Defining your delta

Before we can even *begin* to talk about managing change in Data Warehousing (which roughly encompasses concepts such as 'historisation', 'time-variance' and 'slowly changing') we need to define what a change in data is. A change, as in Change Data Capture (CDC), is essentially the differential between a source and target – the delta of records to process further. This data delta of records between one area in the architecture and the next will subsequently only be referred to as the 'delta'. This post intends to explain that what this delta is depends on the source and target structure.

In a typical interface, the CDC mechanism from a source table to the Staging Area provides the full row of the information including the (updated) attribute that caused the interface to trigger. There are many ways to achieve this, but for the purpose of this post it is sufficient to say that this mechanism (again, typically) delivers the delta into the Staging Area where the records are time-stamped with a Load Date / Time stamp. The design decision to perform this time-stamping here is critical, and many other posts have providing rationale for this – so I won't do it here again.

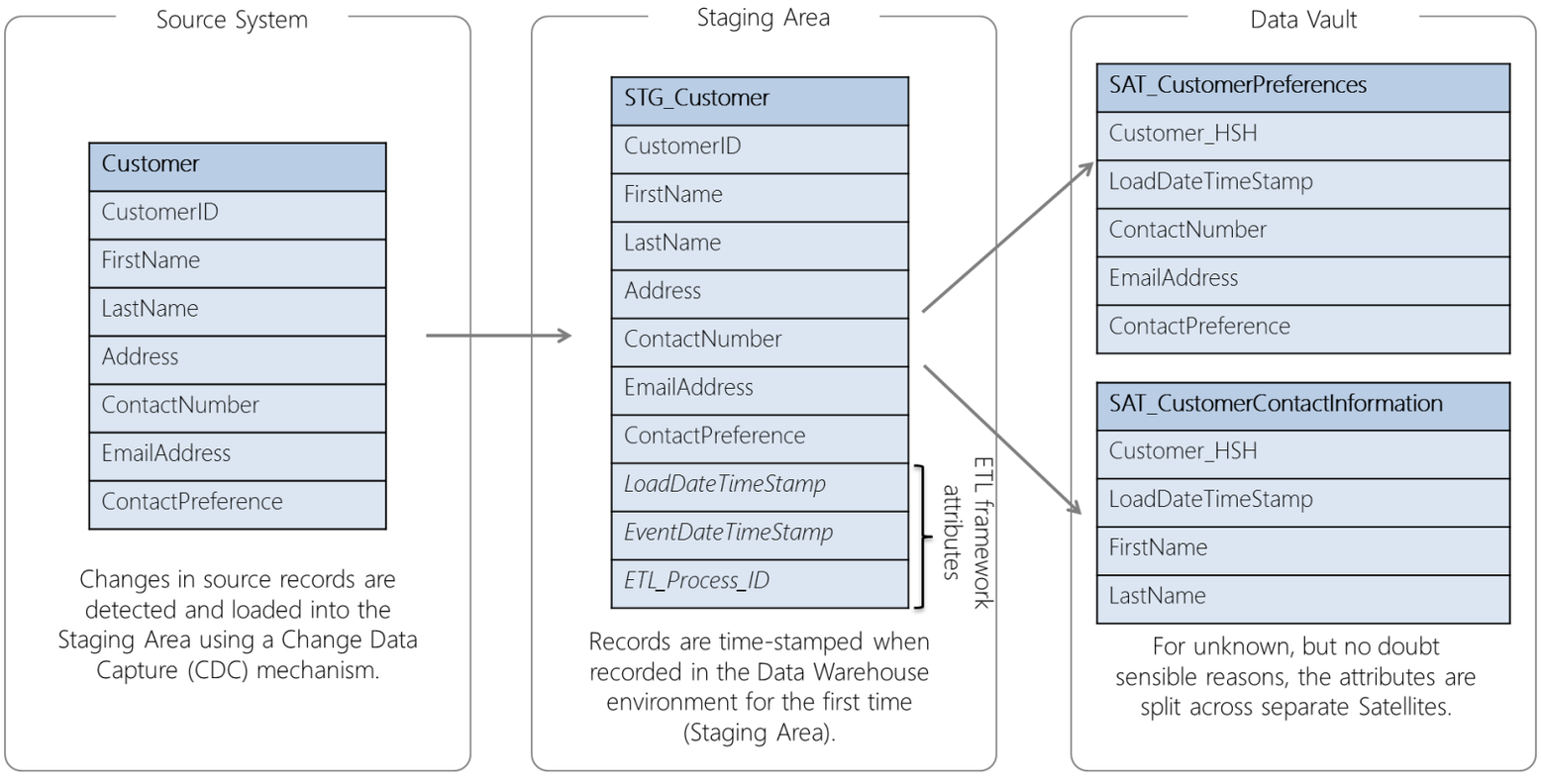
Once the data delta is available in the Staging Area of the Data Warehouse it can be moved to the various subsequent areas of the Data Warehouse model and architecture – directed by the data model and ETL metadata (e.g. source-to-target mappings). These mappings, and their impact on the scope of a 'change', is what this post is about. Subject to modelling decisions, the delta in the Staging Area will basically be broken up into smaller segments *relevant to their target tables*. For instance, 'First Name' and 'Last Name' may go into a Customer Satellite table while 'Contact Preference' may be directed to a Contact Preference Satellite table. The extent of this 'moving around of attributes across the model' depends on the degree of normalisation in the source (OLTP) system and various Data Warehouse modelling decisions made.

The practical implication of the above is that, depending on which segment of the record is mapped (the specific attributes), the delta may not be a change for the corresponding target table. Another way of saying this is that changes in records are always relative to the *scope of attributes* in the target table. A detected delta of information may be used in none, one or more targets.

'Change' management

This may sound a bit abstract, and is probably better explained with some examples. Please consider the scenario below where a hypothetical source system table named 'Customer' is required to be integrated into the Data Warehouse. As per standard design, changes in the source table are temporarily stored in the corresponding Staging Area table, awaiting further processing. As a quick side note: in some cases, you may want to integrate directly into the Data Vault but as the involved concepts don't apply for the management of change records I'll leave this out of the scenario for now.

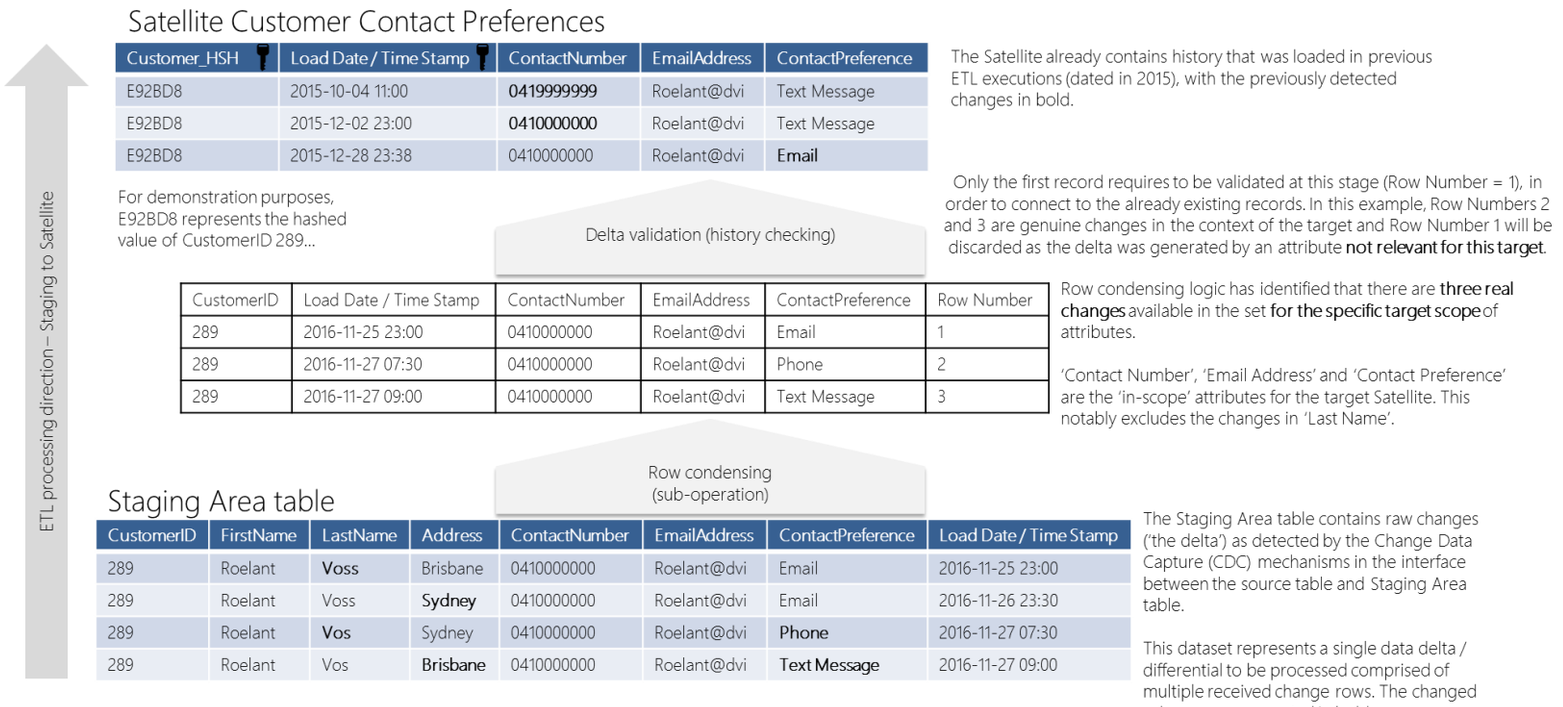
In a more-or-less traditional batch or micro-batch configuration the Staging Area contains the delta as detected by the CDC mechanism. The next step in the architecture sees the attributes modelled across two separate Satellites: Customer Preferences and Customer Contact Information. Note that the 'Address' attribute is not modelled, but since this information is stored in a Persistent Staging Area (PSA – not depicted here) we don't necessarily need to do this immediately. Satellites require a Hub (it would be Hub Customer in this case), but this is not added to the diagram to reduce the number of objects – and it is not relevant for this use-case.



Now, imagine that there are a few changes buffering in the CDC interface (some approaches support this, but not all), and at a certain point you receive all changes in one go. You will now need to do two things:

- Firstly, you need to evaluate if the provided change records are still *really* (individual) changes when you take the target attributes into account. The reason is that the attribute that triggered a delta in the source may not be part of the target table. This is what I refer to as 'row condensing', and is explained in more detail here (and in the diagram below). Examples are the Address not required in the Customer Preferences Satellite.
- Secondly, you need to check that the *remaining* change record(s) are actually a change when compared to the already existing information in the target table. I refer to this as 'history linking' and is explained further in this post.

The diagram below provides an example of these two concepts at work while processing changes to one of the modelled Satellites. In this example, the source-to-staging CDC interface has detected four buffered changes for the same key (CustomerID) since the interface last ran. This means that the Staging Area received four change records in a single ETL pass, indicating source system record adjustments at different points in time.



Delta works: supporting your ETL designs, flexiblity and virtualisation

Supporting the above concept is, in my view at least, one of the Extract, Transform and Load (ETL) best practices. The reason for this is that these mechanisms not only allow you to be flexible in your scheduling / processing, they are a fundamental requirement to reuse your patterns for advanced approaches such as Data Warehouse virtualisation. Think about it this way: if you can't process multiple changes for a given key in one pass, how can you ever simulate the full process using a view? This is why I always implement this kind of functionality across the various layers of the design. You need the same mechanisms going from, say, a Data Vault to a Dimensional Model as well.