# Modifying TPC-C Workload Script to use all Warehouses

This guide has been contributed by Wes Vaske and explains how to modify the TPC-C workload script in order that the virtual users change home warehouses to use all of the data set available. The original post can be found here.



---

## *Context*

The original TPC-C specification is quite different from the default HammerDB behavior. In the original spec, each user has an amount of time that they wait between transactions (Key and Think times). HammerDB supports this operating mode but using it increases the number of users required to saturate a database to ridiculous numbers (a 2S server with SSD storage can generally support more than 100,000 users).

When using key and think times, a test will generally be configured that multiple users are assigned to each warehouse with the spec listing 10 as the max users per warehouse.

Most of us run HammerDB with key and think times disabled which results in far fewer users saturating a database. One side effect of doing this is that very little of the database is actually accessed. If you create a database with 1,000 warehouses (roughly 100GB) but only use 32 users then only 32 warehouses will see significant load (about 3GB of the total dataset).

So even if you sized a database such that it's much bigger than memory, if you don't use enough users the active dataset will still be quite small.

I've modified the original workload script such that all warehouses will be accessed by a user and each user will have a list of warehouses to access.

---

## *Details*

In the run script:

Near the top I add the flag:

```
set allwarehouses "true"
```

Inside the switch -> Default section, just prior to "set w_id". This section will create a list of warehouses for each user such that all warehouses are assigned to a user and each warehouse has only a single user.

```
if { $allwarehouses == "true" } {
```

```
    # List of warehouses for each user if we want all warehouses
    # to be used in the test
    set loadUserCount [expr $totalvirtualusers - 1]
    set myWarehouses {}
    lappend myWarehouses $myposition

    set addMore 1
    while {$addMore > 0} {
        set wh [expr $myposition + ($addMore * $loadUserCount)]
        if {$wh > $w_id_input} {
            set addMore 0
        } else {
            lappend myWarehouses $wh
            set addMore [expr $addMore + 1]
        }
    }
}
```

Inside the execution loop, at the start of each execution we change the w_id to a new random warehouse from the user's list. This ensures that each warehouse is hit.

```
if { $allwarehouses == "true" } {
    # Change warehouse for each transaction
    # if we're using all warehouses
    set myWhCount [llength $myWarehouses]
    set w_id [lindex $myWarehouses [expr [RandomNumber 1 $myWhCount] -
1]]
}
```

---

## *Summary*

With these changes, the entire dataset will be the active dataset. The term "user" has become a little bit of a misnomer but I consider this an implementation of Connection Pooling at the application level.