Batch Control System

Overview

This describes an easy method to build a batch control system. It was built originally for SQL Server but has been through many iterations since then on many platforms. Do not think of it as platform specific but in general terms, I have even implemented the processing features in VB with the database just holding the data. The main constraint on the technology is the expertise of the staff that will have to maintain and troubleshoot issues, to this end part of the requirement is to have a simple implementation with logging that will enable inexperienced staff to at least see what is happening and to make changes to the system without too much risk of errors. To this end a major part of the system is logging, error detection and performance checking.

This is meant to be a simple entry level batch control system. To this end it will maintain simple dependencies and be single threaded. It is not difficult to extend this to become an enterprise level system. This system I would expect to take a few days to implement – usually about two days to implement from scratch after analyzing the requirements of a system. How long to add the batches depends on the complexity of the system but there should be little coding involved, mainly adding entries to tables.

First we need to define what is meant by a batch control system. It is a system to control batches of work. It does not carry out the work itself but detects that work needs to be carried out, issues commands to processes to carry out that work and records the result. The batch aspect is associated with holding information about batches of work that need to be carried out. A batch will be series of steps that need to be executed in order. A dependent batch will be triggered by an external event or by the completion of a (or more than one) batch. A batch will be considered a control unit (i.e. is atomic), the batch succeeds or fails (in which case it can be re-run). There is no instance where part of a batch will be run or that a batch can be considered part successful. Dependencies will be between batches, i.e. a dependent batch will run on completion of the triggering batch completion.

High Level Requirements

Hold definitions of batches of work thread steps
Maintain dependencies between batches
Create batches on defined trigger (external or internal)
Control processing of batches in step sequence
Alert if any problems
Log everything that is processed

These high level requirements seem fairly simple and in concept a workflow controller is simple. It is important to isolate the control processing. This is where systems can increase in complexity, once control is delegated to the work items it becomes very difficult to maintain or even implement the system. It is important to make sure that a work item is called for execution by the controller and returns a result status, the control structures remain under ownership of the control system (i.e. are never queried and especially not updated by the work items)

There are a few decisions to be made which will affect the processing.
What to do on failure. Do you wait for the failure to be rectified before continuing or process further batches? For initial implementations I fail the system and continually alert until the failure is addressed – this means that no further processing is carried out. This method has advantages in that the failure could impact other batches (it shouldn't but for early in the implementation it is good to be safe), information about the error will be easy to find in the tables as it was the last thing processed, it is easier to convince people to write robust processes if the impact is greater.
Later it is advisable to add retries and to allow the system to process further batches

Types of Steps (work items)

These are specific to SQL Server but are probably applicable to most systems (perhaps under another name)
Stored Procedure
SSIS Package
Operating system call

Note: We could implement just any one of these which could implement the others – in fact an SSIS package execution will probably be implemented as an operating system call – perhaps via a stored procedure. I define these separately because they are common and important types of steps that might need bespoke interfaces.
There are others that could be implemented but I usually implement these via the above e.g. I have an SSIS package to carry out FTP operations or carry them out via an OS or SP call.
There will be a single interface to execute the work items so it is easy to add more if needed – the difference will be the parameters that need to be passed.

Detailed Definition

Structures

The structures (tables) we hold to support the processing are

FileProcess (Named for historic reasons)

This holds items to be processed and is the link into the batch control system. Whenever an item needs to be processed it is placed here. This could be due to a file arriving, a daily process being triggerd, a batch dependency on other processes being fulfilled etc. It will hold an identifier for the batch and all information for that batch to be carried out. This will not be formatted for the batch but will be a contract with the process that adds the entry to the table.

ProcessBatch

Holds an entry for each batch controlled by the system. It also holds a status (enabled, disabled) and a priority for the batch. The priority is not complex – just defines the order in which the batches are processed

ProcessControl

This holds the definition of the steps to be executed in a batch and their order. It holds the definition of the call to the process and the values to take from BatchControl to execute the process. In a simple form it will hold a stored procedure call and a reference to the parameter values held in BatchControl for that

batch. In more advanced implementations this will also hold the expected execution duration and a maximum duration value for alerting.

BatchControl

This holds the batches to be executed. It will hold one entry for each entry in FileProcess but will be in a format to enable the execution of the steps in ProcessControl. It will hold the reference of the last step completed successfully, and the status of the batch.

BatchProcessHistory

This holds the history of everything executed. It will hold the command executed, the start and end time and the result status. It is useful to record the command executed in a format that can be copied and executed manually.

Processes

Processing is carried out via stored procedures.
These control processes are executed via the agent. This could all be a single job and in fact a single stored procedure. My initial implementation is often a single job but quite soon it will become apparent that this is a very fragile implementation. It is a lot easier to maintain the system if you can guarantee that batches can be created as soon as possible, then you can see what is due to be processed and what is behind schedule.

s_ProcessCreate

The end product is to take the entries in FileProcess and create BatchControl entries. It does not have knowledge about the structure of the entry to be created in BatchControl – that is delegated to s_PopulateBatchControl. It will call s_PopulateBatchControl for each batch in ProcessBatch, whether or not the batch is disabled. The batch will be created but not processed in this case.
This process will always be very quick, it does not carry out any long processes and I usually implement it as a separate job – if it is included with the same job as that which carries out the processing it can be a long time between runs as someone will always implement a long running process at some point. I usually schedule it to run every 5 minutes.
I also often include in this the creation of time based FileProcess entries e.g. daily, weekly, monthly processes – these should really be in another stored procedure.

s_PopulateBatchControl

This is the process that creates the BatchControl entry. It needs to add all information to BatchControl that is needed by the ProcessControl entry. This is usually the only SP that may need to be changed to incorporate a new batch. If you are not happy doing that (maybe your release procedures make it difficult to make changes to SPs) it could be implemented for each batch as a dynamic sql call to an sp s_PopulateBatchControl_<BatchName> so each batch would mean tha addition of a new sp to populate BatchControl.

s_PopulateDatamart

This is the sp that controls the processing of the batches. Its name derives from the latest implementations which have been for datawarehouse systems. Again this does not execute the work items but calls s_ProcessControl for each enabled batch in turn.

s_ProcessControl

This is the process that carries out the processing of the work items. It is charge of logging execution, updating statuses and tracking batch processing.
The simplest implementation is to always process a batch to completion (success or failure) and to process all outstanding batches for that

Table implementation

FileProcess

This needs to hold the information from the triggering process.
So suggested attributes are

FileProcess_id	identity
Entity	type of entry
FileName	Information about the external entry
Status	ready, paused, complete, …
Data	Used for duplication checks or other processing
z_inserted	audit
z_updated	audit

ProcessBatch

These are the batches you want to control and will be static (ish) data
ProcessBatch_id
ProcessBatchName

Enabled	To disable a batch if issues
Seq	To control the order of the batches
RetryCount	Number of times to retry on any failure
`RetryWaitMins`	Amount of time (minutes) to wait between retries

ProcessControl

This holds information about the steps to run for each batch

ProcessBatch_id	Referencing ProcessBatch
ProcessControlSeq	Order of the step to be processed
ProcessName	Name of the step
ProcessType	Stored procedure, Package, cmdshell, …
`ExpectedDuration`	`For reporting`
`AlarmDuration`	`For alerting`

We now have varchar columns to hold information needed for the steps
You could normalise or use xml for but this is easiest I find

- Data1
- Data2
- Data3
- Data4
- Data5
- Data6

The meaning of these entries depends on the type of call
I rarely use more than a couple of these columns and often implement with 3.

Column	Stored Procedure	SSIS package	O/S command
Data1	Database	BatchName	Command
Data2	SP name	PackageName	
Data3	Parameters	Package parameters	

BatchControl

This holds the batches to be run

BatchControl_id
ProcessBatch_id
PrevProcessControlSeq — Last ProcessControl step run successfully
FileProcess_id
Status — success, complete, paused
RetryCount — number of times to retry – decremented on fail, reset on success
NextRunTime — for retry – will not execute before this time

We now have varchar columns to hold information needed for the steps

You could normalise or use xml for but this is easiest I find

- Data1
- Data2
- Data3
- Data4
- Data5
- Data6

The data that will be held in these Data columns will be specific to the ProcessBatch_id. It will be things like filepaths, filenames, dates – anything that needs to be passed to a ProcessType entry to enable processing.

BatchProcessHistory

This holds the history of everything that has been run as part of a batch.

Procedure implementation

The only SP in the batch control system that needs to have knowledge about the processing of the batch is s_PopulateBatchControl – i.e. is the only SP that is batch type dependent. The only other place that has a batch type dependency is the insert into the FileProcess table, as stated this is the interface into the batch control system so the insert into this table is considered to be outside the scope of the system but is part of the interface contract.

s_ProcessCreate

This just calls s_PopulateBatchControl for each batch in turn.

Logic

For each FileProcess entity
Do until all BatchControl entries created
Call s_PopulateBatchControl

s_PopulateBatchControl

This needs to take an entry from FileProcess and create a BatchControl entry

This needs to know the values to place in each of the Data columns on BatchControl and therefore needs to have bespoke code written for each ProcessBatch entry. It will check to see if a Fileprocess_id entry for that batch already exists in BatchControl and if not creates it.
Logic

Create BatchControl entry for entries from FileProcess that do not already exist for the FileProcess entity

s_PopulateDatamart

This controls the processing of the batches. It just calls s_ProcessControl for each batch in turn that is enabled – in order of the sequence on the ProcessBatch table.
Logic

For each enabled ProcessBatch entry in order of seq
Call s_ProcessControl

s_ProcessControl

For the batch (ProcessBatch entry) in question this gets the next batchControl entry and runs each step from ProcessBatch in turn. It gets the information to execute the step by combining the entry from ProcessBatch with the entry from BatchControl. At the start of the step it adds an entry to ProcessBatchHistory and updates it when the step completes. If successful it then carries on with the next step. It does this until all steps for that batch are complete. On completion of the batch it sets the status of the entries in BatchControl and FileProcess to complete.
It then continues for all outstanding entries in BatchControl for this batch.
Logic

```
For each enabled batch in ProcessBatch
    For each uncompleted batch in BatchControl for ProcessBatch entry
        If status <> 'Success' then error
        For each step in ProcessControl for ProcessBatch starting from
        PrevProcessControlSeq
            Get step entry from ProcessControl
            Replace values in step command from BatchControl entry
            Add entry to BatchProcessHistory
            Execute command
            If successful
                Update entry in BatchProcessHistory
                Update BatchControl entry status to success
                Update BatchControl entry PrevProcessControlStep
            Else
                Update entry in BatchProcessHistory
                Update BatchControl entry status
                Error
        If completed last step
            Update BatchControl entry status to complete
            Update FileProcess entry status to complete
```

Worked example

Here we show the entries needed to run a workflow to process a file.

File detected (external process)

Entry placed in FileProcess

Entity	MyFileType1 (type of file)
FileName	file path + name – format data_yyyymmdd_yyyymmdd.csv
Data	file name (or whatever is needed to prevent duplicates)
Status	Ready

Create BatchControl entry

s_ProcessCreate is polled (probably scheduled via agent)

call s_PopulateBatchControl with batch name MyFileType1 (hard coded).

This sees the entry in FileProcess with status ready which has no corresponding entry in BatchControl so creates one.

ProcessBatch_id	1 (from ProcessBatch, batch name = MyFile)
PrevProcessControlSeq	0
FileProcess_id	from FileProcess entry
Status	success
RetryCount	0
NextRunTime	19000101
Data1	file path
Data2	file name
Data3	start date from file name
Data4	end date end date from file name

Process file

s_PopulateDatamart is polled (probably scheduled via agent)

Uses entry from ProcessBatch to call

s_ProcessControl with batch name MyFile.

This finds the entry in BatchControl for Processbatch_id 1 with status success so processes it (successful run)

For each entry in ProcessControl for the batch starting from PrevProcessControlSeq +1

Get the ProcessControl command data for the step

Replace the entries in the command data with the values in the BatchControl entry

Execute the command