To Delete Or Not?

No One (Really) Talks Delete In A Warehouse - Not surprising though, this is a tricky question. The contention has to be that, there is no definitive answer; there can be no definitive answer. Every situation and every environment has to decide for it what the right answer to the questions below (and similar ones) should be:

- What do you do with the data warehouse when source data is deleted?
- What do you do with the datamart when source data is deleted?
- What happens to historical data?
- What happens to all the reports etc that were produced prior to the deleted data – will there be a need to reflect these changes as they could be legitimate.
- Can historical data be deleted in the source? If yes, then what?
- Can the date you rely on from source to do versioning ever be overwritten in the source to pre-date it? If yes, then what?

■ For the purposes of this post, I will use the word **warehouse** to cover both warehouse and the datamart, either or both or any other consuming downstream system

Search all you want on the web, it is slim pickings. There is an abundance of articles, blogs, tool capability rants dealing with **how to identify** data that is deleted in the source. The "mark as deleted flag" is an all too familiar suggestion/advice. But does that really solve the problems deleted (and updated-in-place implying key columns updated) data in source systems can cause to downstream systems? What needs to happen once we have identified deleted rows is not as simple as it might seem.

The available opinions/dialogs barely scratch the surface (atleast in my opinion):

- Is the business need to report (and I will use this word holistically in the post to mean tracking, reporting and analytics) on changes that happen or only on the current data?
    - What happens then to reports produced and published if data is deleted?
- Is the need (of what to report) current and what the implications are if this changes – we are all too familiar with fluctuating needs?

As it probably is getting clear where I am going with this, lot of questions ofcourse, but what are the potential scenarios and answers. I will try to address some of these here and provide a viable option. In this post, I will deal with data that gets deleted in source systems and what the impact/resolution for downstream is. I will follow up with another post dealing with when random, uncontrolled updating of (essential/core) source data occurs. Finally I will publish the entire thing on my website (http://www.picarossolutions.com/papers.html) later.

Let me begin by providing a simple example: Let us start with a **table T1** with some rows. Now the dates (just accept this nomenclature, the philosophical discussions that exist around how to name certain attributes is perhaps legendary in the data realm) are how we track history/changes. The table below depicts information that we source and have already

consumed into the warehouse. Please note that most OLTP systems that one deals with typically have not cared about downstream needs and end up with just the date that any record was perhaps modified.

T1 Source Data			
ID	**COL**	**Effective Date**	**Comment**
1000	C1	1/1/2011	Starting point
1000	C2	7/1/2011	Basically means the value changes as of this new date and this record got updated on (say) 6/30/2011 to keep things simple
1000	C3	1/1/2012	-- You get the drift --

This is what we captured in the warehouse:

T1	Warehouse Data					**T1** Original Data
#	**ID**	**COL**	**AS_OF_DT**	**DISCONTINUED_DT**	**DEL_FROM_SRC_IND**	
1	1000	C1	1/1/2011	6/30/2011	N	
2	1000	C2	7/1/2011	12/31/2011	N	
3	1000	C3	1/1/2012	NULL	N	

- **#** is the non-intelligent generated key
- **ID** is the intelligent natural key from the source
- **COL** is some random column of data of interest
- **How To Read the example:** Source has information tied to an ID 1000 and the value of COL has changed over time from C1→C2→C3. Initially all the data is intact and nothing has been deleted. Now, how we identified (or can) the record that got deleted in the source – well, there are a myriad ways it can be dealt with
- **NULL:** in the discontinued date implies it is the current record
- The use of AS_OF_DT & DISCONTINUED_DT is my penchant

We are potentially dealing with 3 possible scenarios:

Scenario 1:

~~1000~~	~~C1~~	~~1/1/2011~~	Say on **5/1/2012** this record gets deleted in the source

- A row gets deleted in the source, (say) the business wants the ID=1000 with COL=C1 to have never existed and every record that uses this ID=1000 should point to C2 till 12/31/2011 from the beginning of time, or more precisely for this purpose 1/1/2011
  - If you are thinking "simple", change the date 7/1/2011 to 1/1/2011 for the second row in the warehouse and mark the first deleted – not so fast

- It is the warehouse, we should not be producing or repairing data but capture reality to ensure complete auditability and accountability of what is in the warehouse
    - Now if you are thinking I am on bhang, I have actually encountered these situations, and no I do not do bhang
- So what do we do

Resolution:

- As there is no previous row the second row needs to take over
- But we can lose sight of the fact that a row we have is no longer valid
- Net – there are two rows in the warehouse that are affected – the first and the second ones in this case
- Therefore, the desired end state is:

#	ID	COL	AS_OF_DT	DISCONTINUED_DT	DEL_FROM_SRC_IND	Comments	
1	1000	C1	1/1/2011	6/30/2011	Y	These records are updated as deleted in source	Changes to T1
2	1000	C2	7/1/2011	12/31/2011	Y		
3	1000	C3	1/1/2012	NULL	N		
4	1000	C2	1/1/2011	12/31/2011	N	This is a new insert	

- We have the right state of data, at any given point in time
- We have fully CYAed ourselves
- We can produce any manner of data from a reporting perspective
    a. We can pretend the original 1000→C1 never existed if we need to
    b. We can also substantiate that the value C1 which was once in use, is no longer in use

Scenario 2:

- Some row in the middle got deleted (assuming there are more than the 3 records I have presented above)

Scenario 3:

- The last row is deleted - in the warehouse this will have a NULL discontinued date

Scenario 2 &3: Resolution to these two scenarios essentially has to deal with making the previous (relative to the one deleted) row roll over to the next row's discontinued date – rather than give a laborious explanation here, I will show the final state.

Scenario 2 Resolution

T1

	#	ID	COL	AS_OF_DT	DISCONTINUED_DT	DEL_FROM_SRC_IND	
When #2 (some row in the middle) is deleted	1	1000	C1	1/1/2011	6/30/2011	Y	These records are updated as deleted in source
	2	1000	C2	7/1/2011	12/31/2011	Y	
	3	1000	C3	1/1/2011	NULL	N	
	4	1000	C1	1/1/2011	12/31/2011	N	This is a new insert

Scenario 3 Resolution

T1

	#	ID	COL	AS_OF_DT	DISCONTINUED_DT	DEL_FROM_SRC_IND	
When #3 is deleted -- This is probably the most typical situation	4	1000	C1	1/1/2011	6/30/2011	N	These records are updated as deleted in source
	2	1000	C2	7/1/2011	12/31/2011	Y	
	3	1000	C3	1/1/2012	NULL	Y	
	4	1000	C2	7/1/2011	NULL	N	This is a new insert

Now, if you are wondering if I have had too much caffeine (which incidentally is my drug of choice) to concoct something as elaborate as this (right – how does it matter, or the usual developer rant of it is too complex), the answer lies in the fact that just marking something deleted does not always guarantee the right answer because the mechanism in which data is collected from the source and organized in a warehouse (especially if history is relevant and the strategy is not full refresh) does not make this a simple case.

Part 2 →

When I started my discussion on LinkedIn about this, I started off by saying that this is a very poorly understood and/or discussed item. By no means did I imply that there is no one that understands this or dealt with it. But just that, there is not enough information in the public domain that deals with it head-on. Clearly, I have a handle on it and have attempted addressing it in the best way I knew. But I felt there should be more. As Murray Quarmby put it so elegantly "*if we add one more person of equivalent experience, there is 100 years of IT knowledge*" and he was referring to the

people involved in that discussion alone. Think of the wealth out there, other practitioners who have had to deal with this and I simply want to attempt to bring out more of that knowledge/expertise out into the public domain.

The second part of this deals with the issue that has existed for a very long time and has not gone away – that of primary identifying pieces of data being repurposed in the source systems accidentally or just because. For instance reuse of telephone numbers as has been mentioned in the discussion, and coincidentally this happened to me this afternoon when I was trying to call a friend that I was supposed to meet for lunch. But in my opinion, this is a simpler case in my opinion as telcos probably wait for a good time period before reusing numbers. And the key here is to **identify** the reuse and the warehouse can treat it as completely new based on the telephone number, date and something else -- but won't work if they need to exist consecutively without some elaborate business rules in place.

In the defense of business though, sometimes, this happens inadvertently with the business not realizing the impact on downstream systems and maybe too late for an overhaul of the business process to prevent this and in some cases this may never be overcome – for instance the domain of telephone numbers is finite.

So, let me re-present the example I had in the first post:

Original Data:

T1 Source Data			
ID	**COL**	**Effective Date**	**Comment**
1000	C1	1/1/2011	Starting point
1000	C2	7/1/2011	Basically means the value changes as of this new date and this record got updated on (say) 6/30/2011 to keep things simple
1000	C3	1/1/2012	-- You get the drift --

There are three possible scenarios here for this "random, uncontrolled updating of (essential/core) source data" as I referred to it above. Inadvertently, I might add.

Case 1: In-place update flavor 1

T1 Source Data			
ID	**COL**	**Effective Date**	**Comment**
1000	C1	1/1/2011	Starting point
1000	C2	7/1/2011	Basically means the value changes as of this new date and this record got updated on (say) 6/30/2011 to keep things simple
1000	~~C3~~ → C3.1	1/1/2012	C3 was changed C3.1 and the business do not want to ever see C3 as the result of the ID 1000 from 1/1/2012 → this is what I am referring to as an in-place update

Case 2: Inserting key date based data

T1 Source Data			
ID	**COL**	**Effective Date**	**Comment**
1000	C1	1/1/2011	Starting point
1000	C2	7/1/2011	Basically means the value changes as of this new date and this record got updated on (say) 6/30/2011 to keep things simple
1000	C4	9/1/2011	Effective date which is key identifying data is inserted (1) For whatever reason, between 7/1/2011 and 8/31/2011, the value C4 should have been used (2)
1000	C3	1/1/2012	

(1) Accidentally, I put the Effective Date column after the COL column as Effective Date is key identifying data from a historical perspective
(2) It may not happen frequently, but I have encountered these, whether or not there was a solution defined and executed, to me it appeared I should have a solution pattern that can be applied that will address all these cases in a uniform consistent manner

Case 3: In-place update flavor 2

T1 Source Data			
ID	**COL**	**Effective Date**	**Comment**
1000	C1	1/1/2011	Starting point
1000	C2	7/1/2011	Basically means the value changes as of this new date and this record got updated on (say) 6/30/2011 to keep things simple
1000	C3	~~1/1/2012~~ → 9/1/2011	

It is my belief that a data warehouse is not a data producer, so it should consume all data and not discard anything. How we report off of it will depend on what the business wants. However, from auditability, traceability perspective the warehouse should reflect all of the data changes that it got fed.

Here is what I think the result should be, and I will follow it up with a commentary:

Case 1 Resolution: T1 in-place update

#	ID	COL	AS_OF_DT	DISCONTINUED_DT	DEL_FROM_SRC_IND	
1	1000	C1	1/1/2011	6/31/2011	N	
2	1000	C2	7/1/2011	12/31/2011	N	
3.1	1000	C3	1/1/2012	12/31/2012	Y	Original Row
3	1000	C3.1	1/1/2012	NULL	N	Modified Row

- Say we detected this in-place update on 1/1/2013
- What I am suggesting here is that we do not lose sight of the value C3 because the source changed it and has no trace of C3's existence at all (unless some auditing tables were created)

- But the warehouse has had it, reported it and so forth
- Maybe it is a business imperative that C3.1 be now used/reported – even from a historical perspective (not venturing a why they would want to do that)
- By re-purposing the key (#) it is possible to give the business what they want without losing sight of the once existing C3
- Since the same # now has the latest name/ description, no dependant tables need to be changed – maybe a refresh of this specific (dimension) table in the (say)datamart

But if they ever changed it again, (say) back to C3 or some other value on 4/1/2013:

#	ID	COL	AS_OF_DT	DISCONTINUED_DT	DEL_FROM_SRC_IND	
1	1000	C1	1/1/2011	6/31/2011	N	
2	1000	C2	7/1/2011	12/31/2011	N	
3.1	1000	C3	1/1/2012	12/31/2012	Y	Original Row
3.2	1000	C3.1	1/1/2013	3/31/2013	Y	Modified Row
3	1000	C3	4/1/2013	NULL	N	Modified Row

Case 2 Resolution: Dated key insert

#	ID	COL	AS_OF_DT	DISCONTINUED_DT	DEL_FROM_SRC_IND	
1	1000	C1	1/1/2011	6/31/2011	N	
2	1000	C2	7/1/2011	8/31/2011	N	Modified Row
3	1000	C3	1/1/2012	NULL	N	Original Row
4	1000	C4	9/1/2011	12/31/2011	Y	Added Row

- There are problems with this situation, during the period 9/1/2011 – 12/31/2011, the # value of 3 in dependant tables is not really valid anymore
  a. We potentially have to identify all dependant tables, identify all the rows that are now affected, mark those as deleted and add new rows using this #=4 for this time period
- Most often than not, my recommendation has been to not alter history. However, certain businesses are resistant so I have had to keep an open mind

Case 3 Resolution: In-place update flavor 2

This is a mixed problem:

- While the symptom is similar to case 1, the identifying **date** having changed does not permit us to identify this as an updated row from the source effectively
- This is going to have to be interpreted as a deleted row and an inserted row
- The resolution here is therefore to handle the deleted row as in Scenario 1 above. And then to handle the insert

None of the above cases brought in the complexity of the warehouse being archived. Even with the assumption that the EDW is holding data "forever", these are unusual issues that can throw off even well planned processes if not accounted for. As the saying goes, there is more to the deletes than meets the eye.

What would be interesting to know, is what others have encountered and how they have handled it? Specifics will help.