## Prevention & Patching

Now that we have seen how to exploit race condition vulnerabilities let us discuss how to prevent them. Since race conditions can arise in different contexts, prevention depends on the concrete vulnerability. For instance, if the race condition arises due to simultaneous file accesses, it can be prevented by implementing file locks similar to the PHP session file locks. In our case, the race condition exists because of simultaneous database accesses from multiple threads. To prevent this, we need to implement SQL locks. They work similarly to file locks. There are READ locks which allow the current session to read the table but not write to it. Other sessions are still allowed read access to the table but write access is prevented. Furthermore, there are WRITE locks that allow the current session read and write access to the table and prevent all access to the table by other sessions. Thus, our race condition can be prevented by obtaining a WRITE lock on the users table since the user's balance is updated and a WRITE lock on the active\_gift\_cards table since the gift card code is removed. We can achieve this by executing the following SQL query:

Code: sql

LOCK TABLES active\_gift\_cards WRITE, users WRITE;

After the code has been redeemed, we can release the locks by executing the following query:

Code: sql

UNLOCK TABLES;

This prevents simultaneous access to the database by multiple threads, thus preventing the race condition vulnerability. For more details, check out the SQL documentation on locks [here](https://dev.mysql.com/doc/refman/8.0/en/lock-tables.html).