# ADVANCED KDB – ANSWERS

Q1 2 3 4 Tickerplant

adv\_kdb/config/env.sh – port assignment as well as QHOME/Q are all defined in here, please update to suit local machine q location etc as well as selected ports.

Location of sym.q tick.q r1.q r2.q cep.q etc **- /adv\_kdb/src**

For TP logging every minute output **/adv\_kdb/logs/data/tp.log**

Q5 Logging

Location - **/adv\_kdb/common/logger.q**

Output location - **/adv\_kdb/logs/connections/tick.log**

The logger is loaded into the tickerplant as an example – can be loaded into any component.

Q6 Startup Shutdown scripts

Decided to create two separate scripts start.sh and stop.sh (can be combined into one if necessary).

Upon starting a process, the pid for the process will be stored in **/adv\_kdb/logs/pids**

Run commands:

**./start.sh ALL** – Starts all components (ps –ef | grep “uname” to see details)   
**./start.sh ONE** **TP –** Starts only the tickerplant etc

**./stop.sh ALL** – Stops all components   
**./stop.sh ONE FH** – Stops only the feedhandler etc  
**./stop.sh TEST** – Tests for active connections to see what potentially needs stopped

Q7 TickerPlant log Replay

Location - **/adv\_kdb/common/new\_sym\_file.q**

Run command - **q new\_sym\_file.q symYYYY.MM.DD**

Output Location - **/adv\_kdb/logs/raw/symYYYY.MM.DD\_IBM**

Q8 Create a script that will load a csv file and publish the contents to the Ticker plant.

Location: **/home/mcelvogc/adv\_kdb/common/csv\_reader.q**

Run Command: **q csv\_reader.q**

Output Location – look at quote table on port 5511.

Notes – TP on 5510 must already be started, csv file location manually hardcoded. Stop feedhandler using ./stop.sh ONE fh and clear quote table on 5511 to easily see load.

Q9 Write a script which will take the Ticker Plant log and create a daily partitioned HDB in which all columns are compressed with the exception of sym and time.

Location: **adv\_kdb/common/compress\_hdb.q**

Run command: **compress\_hdb.q 2019.11.14**

Output Location: **adv\_kdb/hdb/**

Notes – Symfile location and hdb are hardcoded. Date is passed as parameter.

Q10.Discuss the effect a schema change to the trade table in the above system e.g. a sequence number column was added to both trade and quote tables. How you would plan a turnover to update the schema.

* The sym.q file would firstly need updated with the new change.
* Need to consider the ordering of columns with the new change, if any scripts rely on the ordering of tables remaining constant, then these will potentially need updated to accommodate the new change.
* The schema change would require a period of downtime, either immediately after EOD or during the day.

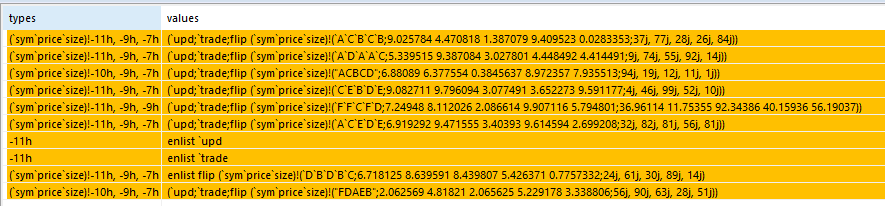
1. If during the day, the replay log file would need recreated and resaved to create a new column in the tplogs, otherwise if the RDB goes down after restart then a length error will be thrown.
2. Additionally, if during the day, the rdb/any downstream subscribers to the trade table will need to be updated in memory.
3. For EOD, with the system shut down the HDB will need modified to ensure each table in each partition is appended with the new column/new column ordering, with each partitions .d file modified to reflect the change.

* From discussion points 1,2 and 3, shutting down the system immediately after EOD will ensure minimal to no data loss pending on how many regions the system is accounting for. During this period of downtime the HDB, sym.q file can be updated, allowing subscribers to subscribe to the new table schemas upon startup.

Q1 tp\_log

There are several incorrect elements to the log file, by running the following:

([] types:({type each first (get `:tplog)[x][2]} each 0 +til 10); values: (get `:tplog))

We get the following output:

Problems:

1. Mismatching type on sym column (some columns string)
2. Mismatching type of size column (floats)
3. Broken log entry – separated list at index [6][7][8]

Due to the very small size of the log file, my solution is to read the old broken log into memory, modify fields where required and the save down a new log file ‘new\_log’. This can be renamed to tplog if required.

**Steps:**

* Load in broken log file
* Fix separated log entry before updating data types. Important to ensure the ordering of log records remains consistent:
* 6 good initial records
* 7th recorded the combined broken record
* 8th record good initial record
* Update datatypes on sym column: each string element needs to have each char casted separately.
* Update datatypes on size column: sizes should be of type “j”.
* Set new empty log file, open handle to log file and send new data to log file.
* For correctness, define a trade table and upd function, then replay the new log file.

**Code:**

l:**get**`:tplog

new\_log : l[0+ **til** 6],(**enlist** fix\_record:l[6],l[7],l[8]),**enlist** l[9]

new\_log : {(`upd;`trade;@[**select** **from** x; **exec** c **from** **meta** x **where** t="c",c=`sym; {`$x} **each** ])}**each** (**last** **each** new\_log)

new\_log : {(`upd;`trade;@[**select** **from** x; **exec** c **from** **meta** x **where** t="f",c=`size; {"j"$x} **each** ])}**each** (**last** **each** new\_log)

`:new\_log **set** ()

h:**hopen** `:new\_log

h new\_log

trade:([] sym:`$();price:"f"$();size:"j"$())

upd:**insert**

-11!`:new\_log

Q2) The 3 tables in the below zip file contain errors. Try to ascertain the error and recommend a suitable fix for each

**Table 1 –** Missing .d file, as the tables all have the same schema, a .d file from t2 or t3 can be copied into the t1 folder.

**Table 2 :**

l1:**get** **hsym** `$"/home/mcelvogc/adv\_kdb/external\_resources/db/db/t2/price"

l1:(l1,0.0)

(**hsym** `$"/home/mcelvogc/adv\_kdb/external\_resources/db/db/t2/price") **set** l1

**Table 3:** The sym column in not enumerated, load the table into memory and replay it using .Q.en to the specified location in order to add enumeration.

Q3 In your current system you have a historical database and a single hdb kdb+ instance to host user queries. Users are complaining about slowness. What is the probable cause of the slowness and suggest 2 changes that could be made to alleviate the problem. Maintain a single entry point to the system if possible

If there is only one hdb, all users are connecting to the one kdb process. Each user query must be executed in turn so if one query is slow all other users must wait for that query to return.

 One suggestion is to have multiple hdbs pointing to the same data. You could use a gateway process with async queries to load balance. Alternatively, if using kdb 3.5 and a recent Linux kernel socket sharing can be used to allow multiple kdb instances to listen on one port. If a particular hdb is busy a new user will be connected to a different hdb in the farm.

 Another suggestion is that you segregate the data such that different hdbs point at different subest of the db. The gateway process then decides which hdb a given user will query. This means if a user runs a slow query, they are only blocking a certain portion of the data from access.

Q4 Improve the performance of function

StringtoDate:{[x]{$[10h~abs type x;x:"D"$x;-14h~ type x;x:x;`date$x]}'[x]} given input: raze 500000#enlist("2010.01.01";2010.01.02). Do not use .Q.fu.

1. StringtoDate:{@[x;where (type each x) in 10h;"D"$]}

**Exercise 3 – API**

Q1 – Python/Unix/Perl or Scala

Completed on work server using Python3

Location: **adv\_kdb/api/scripts**

Run cmd: **python3 python\_api.py**

Requirements:

* TP and RDB modules running on 5510 and psclxd00626 (work server, can be changed where required).
* Quote table with cols sym,bid,ask,bsize and asize and types (s;f;f;i;i) on TP and RDB.
* quote\_test\_api.csv in /adv\_kdb/api/csv\_files

Notes: Filepath to csv is hardcoded as is server and port in python\_api.py. Will need changed to run on tester machine

Q2) C/C++, Java, C#

Completed on home network using windows with Cygwin64 terminal.

Requirements:

* TP (5510) and RDB must be started on windows machine (In this case using anaconda prompt)
* Quote table with sym,bit,ask,bsize and asize cols (s;f;f;i;i) types on TP and RDB.
* GCC compiler for running c/c++ scripts
* External files – common.h/k.h/ c.dll ( dll is needed for windows, would be .o for linux)
* Csv file – quote\_test\_api.csv

Hostname is hardcoded to my local desktop, this line will need changed in publish.c:  
S hostname= "DESKTOP-RGJEKPE";

Run Cmd: **gcc –DKXVER=3 –o publish publish.c c.dll –lpthread**

This will compile the program and create an executable file publish.exe.

Run Cmd: **./publish.exe**

Source: Look at quote table in RDB which will be populated with the csv records

Q3) Web Interface – HTML5

Location: /adv\_kdb/api/html

Requirements:

* Two files **trade\_query.q** and **trade\_query.html**
* Sever and port in **trade\_query.html**  will need updated in this line:  
  ws = new WebSocket("ws://psclxd00626:6018/");
* Port will need updated in **trade\_query.q** to match port set above in this line:  
  \p 6018

Run Cmd: **q trade\_query.q**

After running, type server:port/trade\_query.html into browser search bar ex:

**psclxd00626:6018/trade\_query.html**

Anything can be typed into the q) window which appears, if it is not a sym which exists in the trade table, the user will be prompted with a message indicating this as well as a list of the syms which can be filtered on.