AMA-ZING HACK CHENNAI

Team: Goto shell!

Members: Prabhu

Modules

1. Initializer:

- Sets up the master data by picking up the oldest CSV file(or a CSV file designed for this purpose) from the CSV file repository. This is done to prime the master data. After populating the data, the Epoch Time is stored in a file(*lastchecked.file*)

- The Conditions.config file is revised in such a way that we identify the parameters that occur the most at the top. We push the records with the compound conditions to the bottom.

2. Listener: (Uses **Observer Design Pattern**)

- Runs as a continuous listener process.

- Sleeps for a short time interval of say, 5 mins(or 10 mins) depending on the frequency of CSV file pushes.

- Calls a script filePicker.sh to get a list of CSV files whose last modified time happened during the Interval – which is stored in *lastchecked.file*.

- Calls the UpdateAndNotify procedure to update the master data with new information.

3. UpdateAndNotify:

- Receives the updated CSV file list from Listener

- Checks the notification table and if any subscriber is to be notified,
triggers the alerting mechanism.

- Updates the master data with the updated data from the modified CSV files.

-Also periodically writes data to disk which will be utilized by the **QueryResponder**. The periodicity is determined by the customer request volume.

QUESTION: **It is not clear whether the updated attribute values in the files "file2.csv" to "file50.csv" are unique are not. If they are unique, we could easily spawn off threads to perform the Updates and Notification. Since each thread has complete access to a file without any data sharing, we do not have to deal with synchronization/file locking issues.**

If they are not unique, we need to understand the time-criticality of the updates made to the attributes. Updates that are time-critical should be implemented using a cron job mechanism. The

reason is that we cannot rely upon competing threads to execute a time-critical update since they can alter the sequence or cause starvation of an update. Eg. A Happy-hour sale of electronic items on Amazon for which even a few minutes delay can affect customer satisfaction.

4. QueryResponder: *<Currently receives query requests from file. A separate program* ***Requester*** *fills the file with queries>*

- Check the query queue at brief intervals and retrieve data from the snapshots created by the UpdateAndNotify module.

Efficiency Areas:

- Mechanism to make the data persistent in order to handle unexpected crashes – Maintain a snapshot after every write and delete all previously written data.
-Implement a forking mechanism to handle all incoming Query Requests.