Chunking

Introduction

Jitterbit's multi-use "chunking" feature splits the source data into multiple chunks based on the configured chunk size. The chunk size setting used is the number of source records (nodes) per chunk. The transformation is then performed on each chunk separately, with each source chunk producing one target chunk. The resulting target chunks combine to produce the final target.

Note that LDAP sources cannot be chunked and chunking cannot be used unless records are independent.

> WARNING: Using chunking affects the behavior of global and project variables. See the Considerations section below for details.

Advantages of Chunking

API Limitations

Many web service APIs (SOAP/REST) have size limitations. For example, a Salesforce.com Upsert only accepts 200 records per call. In such a case, you would configure an operation to use a chunk size of 200. The source will be split into chunks of 200 records, and each transformation calls the web service once with the 200 records. This is repeated until all the records have been processed. (Note that Salesforce has a Bulk Upsert, which Jitterbit supports with its Bulk Process wizard, to avoid the use of chunking.)

In the case of an operation writing to a target file using chunking, the resulting target files are combined into a single file.

Parallel Processing

If you have a large source and a multi-CPU computer, chunking can be used to split the source for parallel processing. Since each chunk is processed in isolation, you can use this option to process several chunks in parallel. This only applies if the source records don't depend on each other at the chunk node level. Web services can be called in parallel using chunking, improving performance. However, there are considerations, as described below in Considerations When Chunking.

Limiting Memory Use

In cases when streaming/batch transformation can't be used, you can use chunking to make the transformation use less memory. For more on streaming and batch transformation, see Stream and Batch Transformations. Streaming/batch transformation is preferred when the only issue is memory use. In that case, use as large a chunk size as possible, making sure the data for each chunk fits well inside the available RAM.

Considerations When Chunking

As chunking can invoke multi-threading (if configured to use more than one thread), its use can affect the behavior of variables that are not shared between the threads.

Global and project variables are segregated between the instances of chunking, and though data is combined, changes to these variables are not. Only changes made to the initial thread are preserved at the end of the transformation.

For example: if an operation—with chunking and multiple threads—has a transformation that changes a global variable, the global variable's value after the operation ends will be that from the **first** thread. Any changes to the variable in other threads are independent and are discarded when the operation completes.

These global variables are passed to the other threads by value rather than by reference, insuring that any changes to the variables are not reflected in other threads or operations. This is similar to the `RunOperation()` function when in asynchronous mode.

Enabling Chunking

As chunking is configured for an entire operation, you can use the same transformation in either a chunked or non-chunked mode, depending on the operation it is used in.

To enable chunking:

On This Page

Related Topics

- Operations

Last updated: Sep 17, 2018

- Create an operation and assign a transformation to it.

- Right-click on the **operation** and select **Options…**
- Check the **Enable Chunking** checkbox and enter the chunk size.
- Your chunk size depends on your use case, as described above under Advantages of Chunking.

- Enter the **Number of Records Per File**. 0 means that all records will be in one file.

- Enter a value larger than 1 in the **Max number of threads** field to enable parallel processing. (See the above Considerations When Chunking for consequences of using must-threading.)

- **Source** and **Target chunk node**:
    - For flat sources (and targets) you do not need to specify a chunk node; it will be displayed as "`-flat-`".
    - For hierarchical sources, you need to specify a **Source chunk node**.
    - For hierarchical targets, you need to specify a **Target chunk node**.

    - Click the **Select…** button to browse for a suitable chunk node. The chunk node is the repeating record (or loop node) that you want to base the chunk size on.

    - If you do not specify anything, the default is to use the highest repeating node.

Advanced Options

- If the target is a database, the target data will first be written to several files (one per chunk). These files will then be combined to one target file which will be sent to the database for insert /update.

- If you set the global data element `jitterbit.target.db.commit_chunks` to 1 (or `true`), each chunk will be committed to the database as they become available. This can improve performance significantly if parallel processing is enabled since the database insert/updates will be performed in parallel.