Paper 4629 -2020

How to Master a Risk Data Vault Using SAS Data Integration

Linus Hjorth, Infotrek

ABSTRACT

When implementing a new Risk Analysis platform, the bank's choice was to use SAS for both ETL and Credit Scoring. A part of the solution is a Risk Operational Data Store, with the centrepiece of a Data Vault. We will go through the data architecture, data modelling issues, and sum up our ETL experiences when populating and querying the ODS using SAS Data Integration Studio and using SQL Server as the ODS database.

INTRODUCTION

THE BUSINESS CASE

The bank had a legacy system for managing credit scoring calculations, both for regulatory reporting, and for credit operations, like granting new credits. The system was a sister to the replaced core banking system so in that perspective it made sense to also replace the credit scoring system.

To add to this, the EBA (European Banking Association, have introduced a new definition of default to be used by financial institutions for their regulatory reporting. This was quite different from the definition that the bank had been using and would have put a substantial amount of work into the existing credit scoring system, which was not desired.

The current solution was a black box, and it wasn't easy for the credit risk analysts to do ad-hoc analysis, compare master and challenger model performance, etc.

Default

In daily language, default means the condition of failing to meet an obligation. In the new regulation set by EBA (abe-eba.eu), the definition is strict. A customer is said to be in default if (e.g.)

- 90 days overdue over a certain limit amount
- Is in bankruptcy
- Have an application of order to pay
- Being unlikely to pay

THE CREDIT SCORING PROJECT

To mitigate the new regulation for default, phase out the legacy Cobol-based system and allow for more flexibility and transparency, the credit scoring project was started. The credit risk department was the product owner and key stakeholders, while a third-party India-based consultancy was selected to create the solution; the project manager and architects were provided by the bank.

The scope was to replace the legacy system with a 1-1 approach, except for the definition of default. It was also decided to reuse the existing SAS platform which the IFRS9/ECL (Expected Credit Loss) application has been using since a couple of years.

DATA VAULT WALKTHROUGH

Early in the project a decision was made to store the granular data in the solution which was to be in a data vault structure. One driver was that the existing data warehouse at the bank was implemented using data vault, which meant that the bank had competence and comfort in using a data vault.

THE BIG PICTURE

Data Vault is a data modelling approach, or paradigm, for data warehouses and similar analytical data stores. It has been around for some time now and have become one of the most, if not the general preferred modelling method. One of its main features is the separation of "data component types":

- keys
- relationships
- attributes

This makes the modelling somewhat predictable. The distribution of data components also makes the data model more resilient to change, especially extensions of the data base.

MODELLING COMPONENTS

As stated, the three main data component types are for keys, relationships and attributes. There are also some special constructs that targets query performance and special modelling problems. They are not covered here except for the Keyed Instance construct.

Hubs

The hub's only job is to keep track of business keys. A business key uniquely identifies a Core Business Concept. These are usually any of the following types:

- Place – geographical, organizational, electronic
- Person – physical or juridical entity
- Thing – product, stock item
- Event – monetary transaction, signed agreement, registration
- Concept – being a customer/subscriber/patient

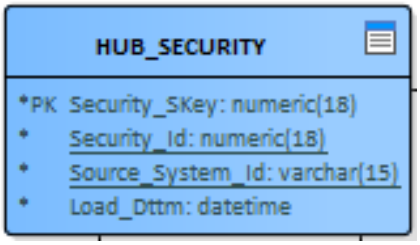

Figure 1 A hub generally consists (only) of a DW generated key, the business key, a load timestamp and the originating source identifier.

Examples of business keys:

- SSN (Person)
- Account Number (Account)
- Real Estate Identifier (Real Estate)
- Bank/Credit Card Number (Card)

Most data warehouse uses a surrogate key to represent the business key. But other approaches are becoming more common, such as hashing the business key, or even just using the business key as it is. The latter might pose some problems regarding privacy principles and legislation, such as GDPR.

The hub will load any new occurrence of a business key; however, no updates are allowed, and records are generally never removed.

Links

To describe relationships between core business concepts, link tables are used. Every link is physically modelled as a many to many (M-M) relationship between the parent tables – the hubs that are involved in the relationship. The hub keys are then defined as FK in the link table. By using M-M, even if the information is structured as 1-M, we will give the data vault flexibility if business rules change. An example of this is if today one loan cannot be linked to several properties, but then perhaps a new product is released when this possible; the data vault is flexible enough to handle this.

There are some special link applications, like the hierarchy link ("HAL"), which define relationships between keys from the same hub. Example of this are organizational and product hierarchies, product bill-of-material (one product is built up using product components), a credit card product might consist of an account, a credit, a main card and potentially additional cards.

Satellites

We can consider hubs and links as the skeleton of our data model, then satellites can be thought of as the flesh. It holds attribute information that we use in analysis and reporting, such as facts, business dates and classifications.

A pure satellite should only be connected to one hub (and in some cases a link), which means the hub PK is propagated as a FK on the satellite. Since we want to keep track changes over time for our attribute values, a date or timestamp is used together to form a composite PK.

In certain situations, you can break the rule of the PK by adding another column, usually a classification. This is known as Multi Active or Multi Value Satellite. An example can be that a customer might have number of different types of scores calculated. If it's done by using pure data vault modelling, you would have columns for all kinds of scores. This table structure would become quite sparse. Inefficient as storage pattern and not optimal for updates. Another solution would be to create one satellite for each type, but that could in some situations generate many tables, which increases maintenance, drive the model complexity and could decrease performance.

Keyed Instance

A keyed instance is a special construct that comprises a link and a hub. They come in handy when you have a relationship, but as you dig into it, you realize that the relationship itself holds valuable information. For instance, a relationship is usually not created out of thin air, there is probably an event that has occurred. So even if it didn't look like a core business concept from the beginning, it is a good practice to create a keyed instance.

A keyed instance usually doesn't have a natural business key (otherwise you probably would have modelled it as a normal hub from the beginning), rather a concatenated business key from the hubs that is combined into the link.

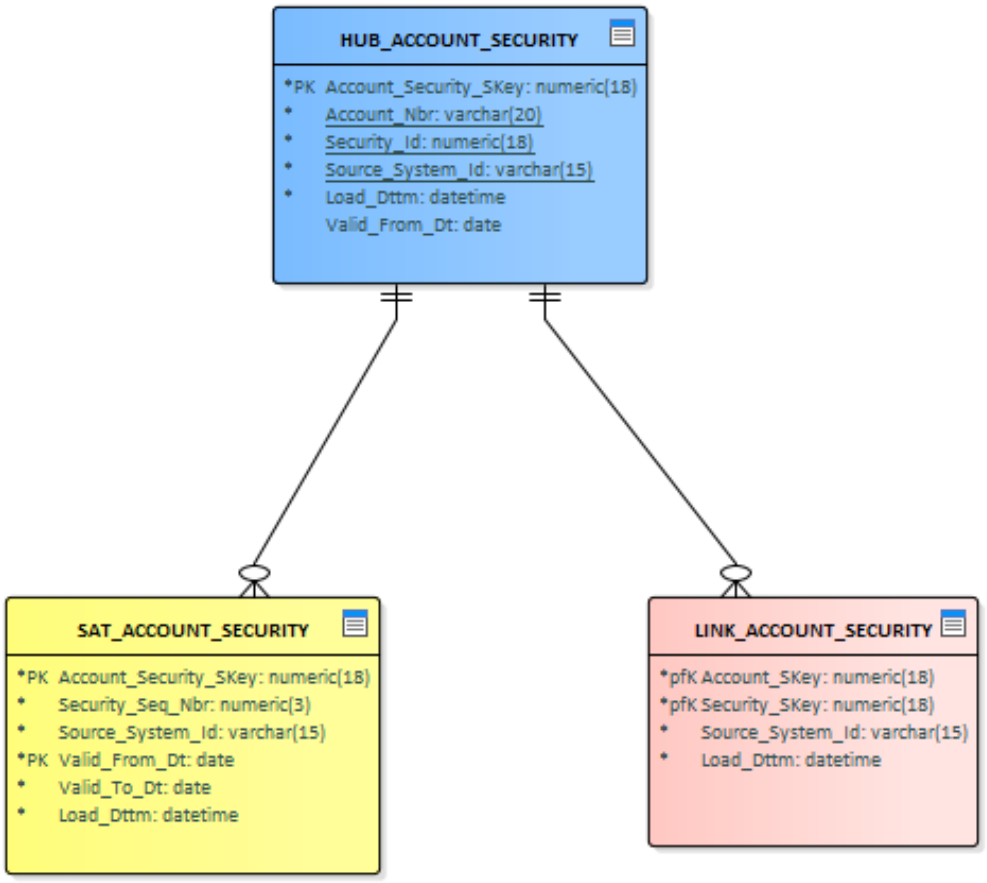

Figure 2 An example of a Keyed Instance. The link is a relationship between the hubs Account and Security (not in the picture).

DATA VAULT AND ANALYTICS ARCHITECTURE

So far, we have the basic components of hubs, links and satellites. This doesn't really tell you how to do modelling with respect to data types, naming conventions and what kind of data to populate. As guidance, there are two main types of data vault implementation patterns: the raw data vault and the business data vault. Even if they too aren't defined in a scientific way, they give some ideas and guidance how to structure your data warehouse:

Raw Data Vault

In the raw vault, data is stored pretty much like they are delivered from the source, with no data cleansing, but some alignment to data types and column naming is generally taking place. Data from different source systems can be stored in separate satellite tables. It is understood that keys are implemented as indicated in the previous section.

Business Data Vault

As you might guess from above, in the business data vault you can have cleansed data, derived data from business rules. Here you can strive for aligning data from different sources into common columns and data values domains, such common classifications – like product codes.

CREDIT SCORING SOLUTION ARCHITECTURE

SOFTWARE

As mentioned earlier, the existing SAS ECL platform is used. This includes SAS® Data Integration Studio™ and some risk modules; Model Implementation Platform™ (MIP) and Risk and Finance Workbench™ (RFW).

An extension of the platform was a separate set of servers that should cope with online service requests in real time. The major component here is the SAS Micro Analysis Service. You can think of it as a server that is more efficient than the SAS Stored Process Server. It executes only SAS DS2 programs.

A main part of the solution is a Risk Operational Data Store. The data was to be shared between the online application and the batch solution, so using a shared SQL Server instance would make that easy. Another reason to choose SQL Server was that the bank could provide a high availability on the database with an existing Nutanix setup. The Nutanix infrastructure software will among other things give you a fail-over capability for the involved databases.

BATCH AND ONLINE PROCESSING

Scoring, as well as default calculations, is performed every night on the full customer base. The base for the calculations is mainly data from core banking systems, but partially from data generated by the online services and some other miscellaneous sources. The vast part is Extract-Transform-Load (ETL) logic built in SAS Data Integration Studio, but the scoring models are implemented using SAS Modelling Implementation Platform (MIP). Execution of models is done from Data Integration Studio jobs calling Modelling Implementation Platform through a REST API.

Online processing is mainly generated by bank agents operating from offices using different channel applications. Also, end customers can, from the bank web page, trigger some online service calls, e.g. when they are applying for a loan or a credit card.

DATA LAYERS

The solution has the following main layers:

- **Landing**: a staging layer that only have the latest version of data delivered by sources or online services. Although we keep these in SAS generation data sets which can help when doing trouble shooting or want to rerun a batch.
- **Persistent Staging**: layer that holds source data but keep track of changes
- **Business Data Vault**: in this layer we store a mix of data, mapped directly from sources, business rules results and scoring output. All data that should be either published or sent downstream should pass/be stored in the data vault.
- **Business calculations and MIP input/output**: since the business rules are quite extensive there's a need to cache the data between jobs. Data here is overwritten each day.

In addition to this we have libraries for reference and control data.

ETL STRATEGY FOR THE BUSINESS DATA VAULT

GENERAL PRINCIPLES

Initially, a few guiding principles were set up on how to use the software at hand, with focus on Data Integration Studio. Some of the principles were inherited from the existing ECL/IFRS9 implementation. For use of transformations, the following principle was promoted:

1. If a standard SAS supplied transformation fits the purpose, use it.
2. If not, and the task might recur, build and use a User Written Transformation.
3. As a last resort, use the User Written transformation.

The rationale to follow this scheme is that the solution hopefully will be robust and easier to maintain. A reason not to use a standard transformation might be:

- The business rule at hand can't be applied using an available transformation
- The standard transformation generates evidently poor performing code

Another principle was that we should have only one (main) target table per job. This was to ensure an easy to maintain, a generic and a predetermined system structure. Error and exception tables from data validation/lookup were not considered as main tables.

HUBS

For hubs the most appropriate transformation is *Surrogate Key Generator*. But from earlier experience it has one major flaw. Its main job is to generate surrogate keys for new instances of business keys. But the transformation outputs either all transactions records, or all transactions and existing records from the target combined. This means that if you are to insert only new keys into the target table, another delta comparison is needed in a subsequent Table Transformation.

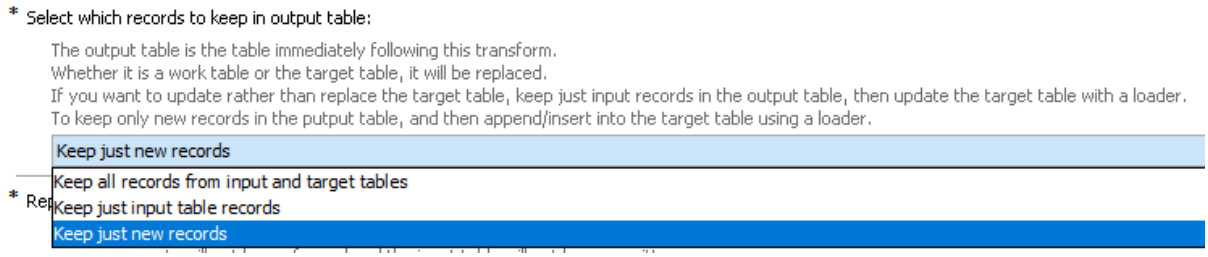

Figure 3 New option that will only output records that have been assigned a new surrogate key.

The Surrogate Key Generator transformation is supplied by SAS Institute, but it's a User Written Transformation. This means it's easy to make a copy of it, and then do modifications to as see fit. I brought in a modified version of the transformation, in which has an added option to output only records which was assigned a new key. The result is that you can chose "Insert only" as loading strategy in the Table Loader for the hub table, which of course is very efficient compared to an "Upsert" method (Update and Insert).

That said, these were the suggested steps to build a hub load job:

1. Make sure that your input data has no duplicate records on the business key
2. Use the transformation Surrogate Key Generator 2.0
3. Use Table Loader to load the Hub, using "Append to Existing" load style.

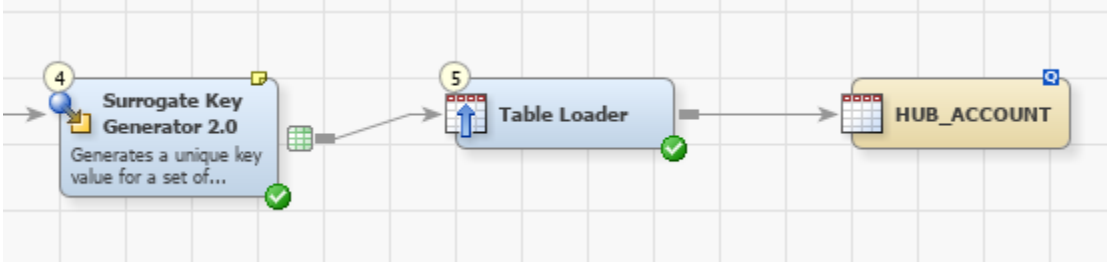

Figure 4 Standard hub job, here with the modified Surrogate Key Generator transformation.

LINKS

1. Make sure that your input data has no duplicate records on the combined business key
2. Use the Look-up transformation to get the Surrogate Keys from the corresponding Hubs
3. Use the Table Loader transformation, using the "Update/Insert" load style Techniques:
    4. Matching Rows: "Skip matching rows"
    5. New Rows: SQL or Proc Append
    6. Select your surrogate keys in the "Match by column(s)" section

SATELLITES

1. Make sure that your input data has no duplicate records on the business key
    2. Exception: if you are loading a multi value satellite, there are additional columns that will define uniqueness for a row in the table. See the data vault model for the specific satellite table.
3. Use Look-up transformation to get the Surrogate Key from the corresponding Hub
4. Use the SCD Type 2 Loader to load the satellite
    5. Use Reporting Date as "Beginning Date" Expression in the Change Tracking tab
    6. Use the hub Surrogate key as a business key
    7. Use all input columns with mappings to target, under "Detect Changes", except Reporting Date and Source System Id
    8. Mappings should preferable be all 1-1, column level transformations should have been done earlier.

KEYED INSTANCES

1. First, load the Hub – see above under Hubs. The Hub will have combined business key corresponding to the related link table.
2. Then load the link – use source data to get the surrogate keys from the Keyed Instance Hub, and the other "normal" hubs respectively. See above under Links.

THE RISK DATA VAULT MODEL AT A GLANCE

The model for this implementation is quite moderate in size. Also, most core business concepts are quite familiar to anyone using a model containing customers and accounts. What stands out is the part managing default, including the obligor concept (an obligor is one or many customers that share one or more accounts).

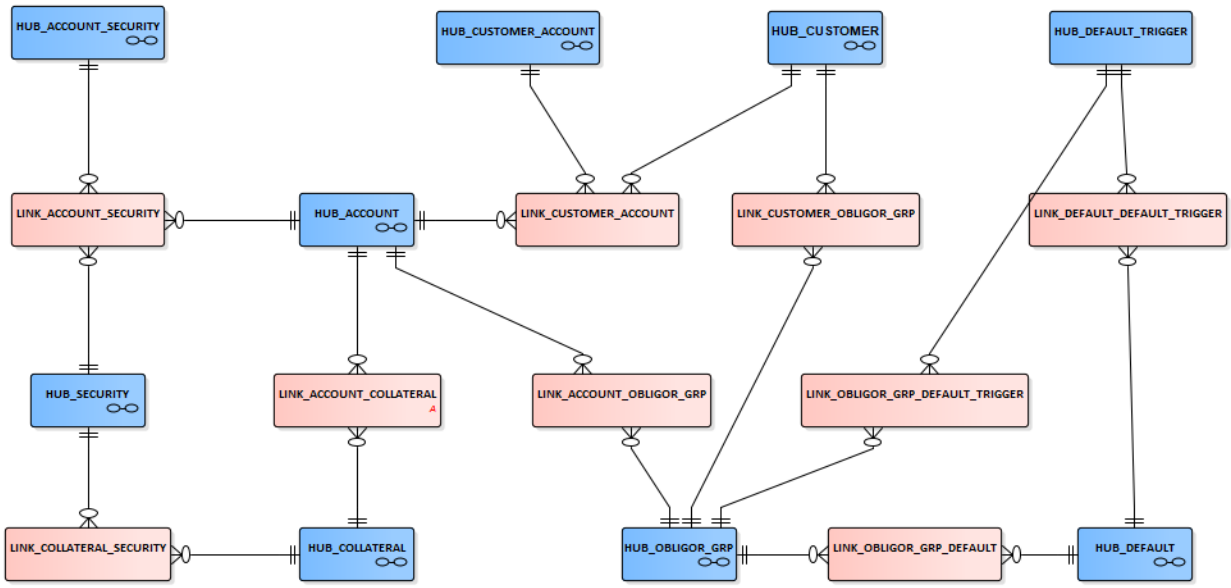

Figure 6 Backbone of the data vault model - hubs and links

If we look at the satellites in the solution, specific attributes pops up, like storing the result of scoring calculations.

A few multi value satellites exist as well, for instance for the scoring values as already mentioned – since they can be of multiple types but share the attribute structure.

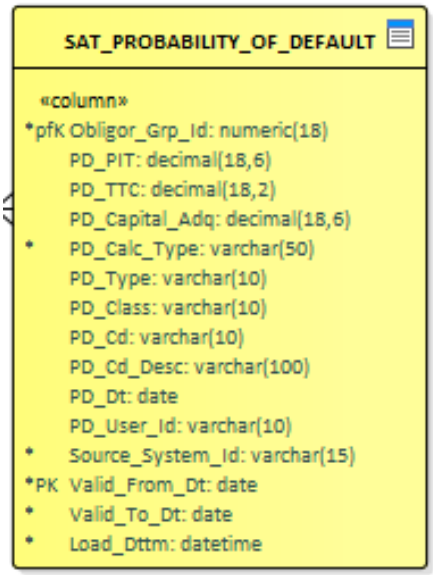

Figure 5 Example of a multi value satellite

IMPLEMENTATION EXPERIENCES

From an ETL/Data Integration Studio structural perspective, the outlined strategy was quite clear and robust. During the project and especially during acceptance testing it became clear that performance became the driver to reengineer the ETL strategy.

MODEL ADJUSTMENTS

After some of the testing was done, the team realized that we didn't cater for when data for a certain business key didn't arrive anymore. An example which lead to duplicates in downstream processing was if a co-borrower left a loan engagement, the relationship in the data model didn't cease. Links that are used to capture these didn't have validity attributes, if once existed, it will always exist.

So, we needed to create status tracking satellites to almost every link to keep track on when a relationship isn't active anymore. Since we already had some Keyed Instance constellations, we decided to go with a streamlined patter to change all links to Keyed Instances. Perhaps just a little bit too ambitious, but a uniform design.

HUBS

During the project, we kept to the original strategy. Any performance considerations that we experienced was solved by tuning the SQL Server connection, rather modifying Data Integration Studio jobs.

LINKS

The selected method to load link tables, Table Loader with Update/Insert – Skip Matching Rows, showed to be a performance killer. This because of a single step within the transformation code which find out which records are new. This is an example of the code that was generated:

```
MPRINT(ETLS_LOADER):   proc sql;
MPRINT(ETLS_LOADER):   create table etls_newrecords as select * from
WORK.TRANS as t where not exists
    (select * from bdv.LINK_OBLIGOR_GRP_DEFAULT_TRIGGER as m
       where m.Obligor_Grp_Id = t.Obligor_Grp_Id
         and m.Default_Trigger_Id = t.Default_Trigger_Id) ;
```

What happens then is the SAS/ACCESS engine transforms the query to generate a separate SQL query for each transaction record to the target table.

```
SQLSRV_31: Prepared: on connection 3
SELECT  "Obligor_Grp_Id", "Default_Trigger_Id"
FROM "CS_ODS_Business_Vault"."LINK_OBLIGOR_GRP_DEFAULT_TRIGGER"
WHERE  ( ( "Obligor_Grp_Id" = 10211 ) AND  ( "Default_Trigger_Id" = 1 ) )

SQLSRV_33: Prepared: on connection 3
SELECT  "Obligor_Grp_Id", "Default_Trigger_Id"
FROM "CS_ODS_Business_Vault"."LINK_OBLIGOR_GRP_DEFAULT_TRIGGER"
WHERE  ( ( "Obligor_Grp_Id" = 10211 ) AND  ( "Default_Trigger_Id" = 2 ) )
```

In our case each key lookup took around 0.01 seconds. Not much perhaps, but still given the quite moderate amount of data (~1 million records for many of our transaction tables) this step took around 2.5 hours! We also tried the Data Step Modify with key method, but that is working exact the same way – a lookup using an index. The conclusion was that Table Loader with upsert method only works for scenarios when you have small transaction tables.

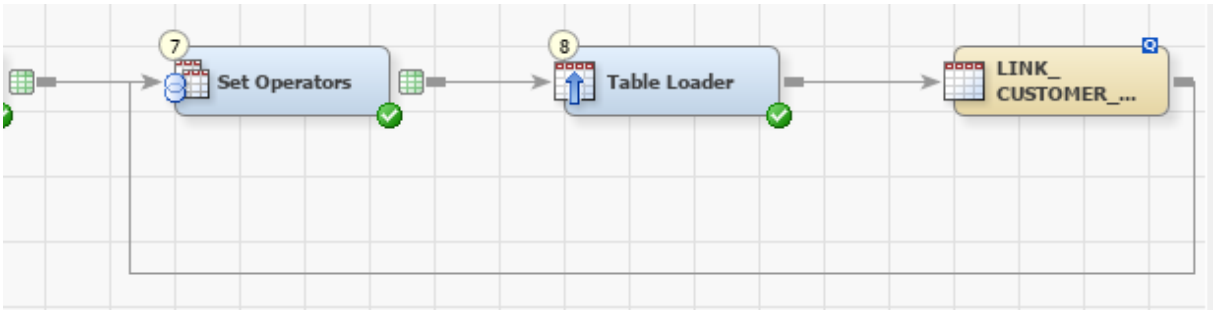

Figure 6 Link job after modification of loading strategy, using SQL SET operator EXCEPT and Table Loader.

We tried then to optimize and transform this specific query outside Data Integration Studio. And we found that the SQL set operator EXCEPT was significantly faster. The above example ran in just 2-3 minutes. With that revelation we redesigned the link jobs, to use the SQL Set Operator transformation, and of course, with the Except operator. We then fed the result into the Table Loader, now only using the Insert load method.

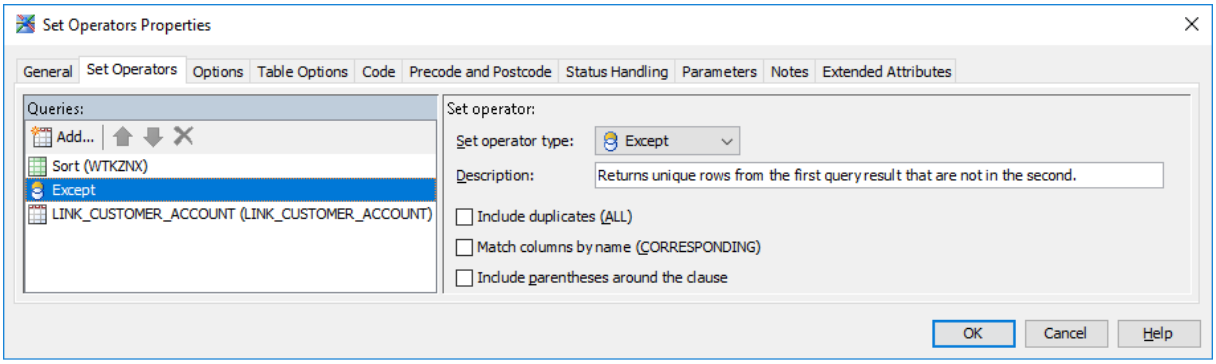

Figure 7 Using EXCEPT in SQL SET operator transformation

SATELLITES

For satellites, the pattern was quite simple – SCD Type 2 Loader which worked pretty much out of the box from a performance perspective. The challenge here was that the detail mapping must be specific to what columns to include in the detect changes list. Examples of columns that sometimes were included by mistake were Source System Id and Reporting Date) – which could result in generation of new records in every load.

KEYED INSTANCES

The Keyed Instance construct is, from a loading perspective, a Hub. Then with a subsequent lookup to load a corresponding link. So the findings for hub loading also apply here.

And here we left the principle to load only one target table. The rationale was that the business key for both the keyed instance hub, and the corresponding link was the same. Hence, we load both table in the same job and avoiding multiple lookups which would have been the case if they had been in separate jobs.

One note of caution here, if the job successfully loaded the hub, but then failed the link load – the job cannot rerun as a whole. To avoiding rolling back the hub table for the rerun, you could specify the Surrogate Key Generator transformation output as a restart point. Setting restart points is a feature of Data Integration Studio.

SQL SERVER CONNECTION CONFIGURATION

Initially when we experienced slow performance, we tried different kind of settings, like raising BUFNO and BUFSIZE. But it was one option only that really boosted performance:

CONNECTION=UNIQUE. It's interesting because the documentation doesn't really hint that this is good for performance. Also, we still can't really explain why this option worked so well for us.

CONCLUSION

SAS Data Integration Studio is a mature and well proven ETL tool. Although, every implementation is unique in some way. Most situations that caused reengineering was due to performance challenges. It's recommended that you as early in a project set an ETL strategy and test it out with production size data.

The outcome from the project was that all performance problems were solved, and by doing so, using SAS Data Integration Studio standard transformations. And now the nightly batch takes around 30 minutes, compared to 3 hours for the legacy system.

REFERENCES

Linstedt, D. 2011. "Supercharge Your Data Warehouse": CreateSpace Independent Publishing Platform

Hultgren, H. 2012. "Modeling the Agile Data Warehouse with Data Vault (Volume 1)": Brighton Hamilton

Hjorth, L. 2019. "Data Modelling for a Better Analytical Environment" Proceedings of the SAS Global Forum 2019, 3127-2019. Dallas, TX: SAS Institute

Hjorth, L. 2014 "Using SAS DI Studio to Load A Data Vault" SAS Communities Library: SAS Institute

ACKNOWLEDGMENTS

Thanks to my paper mentor Christopher Battiston (a.k.a. Darth Pathos) for cheering and valuable suggestions.

Also, my colleagues at Infotrek that let me spend the necessary time to do the necessary preparations, and especially Ylva Andersson for her engagement.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Linus Hjorth
Infotrek
Linus.hjorth@infotrek.se
Http://infotrek.se