

Alternate Means of “Data Compression”

Part IV: Compressing Data

By Thomas O’Hare
Thomas@RedTile.Com

Copyright© 2004, Thomas O’Hare – All Rights Reserved.

Overview

In the previous papers we have talked mainly about the overall concepts of getting the data we need in order to do our data “compression” operations. In this paper we will deal with the actual “compression” mechanism we need in order to drastically reduce the amount of data we will actually store for long term use.

The Mechanism

The mechanism we will use to extract only the data we need will be an external mechanism. What this means is we will use a mechanism external of the processes used to collect the initial raw data for several reasons:

- An external process uses little if no overhead from the apparatus used to collect the initial data streams. This makes for more efficient use of both the initial data collection apparatus and the mechanism used to extrapolate the only the data we need.
- Greater independence of the “data compression” apparatus, as we will call it, makes this actual apparatus portable. This means it can be used easily with any number of initial stream mechanisms if so desired.
- We can fine tune the “data compression” to meet our needs without direct impact on any external apparatus. This also means complete control of data “compression” without the need to be concerned of any repercussions to the original data collection mechanism.

Please Note: By “external mechanism”, we are generally referring to an executable application that can reside on the same PC as the initial data collection apparatus, or on another PC linked via some type of network connection.

The Technique

The technique used to gather the data we will eventually call “compressed” is basically a series of Structured Query Language (SQL) statements that poll the initial “data buffer” we described in part three of this series of papers. Since we are using a database as a buffer, we can easily and effectively execute a preset collection of SQL statements at the intervals described in part one of this series (Time & Delta).

Copyright© 2004, Thomas O’Hare – All Rights Reserved.

It is up to the external data compression apparatus to be responsible for triggering these SQL statements in the appropriate sequences. This is where the real work and technical expertise comes into play.

Obviously there are numerous aspects as to the way this data collection works, depending on the needs of the resulting data sets. The complexity of the external mechanism is also dependent upon the requirements of the resulting data set. In most cases a series of data streams are required to run in parallel. What this means is that usually many unique identifiable data points or data sources must be monitored simultaneously.

This leads us to the concept of a single central core mechanism that uses meta-data (data about data) to create the query needed to gather the resulting data set. If a single core code segment is used then it is more easily maintainable over a long period of time. This core concept means data queries can reside in external entities and be executed by the core code segment at will.

Since we use a central apparatus to do the real work, then the core “engine” is free to monitor all the required data points collectively. It can also call the appropriate extraction mechanism when it deems new data is ready to be copied into our “compressed” data file. Using this concept we will be less likely to trip over ourselves trying to keep multiple entities synchronized, especially while actually continually maintaining the actual operating code.

The Operation

The theoretical operation of the “data compression” mechanism can be thought of as the typical “tree” concept. The core code used to gather the “compressed” data can be thought of as the tree trunk. The operations spawned by the core code, or “tree trunk”, can be thought of as the branches extending from the trunk of the tree. All sub-operations, or branches, are dependent on the core mechanism of the tree trunk.

Each operation, or branch, can have its own data storage mechanism. In other words, data can be broken down into individual streams or branches and stored in separate files that can be assimilated to the leaves on tree branches.

By breaking data down into individual groups, or leaves, we can try to avoid the possibility of data bloating. By trying to avoid “bloating” we mean we can better avoid data going beyond possible artificial limits of the local storage mechanism in use. Carry this idea one step further and we can even span PCs or networks and deposit the data to its respective “compressed” storage “containers” in a distributed fashion.

Limitations

Using the concept described above, there are actually very few limitations. Matter of fact we can gain even more power over the data “compression” system if we run multiple external data collection mechanisms. Since the compression mechanism is external of

the initial data collection apparatus, we can realistically have more than one data “compression” mechanism running without the worry of taxing or disrupting the initial data collection mechanism.

With current technology, and especially the affordability of mainstream technology, we can easily implement multiple systems without causing any unnecessary drain on a PC’s resources. If the core code or “tree trunk” is maintained in a practical manner, then efficiency is better maintained as we are only maintaining one code set.

Conclusion

By utilizing an efficient external and generic “data compression engine” we gain complete control over data “compression” without negative affects on the initial raw data streams or “data buffers”.

It will be paramount to initially set up complete data “structures”, or even relational data tables, before any core code is started. Distribution of data to as many related data storage devices as possible will not only make data more manageable, but will also keep the individual data stores from becoming “bloated”. As an additional benefit, this approach will make it easier to effectively use this “compressed” data in further operations.

The substantial additional benefit of using the “external engine” concept is the ability to run multiple “data extraction” mechanisms simultaneously. Each additional external mechanism can now focus on its own specialized task. With several external mechanisms at work we gain the capability of having each external “data gathering engine” focusing on a more specialized need. So plainly put, we can break down our requirements into sub-categories and have multiple “harvesting engines” handling only that specific sub-category. Keep in mind that not only can we be more efficient by breaking tasks down across several mechanisms, but we also will raise the efficiency of each separate mechanism in the process. This is accomplished by each external mechanism now being responsible for handling only a smaller scope of activity.

Change Log Part IV:

Original: April 9, 2004.