In this diagram, the `MySQL Source` holds the source database, the `MySQL Replica` hosts are replicas, and the `Web Client` machines are issuing database reads and writes. Web clients that issue only reads (and would normally be connected to the replicas) are not shown, as they do not need to switch to a new server in the event of failure. For a more detailed example of a read/write scale-out replication structure, see Section 17.4.5, “Using Replication for Scale-Out”.

Each MySQL replica (`Replica 1`, `Replica 2`, and `Replica 3`) is a replica running with binary logging enabled, and with `--log-slave-updates=OFF`. Because updates received by a replica from the source are not logged in the binary log when `--log-slave-updates=OFF` is specified, the binary log on each replica is empty initially. If for some reason `MySQL Source` becomes unavailable, you can pick one of the replicas to become the new source. For example, if you pick `Replica 1`, all `Web Clients` should be redirected to `Replica 1`, which writes the updates to its binary log. `Replica 2` and `Replica 3` should then replicate from `Replica 1`.

The reason for running the replica with `--log-slave-updates=OFF` is to prevent replicas from receiving updates twice in case you cause one of the replicas to become the new source. If `Replica 1` has `--log-slave-updates` enabled, which is the default, it writes any updates that it receives from `Source` in its own binary log. This means that, when `Replica 2` changes from `Source` to `Replica 1` as its source, it may receive updates from `Replica 1` that it has already received from `Source`.

Make sure that all replicas have processed any statements in their relay log. On each replica, issue `STOP REPLICA | SLAVE IO_THREAD`, then check the output of `SHOW PROCESSLIST` until you see `Has read all relay log`. When this is true for all replicas, they can be reconfigured to the new setup. On the replica `Replica 1` being promoted to become the source, issue `STOP REPLICA | SLAVE` and `RESET MASTER`.

On the other replicas `Replica 2` and `Replica 3`, use `STOP REPLICA | SLAVE` and `CHANGE REPLICATION SOURCE TO SOURCE_HOST='Replica1'` or `CHANGE MASTER TO MASTER_HOST='Replica1'` (where `'Replica1'` represents the real host name of `Replica 1`). To use `CHANGE REPLICATION SOURCE TO` | `CHANGE MASTER TO`, add all information about how to connect to `Replica 1` from `Replica 2` or `Replica 3` (*`user`*, *`password`*, *`port`*). When issuing the statement in this scenario, there is no need to specify the name of the `Replica 1` binary log file or log position to read from, since the first binary log file and position 4, are the defaults. Finally, execute `START REPLICA | SLAVE` on `Replica 2` and `Replica 3`.

Once the new replication setup is in place, you need to tell each `Web Client` to direct its statements to `Replica 1`. From that point on, all update statements sent by `Web Client` to `Replica 1` are written to the binary log of `Replica 1`, which then contains every update statement sent to `Replica 1` since `Source` stopped.

The resulting server structure is shown in Figure 17.5, “Redundancy Using Replication, After Source Failure”.