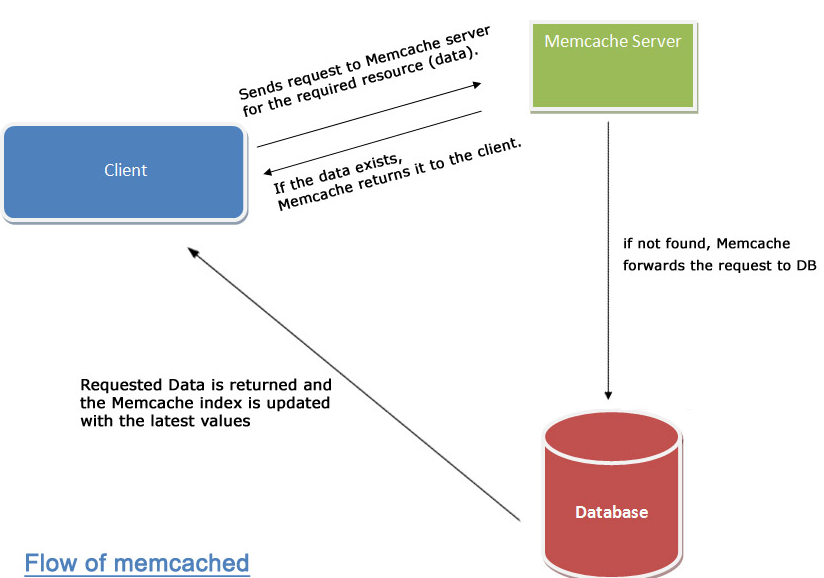
**Getting Started with Memcached Distributed Memory Caching**

Wikipedia describes [Memcached](http://memcached.org/) as a general-purpose distributed memory caching system, but what exactly does the term Memcached mean? Cache is memory used to store the most frequently used resources (e.g. browsers store every website visited during a session in cache), because accessing resources from a cache is faster than accessing them from a disk drive. So Memcached means "memorycached," which simply is caching resources in the memory. These resources can be data retrieved from API calls, database operations or even HTML pages. The data is stored in key/value pairs in the form of large hash tables.

As distributed system is part of the Memcached definition, you can install Memcached on various servers to make a larger caching server. In this way, Memcached helps reduce database loads to a minimum, resulting in faster and more responsive Web applications. Figure 1 shows how Memcached works when used with database.

How it works when used with database



### How Does Memcached Work?

Figure 1 will be familiar to anyone who has ever written a script that interacts with a database. Here is the step-by-step explanation of how it works when you want to pull certain data from the database:

1. Check whether the desired data exists in the cache. If it is in the cache records, then just retrieve the data and hence there is no need to query the database.
2. If the data that you are looking for is not in the cache, then query the database. Return the required data to the script and store the information in the cache.
3. Keep the cache fresh. Whenever the data is changed (i.e. altered or delete for some reason), update this information into the cache. That way, when the cache is queried for the old data then it should either redirect to the database or give the updated information.
4. Apparently, Memcached is best implemented for queries that are triggered multiple times in a second and demand huge data as output. Access to Memcached data is faster than the access time to disk drives because the Memcached data is stored in temporary memory.

### How to Install Memcached?

You will find Memcached preinstalled on almost any production server. But if you are trying Memcached for the first time, the first step is to install the Memcached extension on your server. First, check whether Memcached is already installed on your system using this command:

memcached -h

If Memcached is installed, the command will output the current version. Otherwise, it will return an error

The next step is to install Memcached. The following command installs Memcached on a CentOS Linux distribution.

yum install memcached

It will search the packages and install the latest version of Memcached if not install already.

### When -- and When Not -- to Use Memcached

When should you use Memcached and when should you avoid it? By now you might have figured that Memcached was designed to shed the load from the database. But you should implement this cache system with a strategy to ease more expensive queries as well. You can also write a log that records the query execution time, which will then help you to analyze performance.

For instance, suppose you operate an e-commerce site. You can cache a query that fetches the description, shipping options and availability of different products by firing a complex query (involving joins, etc.). So every time the product page is loaded the database query is skipped and the result is fetched from cache. Certainly, caching a query like this will improve the overall performance of the webpage. You would just need to update the product details when they change.

Sometimes caching your queries is not a good idea, however. This may be the case when a database operation involves more update queries than fetch queries. So, every time the database is updated, you need to update the cache as well, which causes the overall performance of the Web application to drop. Hence, querying the database becomes a better solution than using a cache solution.

### Memcached and Security

If you follow how Memcached works, you might have noticed that the data that gets stored in the cache resides in the memory and the memory does not have any access-level control over it. But if your data is not crucial, you should not worry much about security. However, you can secure your cache data in the following ways if you need to:

1. **Choosing unique keys:** Because data stored in Memcached can be taken as a large associative array, you should use unique keys. The only way you can access data is by keys and there is no other way to query Memcached. This limitation can be used to your advantage by choosing unique keys that are hard to guess. You can name the keys using some unpredictable convention.
2. **Securing Memcached server:** A server that has Memcached installed and is queried remotely should be behind a firewall. You can define an access-level control in the server, which allows access to only authorized machines.
3. **Encode your data:** You can encode your data/key or even both before storing them in the cache. That involves an extra CPU cycle and is sometimes expensive in the case of larger data, but you still can give it a try and check the performance.