Data Vault's XTS pattern on Snowflake

A problem that could occur when loading batch files to the enterprise data platform is the arrival of data ***out of sequence***, yes data vault's satellite tables are change capture tables. That is, only new records are loaded to the target satellite table either if:

- We have never seen that record before for that *parent* key (hub or link).
- Or the record differs from the current record we have in the satellite for that *parent* key. The keyword here is *current*.

What do we do with a batch of records that arrive that are ***older*** than the current records in the target satellite table? Do we ***rewind and replay*** the records in the correct order? What of the downstream information marts, dashboards, reports… oh my!? Manual operations to fix the data order incur human a cost to identify the error and to correct the data in place. Alternatively, automation identifies that the batch is not the right sequence and human intervention is again needed to pause processing, and then process the batch in the correct order. In both scenarios there is a delay in analytic value.

Is there another way? Yes, there is, let's describe it below.

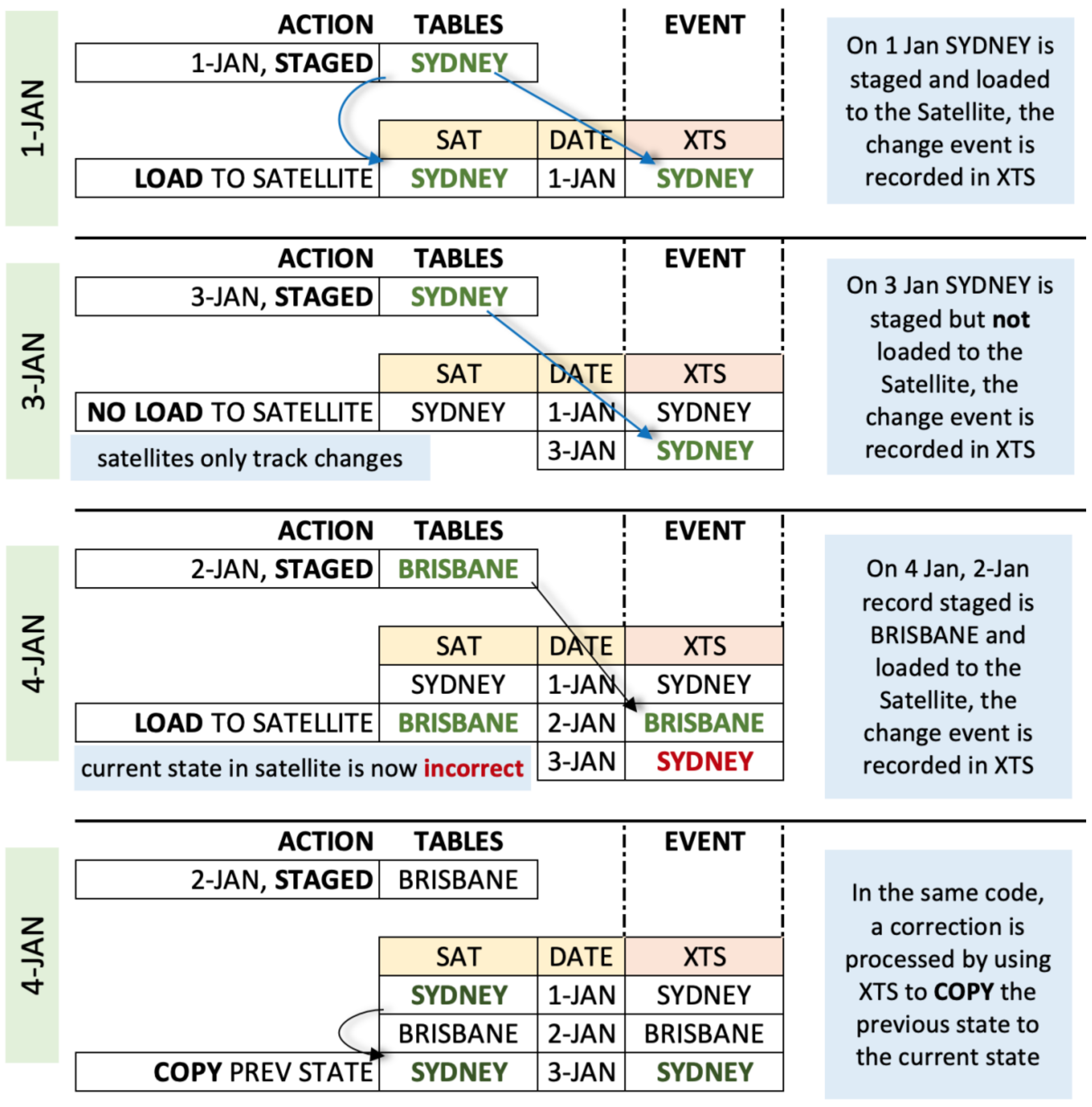

1-JAN
ACTION
TABLES
EVENT
1-JAN, STAGED
SYDNEY
SAT
DATE
XTS
LOAD TO SATELLITE
SYDNEY
1-JAN
SYDNEY
On 1 Jan SYDNEY is staged and loaded to the Satellite, the change event is recorded in XTS
3-JAN
ACTION
TABLES
EVENT
3-JAN, STAGED
SYDNEY
SAT
DATE
XTS
NO LOAD TO SATELLITE
SYDNEY
1-JAN
SYDNEY
3-JAN
SYDNEY
satellites only track changes
On 3 Jan SYDNEY is staged but not loaded to the Satellite, the change event is recorded in XTS
4-JAN
ACTION
TABLES
EVENT
2-JAN, STAGED
BRISBANE
SAT
DATE
XTS
SYDNEY
1-JAN
SYDNEY
LOAD TO SATELLITE
BRISBANE
2-JAN
BRISBANE
current state in satellite is now incorrect
3-JAN
SYDNEY
On 4 Jan, 2-Jan record staged is BRISBANE and loaded to the Satellite, the change event is recorded in XTS
4-JAN
ACTION
TABLES
EVENT
2-JAN, STAGED
BRISBANE
SAT
DATE
XTS
SYDNEY
1-JAN
SYDNEY
BRISBANE
2-JAN
BRISBANE
COPY PREV STATE
SYDNEY
3-JAN
SYDNEY
In the same code, a correction is processed by using XTS to COPY the previous state to the current state

Figure 0-1 Load Date in Green on the left, Applied Date in Staged and Satellite

In Data Vault we have a pattern to solve this, with the help of the e**X**tended Record **T**racking **S**atellites (or XTS). For a full discussion on this pattern please visit: bit.ly/3y4mUdV. *XTS stores only the HashDiff for every record, the satellite has the complete change record.*
What this article is designed to show is how the XTS pattern performs in Snowflake. With this Data Vault XTS ***architecture*** and ***model*** pattern the enterprise data platform is suddenly **self-healing**. It doesn't mean however that you should leave the platform to *excessively* heal itself! You should investigate as to why data is arriving out of sequence and look to resolve that issue!
In short this is what the XTS pattern looks like, XTS is used to monitor and manage adjacent satellite tables around a hub or a link.

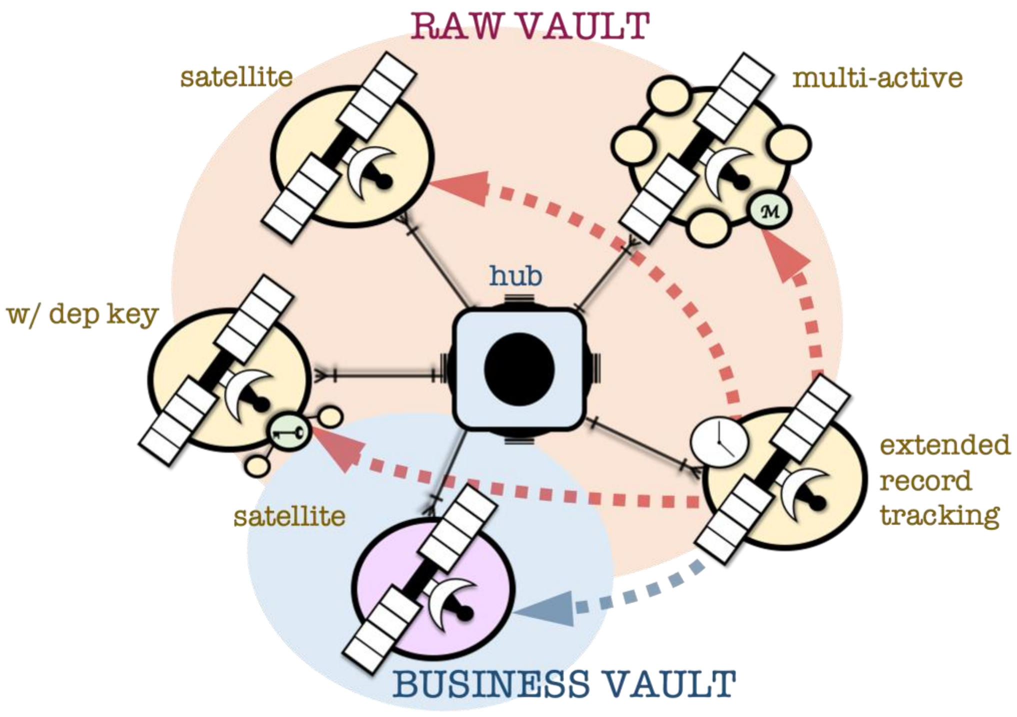


Figure 0-2 XTS can assist in correcting multiple satellite table types

As the modelled XTS is designed to facilitate timeline correction, its implementation within an automation architecture remains scalable and robust, observe:

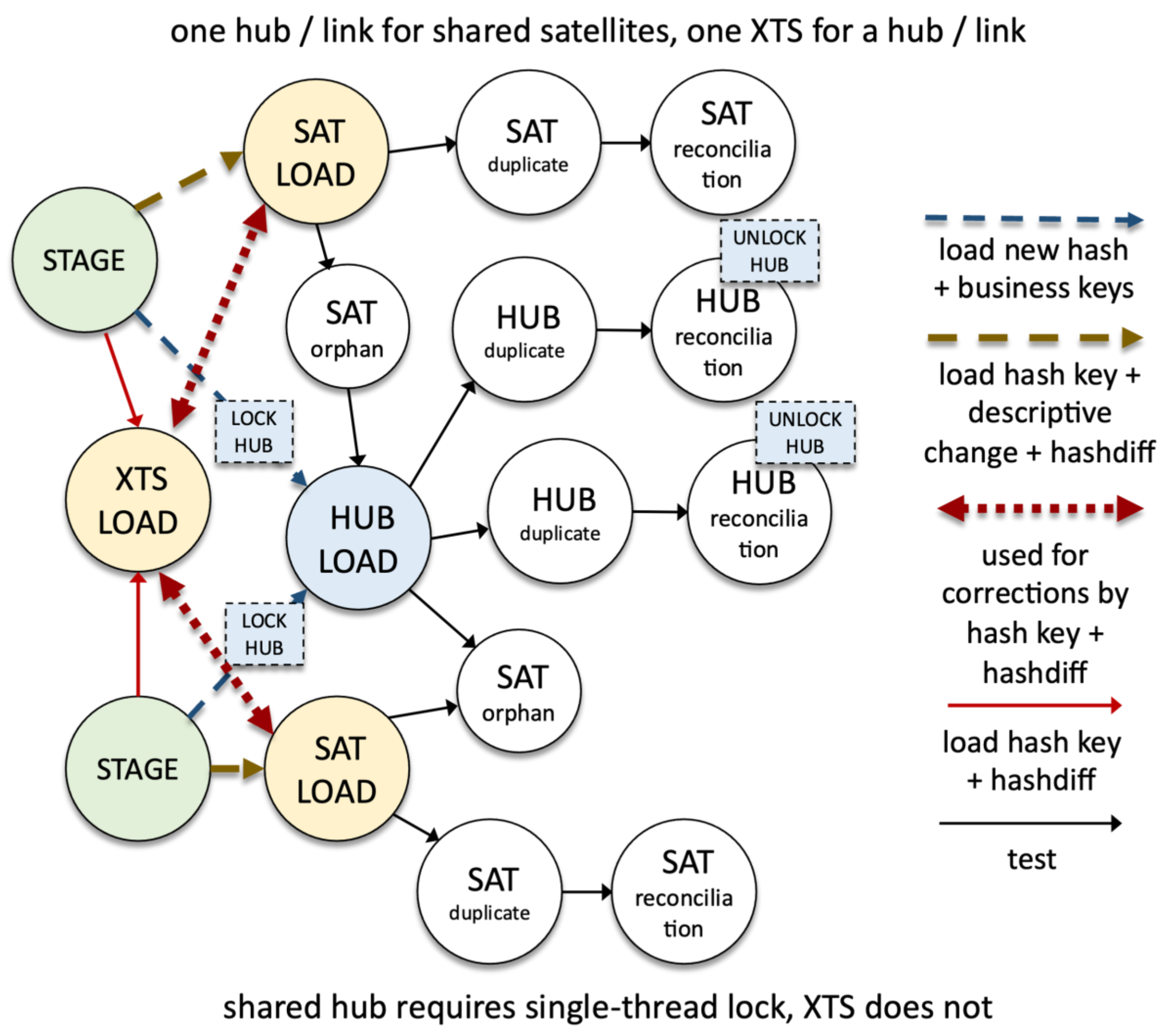


Figure 0-3 Load DAGs

This extends on the article describing data vault test automation patterns described here: bit.ly/3dUHPIS
XTS satellite can be used with raw and business vault satellites; in addition, status tracking and record tracking satellites can also be used with this pattern and an XTS satellite can be hung off a hub or a link table tracking and assisting with satellite timeline correction events. All that is recommended is that you add a new data vault tag to your satellite tables: dv_xts_event to track if the record was either 'inserted' straight from staging or 'copied', the latter of which is the correction event.

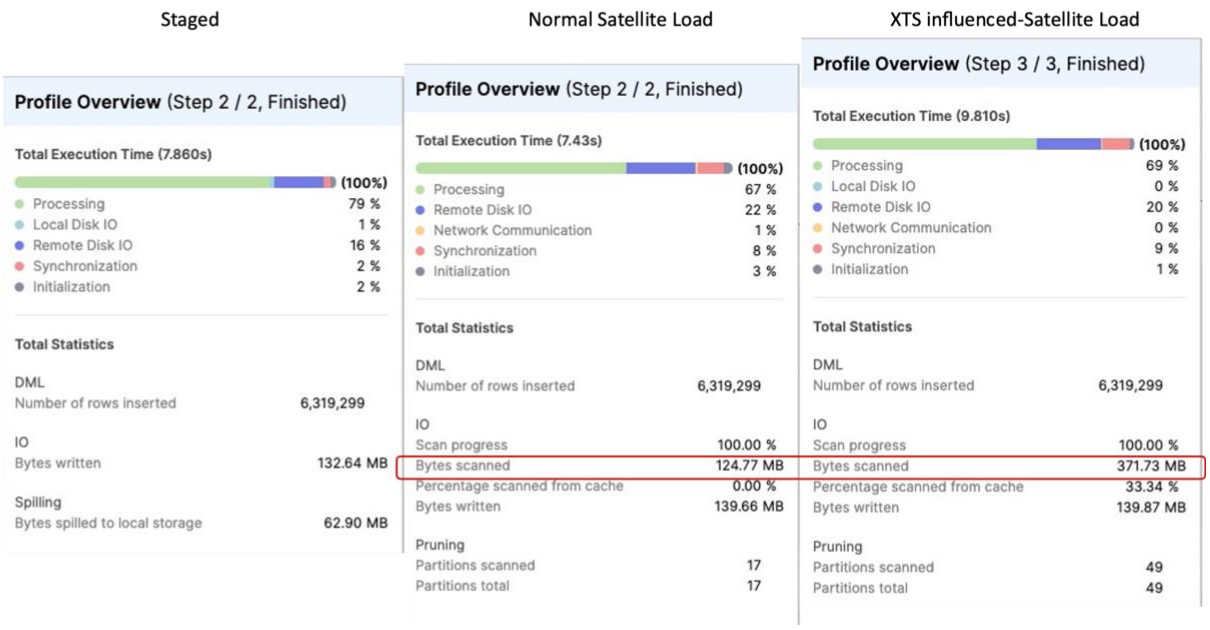
HASHKEY	LOADATE	APPLIEDDATE	HASHDIFF	XTS_EVENT	CITYNAME
hash(1)	1-JAN	1-JAN	hash(SYDNEY)	insert	SYDNEY
hash(2)	1-JAN	1-JAN	hash(SYDNEY)	insert	SYDNEY
hash(2)	3-JAN	3-JAN	hash(BRISBANE)	insert	BRISBANE
hash(2)	4-JAN	2-JAN	hash(MELBOURNE)	insert	MELBOURNE
hash(3)	1-JAN	1-JAN	hash(SYDNEY)	insert	SYDNEY
hash(3)	3-JAN	3-JAN	hash(BRISBANE)	insert	BRISBANE
hash(4)	1-JAN	1-JAN	hash(SYDNEY)	insert	SYDNEY

HASHKEY	LOADATE	APPLIEDDATE	HASHDIFF	XTS_EVENT	CITYNAME
hash(4)	4-JAN	2-JAN	hash(BRISBANE)	insert	BRISBANE
hash(4)	4-JAN	3-JAN	hash(SYDNEY)	correction	SYDNEY
hash(5)	1-JAN	1-JAN	hash(SYDNEY)	insert	SYDNEY
hash(5)	3-JAN	3-JAN	hash(BRISBANE)	insert	BRISBANE
hash(5)	4-JAN	2-JAN	hash(BRISBANE)	insert	BRISBANE

Table 0-1 Refer to "Time Crime" article for scenario descriptions

All that aside, let's get into the loading statistics!

First Load (empty satellites), staged on the left, normal satellite in the middle and an XTS-influenced satellite on the right, all stats are based on XSMALL Warehouse size.



Staged

Profile Overview (Step 2 / 2, Finished)

Total Execution Time (7.860s) (100%)

Processing	79 %
Local Disk IO	1 %
Remote Disk IO	16 %
Synchronization	2 %
Initialization	2 %

Total Statistics

DML

Number of rows inserted	6,319,299

IO

Bytes written	132.64 MB

Spilling

Bytes spilled to local storage	62.90 MB

Normal Satellite Load

Profile Overview (Step 2 / 2, Finished)

Total Execution Time (7.43s) (100%)

Processing	67 %
Remote Disk IO	22 %
Network Communication	1 %
Synchronization	8 %
Initialization	3 %

Total Statistics

DML

Number of rows inserted	6,319,299

IO

Scan progress	100.00 %
Bytes scanned	124.77 MB
Percentage scanned from cache	0.00 %
Bytes written	139.66 MB

Pruning

Partitions scanned	17
Partitions total	17

XTS influenced-Satellite Load

Profile Overview (Step 3 / 3, Finished)

Total Execution Time (9.810s) (100%)

Processing	69 %
Local Disk IO	0 %
Remote Disk IO	20 %
Network Communication	0 %
Synchronization	9 %
Initialization	1 %

Total Statistics

DML

Number of rows inserted	6,319,299

IO

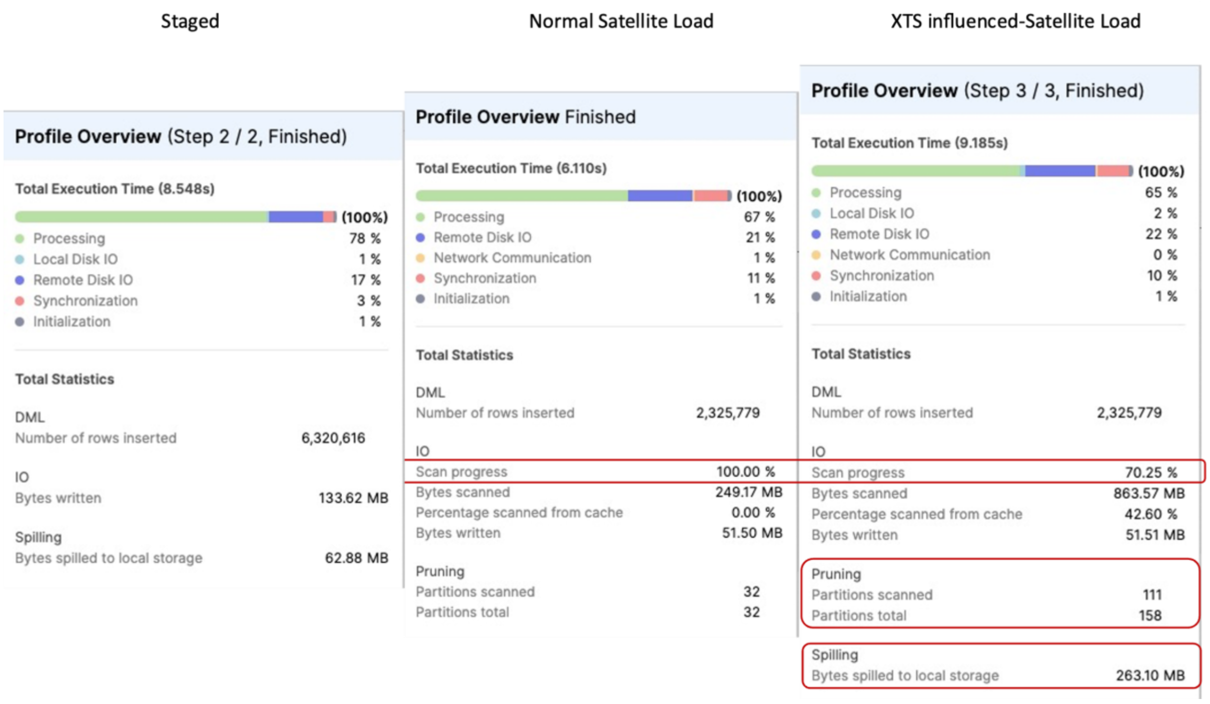
Scan progress	100.00 %
Bytes scanned	371.73 MB
Percentage scanned from cache	33.34 %
Bytes written	139.87 MB

Pruning

Partitions scanned	49
Partitions total	49

Figure 0-4 Note the bytes scanned, XTS includes both Staging and XTS to load a satellite

Now let's see what happens when we load the next day's data.


Staged

Profile Overview (Step 2 / 2, Finished)

Total Execution Time (8.548s) (100%)

● Processing	78 %
● Local Disk IO	1 %
● Remote Disk IO	17 %
● Synchronization	3 %
● Initialization	1 %

Total Statistics

DML	
Number of rows inserted	6,320,616
IO	
Bytes written	133.62 MB
Spilling	
Bytes spilled to local storage	62.88 MB

Normal Satellite Load

Profile Overview Finished

Total Execution Time (6.110s) (100%)

● Processing	67 %
● Remote Disk IO	21 %
● Network Communication	1 %
● Synchronization	11 %
● Initialization	1 %

Total Statistics

DML	
Number of rows inserted	2,325,779
IO	
Scan progress	100.00 %
Bytes scanned	249.17 MB
Percentage scanned from cache	0.00 %
Bytes written	51.50 MB
Pruning	
Partitions scanned	32
Partitions total	32

XTS influenced-Satellite Load

Profile Overview (Step 3 / 3, Finished)

Total Execution Time (9.185s) (100%)

● Processing	65 %
● Local Disk IO	2 %
● Remote Disk IO	22 %
● Network Communication	0 %
● Synchronization	10 %
● Initialization	1 %

Total Statistics

DML	
Number of rows inserted	2,325,779
IO	
Scan progress	70.25 %
Bytes scanned	863.57 MB
Percentage scanned from cache	42.60 %
Bytes written	51.51 MB
Pruning	
Partitions scanned	111
Partitions total	158
Spilling	
Bytes spilled to local storage	263.10 MB

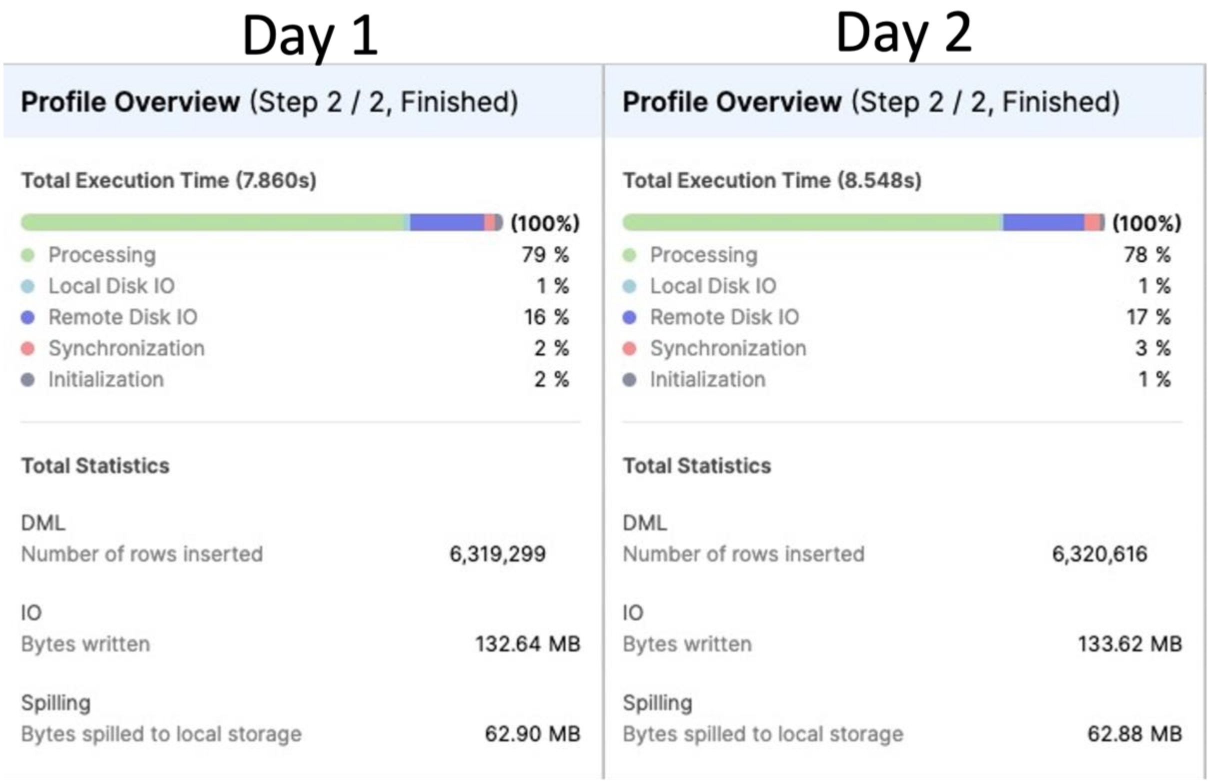
Figure 0-5 Now notice the amount of spillage under the XTS-influenced Satellite Load, including more micro-partitions scanned!

Yes, XTS involves more table scans, an option in applying this pattern would be to include a switch in your loading-pipeline, that is, if the staged applied date is older than the max applied date in the target satellite table then execute the XTS-influenced satellite loading pipeline instead. This switch-query will be free on Snowflake because the max value of every column is cached in Snowflake, essentially comparing the max applied date in staging versus the max applied date in the target satellite table.

So, you see, the Data Vault XTS pattern is a model and architecture pattern without the need to refactor anything! The optional loading pattern described above must be vetted by the data management team and above all the business. Every timeline correction event could affect reports, extracts, dashboards, and intelligence already acted upon and thus not only an alert should be created when such an event occurs, but the cost of the event calculated before publishing the correction. Yes, although this correction can happen in real time (and it is data driven), assess whether it is a pattern that suites your business & budget!

RV + BV + XTS = (self-healing) DV

Also consider, you're also loading content to an additional satellite table…



Day 1		Day 2	
Profile Overview (Step 2 / 2, Finished)		**Profile Overview** (Step 2 / 2, Finished)	
Total Execution Time (7.860s)	(100%)	Total Execution Time (8.548s)	(100%)
● Processing	79 %	● Processing	78 %
● Local Disk IO	1 %	● Local Disk IO	1 %
● Remote Disk IO	16 %	● Remote Disk IO	17 %
● Synchronization	2 %	● Synchronization	3 %
● Initialization	2 %	● Initialization	1 %
Total Statistics		**Total Statistics**	
DML		DML	
Number of rows inserted	6,319,299	Number of rows inserted	6,320,616
IO		IO	
Bytes written	132.64 MB	Bytes written	133.62 MB
Spilling		Spilling	
Bytes spilled to local storage	62.90 MB	Bytes spilled to local storage	62.88 MB

The views expressed in this article are that of my own, you should test implementation performance before committing to this implementation. The author provides no guarantees in this regard.