Less frequently changing & often accessed data is usually kept in cache to avoid expensive database query operations. This infrequent changing data has to be refreshed at constant intervals so that the cache doesn't go stale.

Possible strategies of cache maintenance could be of three types :

- Write Through Cache(writing to cache whenever a DB operation happens)
- Pull Mechanism (Application Polls at regular intervals to update cache)
- **Push Mechanism (Database notifies application when a record has changed)**

All the above approaches are popular , here in this blog we will be discussing the **third** approach in detail.

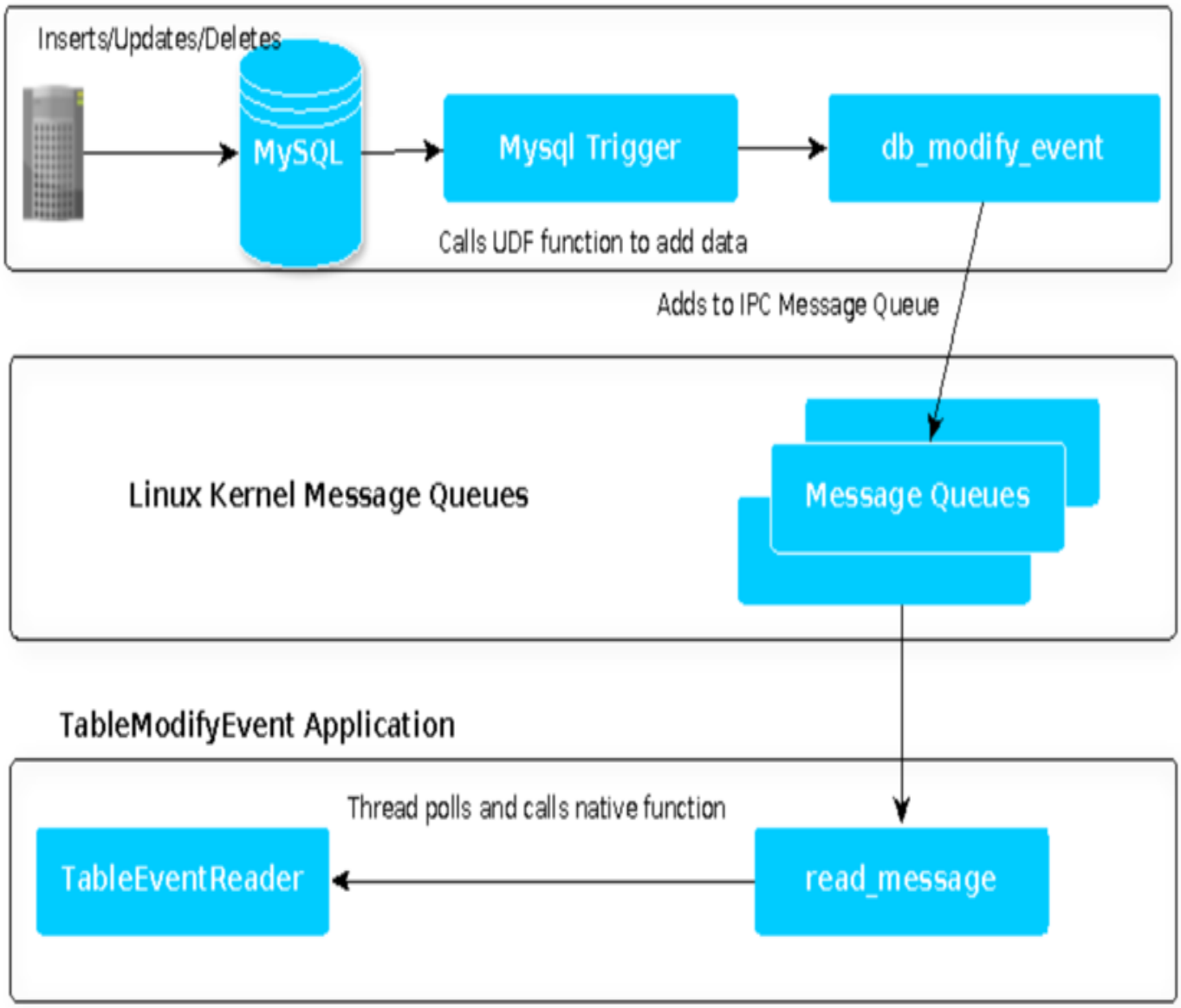
Before jumping into it , I am assuming you have basic understanding on IPC, if not , have a look on the article by Jean-Sébastien Herbaux on IPC

> *This example uses MySQL (RDBMS) and Linux Message IPC Queues and then sending it as Notification using Rabbit MQ AMQP to which multiple application listeners consume.*

Let's talk Code

Source code of the TableModifyEvent project is as follows :

Lets see the flow diagram of this TableModifyEvent Application first :



You require two classes namely TableEvent and TableEventReader which uses Java Native Access (jna.jar) native code for reading the contents of IPC message queue and forward it to our Rabbit MQ based AMQP Queue.

Secondly you will require to set up MySQL UDF's(User Defined Functions), as in the example tme.c & db_modify_event.c . Both these c programs need to be compiled as shared files (.so files) and added to mysql plugins folder.

Further More , you will also require a trigger to be created on the Database table on update/insert/delete to send data to IPC message queue.