First derivatives plc

CMTP

Advanced Kdb+ Exam

NOTE: Directory paths should not be hardcoded in numerous scripts. Either have it dependent on the location it is ran from or have a configuration file loaded into each script.

**Exercise 1 - Tick**

Create a project with the following components and requirements. The basis of the exercise will be tick scripts available here:

<https://github.com/kxsystems/kdb-tick>

| Number | Component | Requirements |
| --- | --- | --- |
|  | Ticker Plant | Create a tickerplant which contains the schema for the below tables:   1. Trade 2. Quote 3. Aggregation table which will contain following metrics by symbol    1. Max/min trade prices    2. Traded volume    3. Top of book details   The tickerplant should also log every minute: instead of every minute, I included it in .z.ts   1. number of messages that have been processed by table up until that time 2. subscription details of subscribers |
| 2. | RDB | Create 2 RDB which subscribe as follows:   1. Subscribes only to Trade and Quote tables 2. Subscribes to aggregation table |
| 3. | Feed Handler | Create a mock feedhandler that will supply trades and quotes to the above system. |
| 4. | CEP | Create a Complex Event Processer/Calculation Engine which will subscribe to trades and quote from tickerplant and then calculate metrics for the aggregation table and publish the data back to the Ticker Plant. |
| 5. | Logging | Write a logging script that can be loaded into the above components and will log:   1. details of connections opened 2. details of connections closed 3. all logging statements should include username of calling process where applicable and memory usage details from .Q.w[] 4. functions should be available so that can write internal logging statements to write to standard out and error |
| 6. | Startup/shutdown scripts | Create a startup script in whichever language you prefer e.g. ksh, bash, perl etc that will work in 3 main modes:   1. START - give the ability to start up the components in 1-4. You should be able to start all or any specific one 2. STOP - give the ability to stop the components in 1-4. You should be able to start all or any specific one 3. TEST – test which components are currently running and supply details 4. Make the script as configurable as you see fit e.g. how port numbers will be assigned, log locations, db directories |
| 7. | Ticker Plant log replay | Write a script that reads in a tickerplant log file which contains trade and quote updates and creates a new tickerplant log file which only contains the trade updates for ibm.n. |
| 8. | CSV File load | Create a script that will load a csv file and publish the contents to the Ticker plant. |
| 9. | EOD Process | 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. |
| 10. | Schema Change | 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. |

**Exercise 2 - Debugging**

Debug the following

| Number | Component | Requirements |
| --- | --- | --- |
|  | TP/RDB problem | The below tickerplant log contains and error.   * + 1. Locate the error     2. Fix the error   Show how each was performed.      l: get `:tplog;  l:(6#l),(enlist raze l[6 7 8]),-1#l;    t:update {`$x} each sym from l[2][2];  l[2]: (2#l[2]),(enlist t);  t:update {`$x} each sym from l[7][2];  l[7]: (2#l[7]),enlist t;  A black sign with white text  Description automatically generated  t:update ceiling size from l[4][2];  l[4]:(2#l[4]),enlist t;  A close up of a keyboard  Description automatically generated |
|  |  |  |
|  | Splay Table problem | The 3 tables in the below zip file contain errors. Try to ascertain the error and recommend a suitable fix for each.     1. `:t1/ set .Q.en[`:.]`sym`price`size xcols flip t1 2. p: exec price from t2; p,:0.0f; `:t2/ set .Q.en[`:.] update price:p from `t2; 3. The sym column has not been enumerated on t3: `:t3/ set .Q.en[`:.]select from t3; |
| 3. | Blocking calls | 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.  The problem is probably that synchronous user queries are locking the instance until a result is returned:   1. You could add a timeout to the server, so that any overlong queries are terminated 2. You could launch another instance and add a gateway and load balancer to divide requests up between them. This would maintain a single entry point through the gateway |
| 4. | Query Performance | Improve the performance of function  StringtoDate:{[x]{$[10h~abs type x;x:"D"$x;-14h~ type x;x:x;`date$x]}'[x]} given input: raze 500000#enlist("2010.01.01";2010.01.02). Do not use .Q.fu.  {{$[-14h~ type x;x;10h~abs type x;"D"$x;`date$x]}'[x]} reduces it by about 50 ms  If the input is a date it will return it immediately, which cuts down on evaluation  Also removed unnecessary assignments  The “abs” isn’t relevant to the input, and is there to catch single characters, but if you’re looking to just improve performance without changing any functionality this will do. Removing abs reduces it a further 50ms  If you’re tailoring it specifically for the input remove the second evaluation and `date, and just use “D”$x  Which reduces it another 25 ms roughly |

**Exercise 3 - API**

| Number | Task | Requirements |
| --- | --- | --- |
|  | Create a small script in another language to accomplish requirements listed. Select language from:   1. Unix shell 2. Perl 3. Python 4. Scala | The script should read in a csv file and publish each row to a tickerplant. |
| 2. | Create a small script in another language to accomplish requirements listed. Select language from:   1. C/C++ 2. Java 3. C# | The script should read in a csv file and publish each row to a tickerplant. |
| 3. | Web Interface – HTML5 | Create a simple web interface using HTML5 that would be able to query a trade table, filtering on symbol input by user into a text box and display the results in table. |