### Delta Load Implementation

A very common question is how to implement delta loads with Data Integrator. The tool itself does not have a single delta load method, it aligns itself to whatever the source system provides. So it does not force the user into one way of providing delta information, the developer can use whatever is there.

### Why delta?

The reasons to implement a delta load are manifold. Most common is reduce the time the load takes - instead of loading all the billions of rows every day again, we process just the few changed ones. Another reason could be historical data - we want to keep the old data in the Data Warehouse and add the current state so the user can see the changes over the time.

### Delta Methods

The first and most important question is “How can you safely identify the rows that got changed in the source system?”. There could be a last_changed_date, then we have to consider just these rows for deltas and can ignore all the others. There could be a log table where all changes are protocoted by the source system. Either way, we have to take whatever the source system provides us with and if the source system does not have any method to tell what got changed, we have to read all the data and compare it with the target to identify the changes.

Here is a table of common delta methods and their pros and cons:

<table>
<thead>
<tr>
<th>Method</th>
<th>Complexity</th>
<th>Joins</th>
<th>Performance Fact</th>
<th>Performance Dim</th>
<th>Overlap</th>
<th>Deletes</th>
<th>Requirement in the Source System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timestamp based delta</td>
<td>easy</td>
<td>simple</td>
<td>good</td>
<td>good</td>
<td>yes</td>
<td>no</td>
<td>Whenever a change in the source system is made, a date column is updated</td>
</tr>
<tr>
<td>Logtable based delta</td>
<td>medium</td>
<td>medium</td>
<td>okay</td>
<td>okay</td>
<td>yes</td>
<td>yes</td>
<td>Each change causes a corresponding entry in a log table</td>
</tr>
<tr>
<td>Database Transactionlog based delta</td>
<td>very</td>
<td>complex</td>
<td>okay</td>
<td>okay</td>
<td>no</td>
<td>yes</td>
<td>You could read somehow the source database transaction log as it contains all changes</td>
</tr>
<tr>
<td>Replication based delta</td>
<td>medium</td>
<td>simple</td>
<td>okay</td>
<td>okay</td>
<td>yes</td>
<td>no</td>
<td>Use the database vendor's replication mechanisms to capture changes</td>
</tr>
<tr>
<td>Messaging based delta</td>
<td>very</td>
<td>complex</td>
<td>bad</td>
<td>bad</td>
<td>no</td>
<td>yes</td>
<td>Whenever a change is the source is made it sends a message in realtime</td>
</tr>
<tr>
<td>Table Comparison based delta</td>
<td>simple</td>
<td>simple</td>
<td>bad</td>
<td>bad</td>
<td>yes</td>
<td>yes</td>
<td>There is no change indicator, we have to read all and compare with the target</td>
</tr>
<tr>
<td>Initial load as delta</td>
<td>simple</td>
<td>simple</td>
<td>bad</td>
<td>good</td>
<td>yes</td>
<td>no</td>
<td>Simply truncate the target table and reload all</td>
</tr>
<tr>
<td>Ignore data not possible to be modified for delta</td>
<td>simple</td>
<td>simple</td>
<td>okay</td>
<td>n.a.</td>
<td>yes</td>
<td>no</td>
<td>Often there is a mixed situation, most data cannot be changed and only the recent ones can</td>
</tr>
</tbody>
</table>

- **Complexity**: How complex is it to build a delta for a given initial load dataflow without tool support?
- **Joins**: If the initial load contains a join of two or more tables, how complex is it to convert that into a delta load?
- **Performance Fact**: Here we assume a very large table with just a few changes, what performance can you expect in comparison to the other methods?
- **Performance Dim**: Either for a small table or a table with more than 25% of the rows being altered every day, how good is the method performance wise?
- **Overlap**: Does the delta give you just the rows that got changed for processing or is there an overlap? For example, if the data is going to be aggregated and merged with the current target, all has to be done transaction save. You need to know at every single moment if the +100USD have been added to the aggregated table or not. Otherwise you are in danger to add the yesterdays +100USD again. You have no way to know just by looking at the target table.
- **Deletes**: Not all methods support deletes, e.g. in the timestamp based delta you would have a change date in the row.....that got deleted? That won't work.

### There is no applicable delta method

Way too often you talk to the source system owner about how to safely identify deltas and you figure out there is none. And all the other methods would increase the time to load the target too much. This scenario is so common as the source system is typically not designed to be read. They are built to file a new order and let that be manipulated over the time until it got delivered. But the question “Gimme all orders that got changed” is uncommon within the application. Only recently ERP systems started to acknowledge that there is such a demand by external systems and they started to implement ways of getting that information. In SAP ERP there are so called “Extractors” built by the ERP team and meant to be used for that purpose - but for SAP BW as a consumer only.

One way to solving that is to bypass the source application altogether and setup its underlying database to provide that information for us. We could ask to create a database trigger on the most important tables that fills the additional last_changed_date column automatically. People will not be that happy as the database structure is usually unchange-able. Okay, then the database trigger can load a foreign table with this information. Problem with the trigger is it costs performance in the source system, not much but enough to concern people.

A second options is to setup table replication, to configure the source database to replicate all changes via the database mechanisms to a different database. That would require a large setup and under the cover of the database triggers are used again or at least a mechanism that is similar with the same effect as before - the source system performance degrades a bit.

What is considered a viable solution is to scan through the database transaction logs to figure out the changes. So no change in the database configuration, no performance downside in the daily operation. However, at the point in time the transaction log is actually scanned, this is an expensive operation. The transaction logs contain all the changes of the day and there is no way you can find the ones for your table easily. Really each and every transaction log has to be read, decoded and only then the process knows what kind of change in what table it is. If the
delta is done just once a day it might still be a good option.

And finally, databases itself support CDC mechanisms for just that. Some API calls to setup the CDC and then the delta is made available to us by the database. What sounds promising in first place is actually nothing else than what we said above: Either a trigger, a log table, replication, transaction log reader,... and their intrinsic downsides. All the database CDC mechanisms really are is a simple way to setup the underlaying base process.

When talking to the people responsible for the source database, keep in mind they will not want to make any changes to their existing system for good reasons, very good reasons in fact. On the other hand, their task per definition is not to have to source up and running for no reasons, it has to support the daily processes, and that is not only order entry but also decision making - they have to support your needs.

**Data Integrator and Delta**

Our goal with Data Integrator is not to force the user into one method, but to provide simple methods to support any delta load method you can possibly envision. Timestamp based and log table based deltas are simple dataflows with a where clause, the date being parameterized. Oracle CDC and SQL Server 2005 CDC can not only be consumed, the tool actually helps to setup the database also.

You can use the Attunity Connector to benefit from this company’s technology like SQL Server 2005 transaction log reading, CDC on AS400 etc.

- Timestamp based delta
- Logtable based delta
- Database Transactionlog based delta
- Replication based delta
- Messaging based delta
- Table Comparison based delta
- Initial load as delta
- Ignore data not possible to be modified for delta