INTRODUCTION TO THE FOCAL FRAMEWORK

How to Model, Design and Code in a Focal Data Warehouse

Summary

This document gives an introduction to how a Focal Data Warehouse works, both from a modelling perspective, but as well from an implementation perspective. All examples are simplified for ease of understanding

Top of Minds Focal

Innehåll

Background

The Focal pattern was implemented the first time as a Data Warehouse in the year 1998. That Data Warehouse is still running on the same base code today. Over the years the Focal pattern has been used in different implementations.

Today there is a company, Top of Minds Focal, that works with the continuation of the Focal Framework and teaches others how to use it.

For more information contact - Patrik Lager – patrik.lager@topofminds.se

Introduction to the Focal Framework

The Focal framework has been developed to support implementation of integration solutions. The Framework is driven through the Focal Methodology that helps the user to implement Architecture, Data Model and Code for an integration solution.

The Focal Framework aims at fostering an information modelling maturity to achieve better integration systems. The whole concept of implementing something according to Focal is based on the focal information modelling process that helps the user to ask questions that helps them understand the data. The physical implementation is fully align with the modelling process and its automation is a product of engineered design principles applied to system design and not think “coded solutions”. The difference too many other automation product/initiatives is that they build a “code generator” and think they have solved the problem of building integration systems.

The Focal Framework is not a code generator. It’s a Framework with methodology, modelling technique, architecture design principles and code patterns that are fully align and supports whole process of building an integration system.

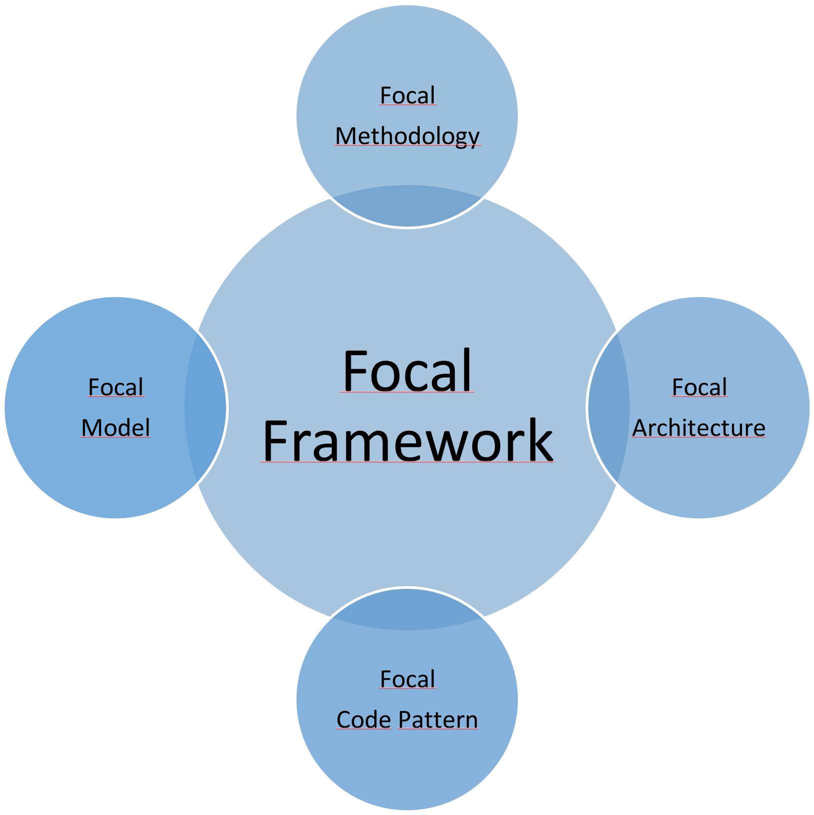


Figure 1: Focal Framework

In following chapters we will look closer on how the Modelling process focus on the understanding of data and how that process align itself to physical tables that in their part are aligned to certain code patterns and how that creates automation in the Focal Data Warehouse for ingesting and integrating data.

Focal Data Warehouse

The Focal Data Warehouse is implemented on a layered architecture. Each layer has a purpose and function. The layered approach break up the logical problems that has to be solved in a Data Warehouse implementation and aims at a modular code approach.

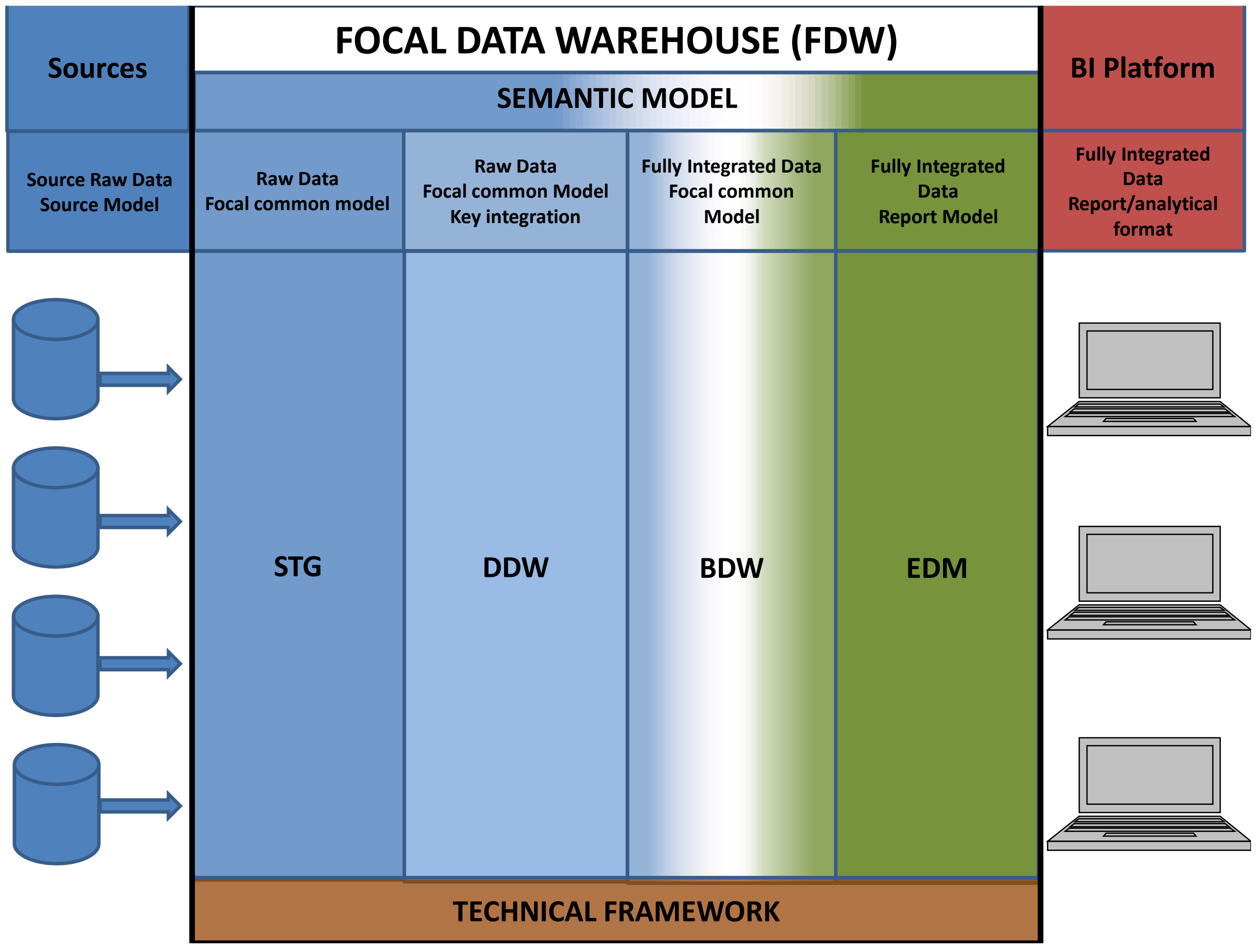

Sources
FOCAL DATA WAREHOUSE (FDW)
BI Platform
SEMANTIC MODEL
Source Raw Data Source Model
Raw Data Focal common model
Raw Data Focal common Model Key integration
Fully Integrated Data Focal common Model
Fully Integrated Data Report Model
Fully Integrated Data Report/analytical format
STG
DDW
BDW
EDM
TECHNICAL FRAMEWORK

Figure 2: Focal Data Warehouse layered architecture overview

The layers STG → BDW are focused to apply the three areas of integration, to ensure a common data repository in BDW that can be used for reporting and analytics. The integration parts are

1. Common model
2. Key integration
3. Semantic integration

The Implementation pattern

The Focal methodology supports an architectural design that is fully aligned with the Focal information modelling process. The Focal information modelling is supported by table patterns ready to accept the data and code patterns to load these table patterns.

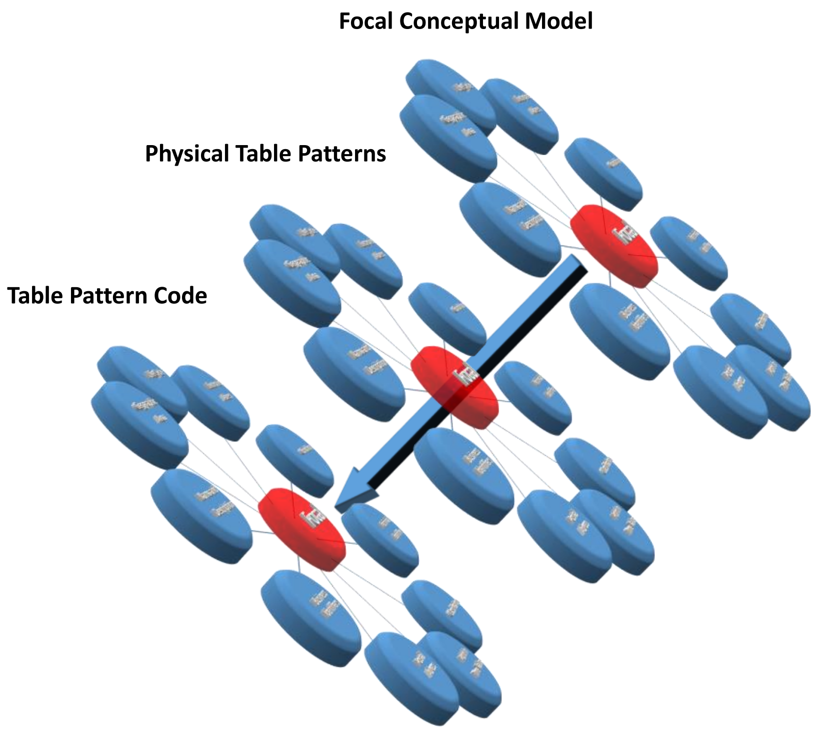


Figure 3: Focal implementation pattern

Each level in the implementation pattern are aligned to support an ease of implementation and focus on information modelling.

Focal Conceptual Model

Focal Conceptual Model is the representation of data in Core Business Concept and Descriptor Concepts. A Core Business Concept can be an Event, Thing. The Descriptor Concepts groups attributes, which describes the Core Business Concepts, together in information areas.

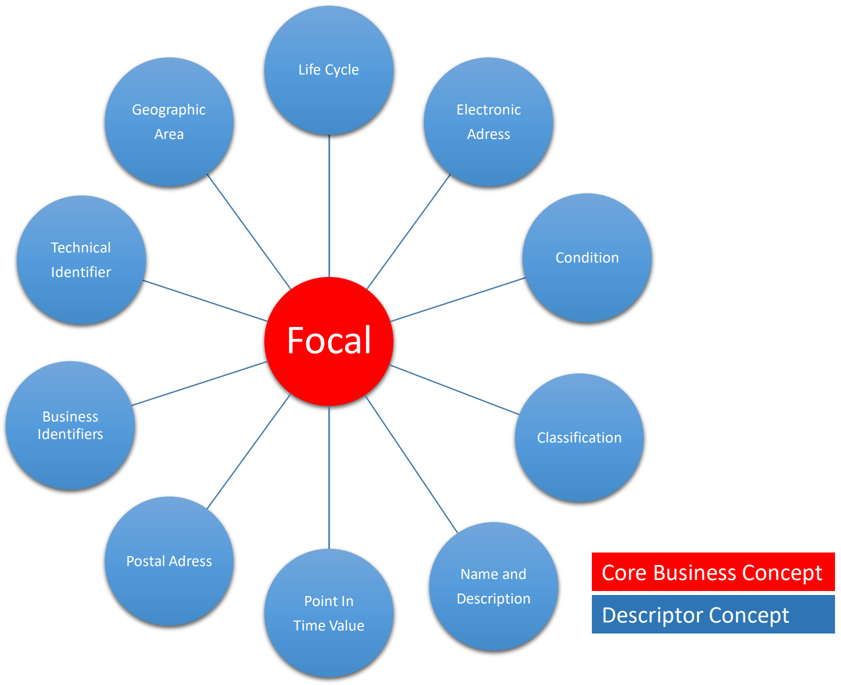


Figure 4: Focal Conceptual Model

This is done to support the Focal Modelling Process and create an aligned implementation pattern.

Focal Modelling

The Focal Modelling is based on the modelling process in the Focal Methodology, which focus on concepts and meaning of things. Focal Modelling is closely align with the Ensemble Modelling theory. The process aims at creating an information model that are aligned with the Focal Conceptual model.

The Focal modelling process is done as two main steps, First focus is on understanding the business and their information requirement and represent that in a data model. That is done through the Focal Forms, from 1 up to 5. That becomes your target model for the BDW (Business Data warehouse) layer. Next step is to use the Focal Forms with source system data and map it to the business target data model.

Since the Focal Data model is an Ensemble modelling technique. The Focal Model works with three things in the modelling session. **Core Business Concept**, **Attributes** and **Unique Natural Business Relationships**. The **Core Business Concepts** are represented by a **Focal**, the Core Business Concept is described by attributes that are grouped into **Description Concepts** which are represented as

Descriptors and the **Unique Natural Business Relationships** between Core Business Concepts are represented by **Relationship**.

Focal Forms

The Focal Forms are applied with two different purposes. First to understand the business then to understand how the source systems can support with data for that model. You do not need to run through all Focal Forms for business to start with source system analysis. Therefore the document describes each Focal Form both the business focus and the source system focus, since once you done one Focal Form for business you can start your source system analysis for that Focal Form. Below is a high level overview of each Focal Form.

Focal Form One

Business - Identify your Core Business Concepts

The first step is to understand the business concepts that the business want to capture data about. These concepts are often like Customer, Product and Contract etc. These Core Business Concepts will be your Focal(s). A Core Business Concept is the representation of Event, Thing, and Person that has a meaning to the business. Examples are Customer, Sales, Employee, and Product.

Source system – Find the best identifier for Core Business Concepts

Once we have found out the Core Business Concept(s) in the Focal Form One for Business, the source system analysis for Focal Form One can start. The focus for Focal Form One for source system is to find the best possible identifier(s) to represent and integrate instances of a Core Business Concept. Example, if the Business has a Core Business Concept of Customer, what is the best way of Identifying a specific customer(instance) to achieve uniqueness and, when needed, integration in the Customer Concept.

Focal Form Two

Business – Find Unique Natural Business Relations

The second step is to find and understand the unique natural business relations. Which means that the process focusing on to understand how our different Core Business Concepts relate to each other in different business scenarios/processes/use cases.

Source system – Understand information around relationships in source

The second step is to see how the source system(s) support these unique natural business relations.

Focal Form Three

Business – What describes the Core Business Concept

In the third step we find out under which what attributes the business wants to use to describe the Core Business Concepts. At this point the source system analysis will wait until Focal Form Five.

Focal Form Four

Business - Group attributes into Descriptor Concepts

Once the attributes are in place in relation to its Core Business Concept. The Focal Form Four focus on getting a deeper understanding of the meaning of the attribute(s). This is done by grouping them into the Focal Frameworks Descriptor Concepts. These concepts carries a definition which we matches against the definition of the attribute(s). When an attribute(s) definition matches a Descriptor Concept it is mapped to it.

Focal Form Five

Business - Define the Atomic Contexts

Atomic Context is a way of digging even further into the meaning of attribute(s) to get a clearer understanding of what the business means. The Atomic Context is also basis of the physical implementation and cannot be ignored.

An Atomic Context is one set of attribute(s), 1 to many, that describes a certain information at an atomic level. That means that it only describes the certain information and nothing else in one row of data. You can also say that an Atomic Context will answer an atomic question about a Core Business Concept.

Each attribute that the business has described has to be defined in its Atomic Contexts. First write a long text explaining what the Atomic Context means and how it is defined. When that is done we make a short version of the definition. From that short version we build a text string that represents the Atomic Context. This string is the name (id) of the Atomic Context. This is important for many aspects.

1. This will give us an deeper understanding of the attributes we work with
2. While writing the definition of the Attribute, you will find out what are the components of the Atomic Context.
3. The identifier we find that represents the Atomic Context is fundamental for the implementation of the physical data model.

Atomic Context

The idea of Atomic Context is rather unique to the Focal Form modelling, so here is a bit more information on how to think about Atomic Context. As said above the Atomic Context should answer an atomic question about a Core Business Concept. Here are a few examples.

Question 1: When were you born?

Answer 1: That was the 1969-06-01

Question 2: What is your phone number?

Answer 2: it is 0701-103345

So far the Atomic Context has been simple. One attribute has been able to answer the question on its own. Now we will look at a set of attribute to answer a question.

Question 3: What is the Balance on you deposit account?

To answer that we need to add more than one attribute.

1. Amount
2. Currency
3. Date of the Balance

Answer 3: My balance is 100 EUR at 2016-01-23

To create with Atomic Context we work with definitions to localize and understand the meaning.

This is an example of the attribute Binding Period:

- In Focal Form Three the business describes that they want to keep track of the Binding Period that a customer sign for a contract. The Binding period gets placed as a descripted attribute for a Contract.
- In the Focal Form Four the Binding Period is mapped to the Focal Descriptor Concept – Condition, since it is an attribute that represent that a counterpart has signed a contract and by that agreed to fulfill it's the condition(s) of the contract of the allotted time period.
- In Focal Form Five the Atomic Context for Binding Period is set, that is done through the work of defining what a Binding Period means to the business.

Example definition of: Binding Period

The Binding Period is legally agreed upon between the counter parts involved in a transaction. The Binding Period defines the time period where the counter parts are legally bound to fulfill their part of the agreed condition(s). The Binding Period is set in a finite time period where there is a start and end date to the period. This period is described in a time measure that represent the length of the period between start and end date.

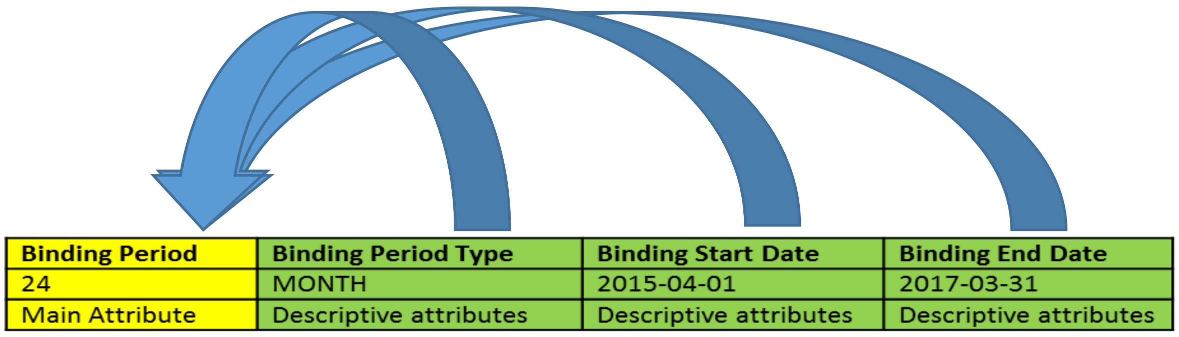
Short Description
The Binding Period is legally agreed upon between the counter parts involved in a transaction.

Name: BindingPeriodAgreed

From the definition we understand that a binding period really has multiple parts in it. The length of the binding period, the other is the representation of From and To of that length, and then in what unit of measure we represent the length in.

From this process we can also understand that the main attribute in the Atomic Context is the attribute that holds the length of the Binding Period and that start and end date are descriptions of the period.

Descriptive attributes describes the main attribute. In this case Binding Period Type describes that the unit of measure of the Binding Period, is Month. The Binding Start and End Date describes within which time period that the 24 month Binding Period is active.



Binding Period	Binding Period Type	Binding Start Date	Binding End Date
24	MONTH	2015-04-01	2017-03-31
Main Attribute	Descriptive attributes	Descriptive attributes	Descriptive attributes

Figure 5: Main and Descriptive attributes in an Atomic context

When we create a definition for an atomic context it is important to try to keep it generic and really represent atomic context as such. Once we have the definitions in place we can much more easily identify atomic contexts in source systems and also understand if they are complete. If a source system would send Binding period information but only sending *the From and To date* of the period, we would quickly recognize that we are missing parts (period length and measure type) in the atomic context.

Source System – Map source system attribute(s) to Atomic Context

At this point it is time to understand how the source system(s) can support the atomic contexts that the business has asked for to describe the Core Business Concept(s). The job is to map the source system attributes against the atomic contexts definitions.

Conclusion

The Focal Forms are there to support an information modelling process which helps us implement a well-defined and integrated Data Warehouse. While the steps might seem hard, the reason to do them is basis if you want to succeed to implement a real data warehouse. The Focal Methodology want the implementers of Data Warehouses to focus time and effort on the modeling part and then support an automatic implementation of the result of the modelling effort through reusable table patterns and reusable code patterns.

The physical table patterns

For each conceptual object in the Focal Conceptual Model there is a physical table pattern that supports it.

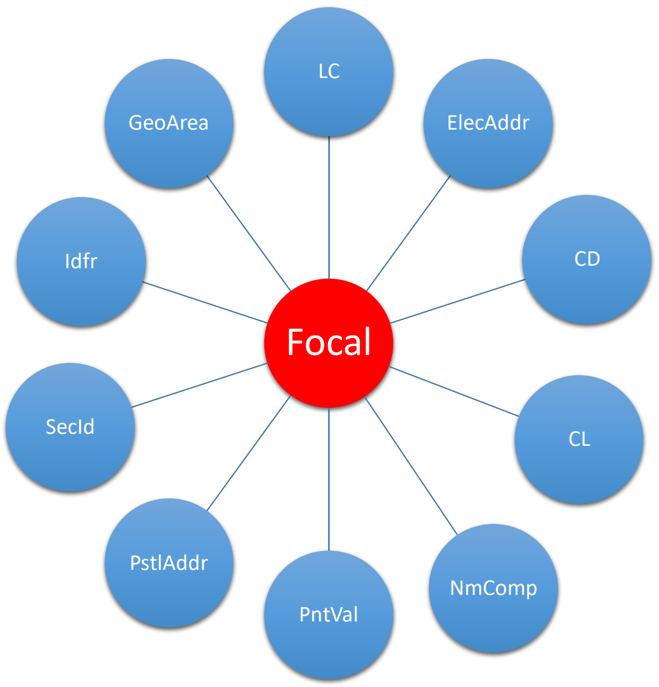


Figure 6: Physical table patterns

Since each of these table patterns represents a generic descriptor concept, they can be reused for any Core Business Concept. So independent on any number of Focals (Core Business Concepts) that are implemented, the number of physical table patterns are the same. As an example, we will reuse the Classification pattern to implement a classification table for every Focal that has Classification attributes.

Through the Focal Forms, the specification for loading the physical model is done. Not through normalization as such, but through the process of breaking down the data into atomic context to understand it.

A table pattern never gets implemented independent of Focal. That means that a table pattern gets an instantiation for every focal it belongs to. As an example, there is not one CD table for all Focals, but one instantiation of the CD table pattern for each Focal.

Table Pattern Code

When building code to process data, the format of the source for the data and the target where the data should end up defines much of how the code in the job will look like. If you, as in a Data Warehouse, has 10 to 100+ different sources with data we need to process, and 50-500 different tables to load that also looks different, then we can easily understand that the number of unique jobs can easily explode and become very hard to maintain.

Focal Data Warehouse has approached that challenge with a pattern based design. For each architectural data layer, there are table pattern code that loads a specific table pattern. Each codebase for each layer has also unique functionality that is will support certain transformation logic.

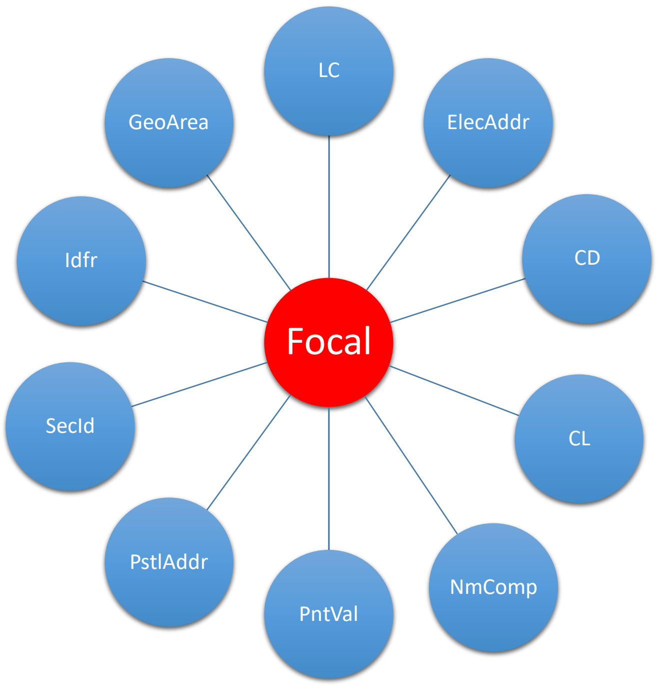


Figure 7: Generic table pattern code

So for each data layer (STG, DDW, BDW) there are jobs that loads a specific table pattern.

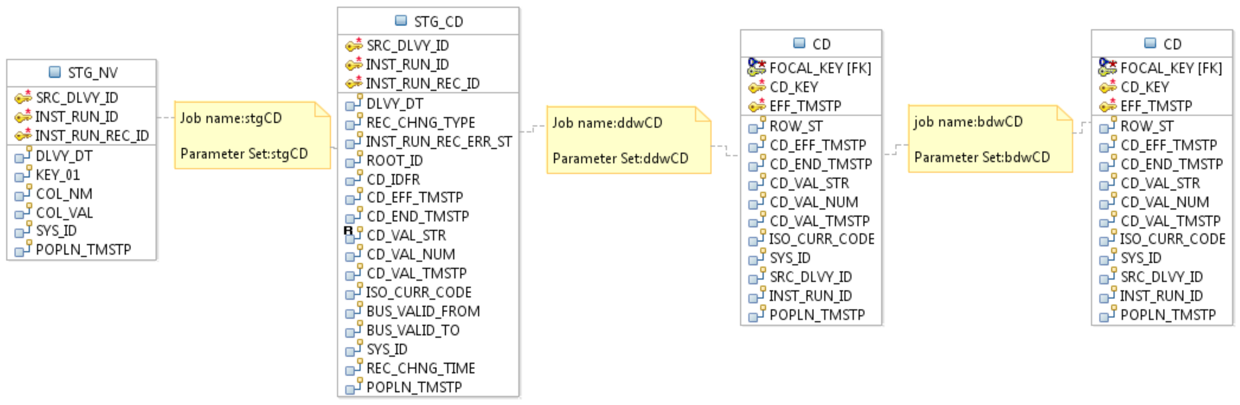


Figure 8: Code pattern aligned with Table pattern

Each job is aligned to the table pattern and then gets instantiated with parameters to load a specific instantiated table. This way the number of jobs will not grow dependent on how many Focals (Core Business Concept) that gets implemented.

Automation in Focal Data Warehouse

Hurry slowly – is the proverb that Focal Data Warehouse works by. It means that, if you do your information modelling correctly the implementation will be almost automatic.

We will in this section look at the different layers in a Focal Data Warehouse and how the code used in each layer has a strong connection to achieve the main purpose of a Data Warehouse – Integration.

The Focal Data Warehouse has four layers, where the first three layers are implemented with Focal Model. These three layers (STG, DDW, BDW) as a strong connection to integration theory. The last layer, is EDM, where we build our reporting Data Marts to support data consumption of the user.

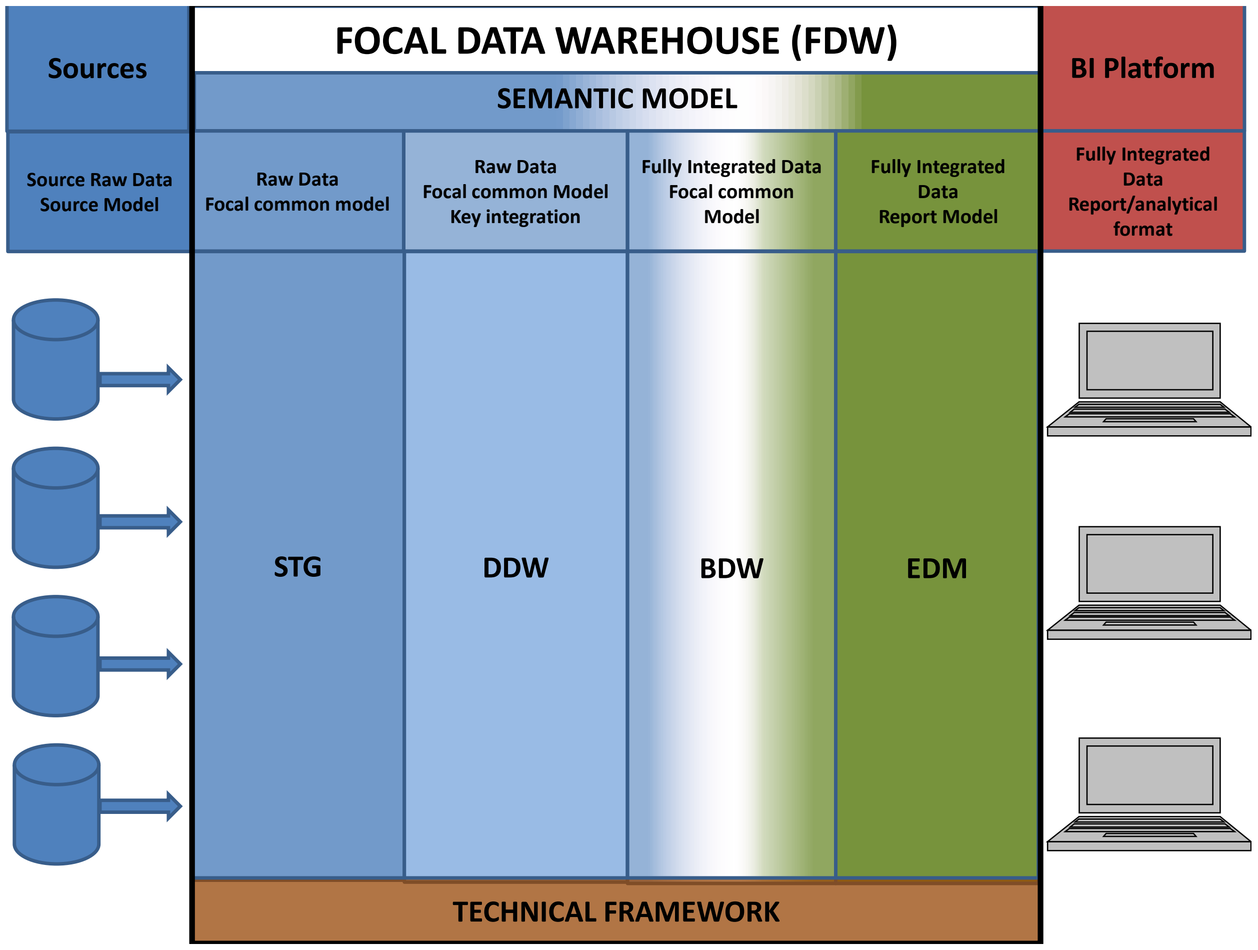
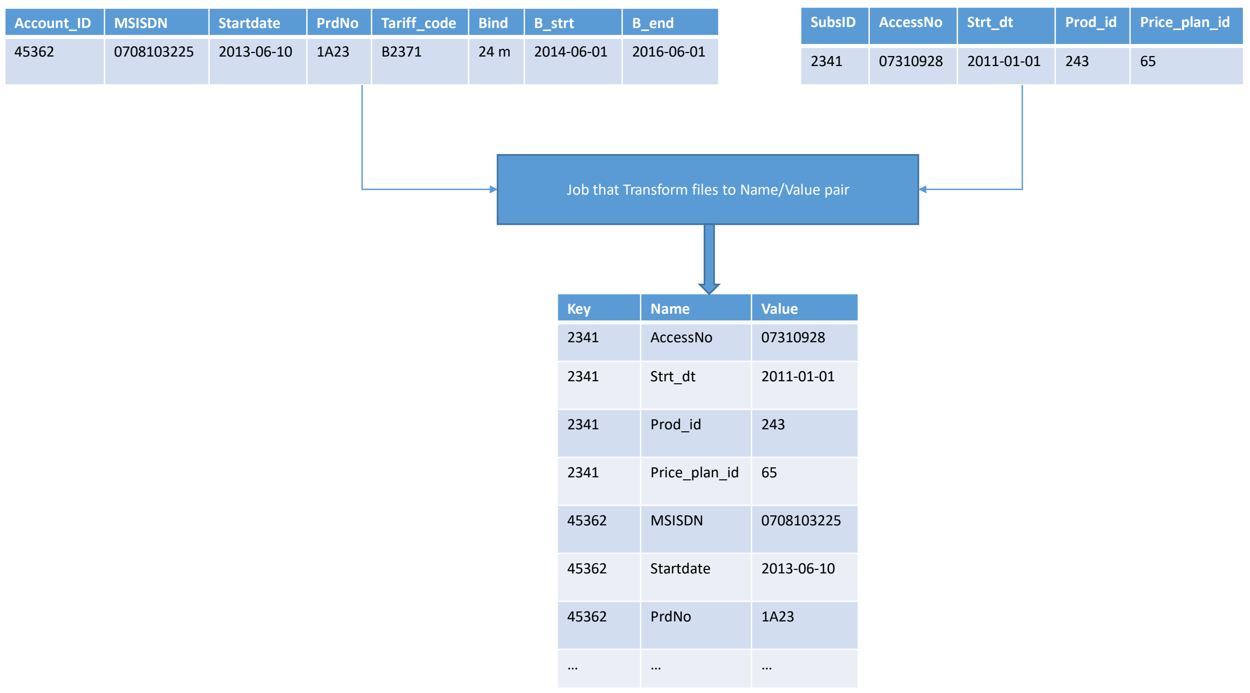

Sources
FOCAL DATA WAREHOUSE (FDW)
BI Platform
SEMANTIC MODEL
Source Raw Data Source Model
Raw Data Focal common model
Raw Data Focal common Model Key integration
Fully Integrated Data Focal common Model
Fully Integrated Data Report Model
Fully Integrated Data Report/analytical format
STG
DDW
BDW
EDM
TECHNICAL FRAMEWORK

Figure 9: Focal Architectural Layers

Neutralize the source

Often there is many different sources that feeds a Data Warehouse with data. If we do not have a generic/common semantics of the file header we will have the problem of accessing the data from the file(s) through a generic function. There are different ways of doing this. One way is to break down a file into Name/Value pairs. That means that we neutralize each source file to a common format.

We will use the Name/Value pair example to describe the idea. In the picture two different files are broken down to Name/Value pair and loaded into a pre staging table that will be the source for the code patterns to load staging.



Account_ID	MSISDN	Startdate	PrdNo	Tariff_code	Bind	B_strt	B_end
45362	0708103225	2013-06-10	1A23	B2371	24 m	2014-06-01	2016-06-01

SubsID	AccessNo	Strt_dt	Prod_id	Price_plan_id
2341	07310928	2011-01-01	243	65

Key	Name	Value
2341	AccessNo	07310928
2341	Strt_dt	2011-01-01
2341	Prod_id	243
2341	Price_plan_id	65
45362	MSISDN	0708103225
45362	Startdate	2013-06-10
45362	PrdNo	1A23
…	…	…

Figure 10: File(s) to Name value pairs

By doing this the Data Warehouse has now one format to access data from (KEY, NAME.VALUE). This means that the first obstacle of having multiple source formats to read from has been mitigated. The jobs now can be written generically to access data from sources. But what about the output part then? If the tables all look differently we will still have the issue of multiple targets to apply the result in.

Neutralize the target

The Focal model is pattern based and only has 12 different table patterns. 1 Focal, 10 Descriptors, 1 Relationship. The Focal jobs are almost one to one with the table patterns. The Technical Identifier job loads the Focal pattern as well as the Technical Identifier pattern and the Relationship job also loads the Relationship tracker pattern. So all in all there are 10 jobs in the Focal solution for each layer. Staging, DDW and BDW. That means that we can build a fully integrated Data Warehouse with 30 jobs, independently of how many source system, files and subject areas that the Data Warehouse supports.

The Staging Layer (STG)

The Staging layer is non persistent and its focus is to extract data from the source and load it into a common model. The common model here is based on the information modelling done in the modelling process. In this example we will look at how this is done with a neutralization of source by Name/Value pair.

When we had reached *Focal Form 4* in our modelling process we knew which Focal(s) we had, what attributes belonged to a certain Focal/relationship and what Descriptor Concept the attributes belonged to. When we applied the *Focal Form 5* we also identified how each atomic context looks like.

Once the source is neutralized through the use of the Name/Value pair job the Staging Focal Jobs can start moving data from the Name/Value pair table, into the designated tables staging tables that we found in our modelling session.

So let's look at our source and our target and the Atomic context that we will load. Everything is simplified for ease of understanding.

We will use our Atomic Context – Binding Period in this example

The source is a Name/Value pair table which looks like this with one subscription loaded with its Binding Period data

Key	**Name**	**Value**
123	BINDING PERIOD	24
123	BINDING PERIOD TYPE	Month
123	BINDING PERIOD START DATE	2014-06-01
123	BINDING PERIOD END DATE	2017-05-31

Figure 11: Name Value Pair table

Now we want to load our Subscription Condition Table (SUBS_CD) in our staging area, so it looks like this

SUBS_ID	**CD_NAME**	**CD_START_DATE**	**CD_END_DATE**	**CD_VAL**	**UOM**
123	BINDING PERIOD	2014-06-01	2017-05-31	24	Month

Figure 12: Subscription Condition Table (SUBS_CD)

A simplified SQL how we transform the Name Value Pairs into the Atomic Context of Binding Period

```
SELECT
KEY AS SUBS_ID
MAX(WHEN NAME='BINDING PERIOD' THEN NAME ELSE NULL END)CD_NAME
MAX(WHEN NAME='BINDING PERIOD START DATE' THEN VALUE ELSE NULL END)CD_START_DATE
MAX(WHEN NAME='BINDING PERIOD END DATE' THEN VALUE ELSE NULL END)CD_END_DATE
MAX(WHEN NAME='BINDING PERIOD_TYPE' THEN VALUE ELSE NULL END)UOM
MAX(WHEN NAME='BINDING PERIOD' THEN VALUE ELSE NULL END)CD_VAL
FROM NV_TABLE
GROUP BY
KEY
```

So we know the source and we know the target. When we set parameters into this job, it looks like this.

```
SELECT
KEY AS #P_FOCAL#||_ID
MAX(WHEN NAME='#P_CD_NM#' THEN NAME ELSE NULL END)CD_NAME
MAX(WHEN NAME='#P_CD_START_DATE#' THEN VALUE ELSE NULL END)CD_START_DATE
MAX(WHEN NAME='#P_CD_END_DATE#' THEN VALUE ELSE NULL END)CD_END_DATE
MAX(WHEN NAME='#P_UOM#' THEN VALUE ELSE NULL END)UOM
MAX(WHEN NAME='#P_CD_VAL#' THEN VALUE ELSE NULL END)CD_VAL
FROM NV_TABLE
GROUP BY
KEY
```

So if we load the other Atomic Context we had found in our modelling, Agree_Intial_Install_Dt. This is also stored in the Name Value Table

Key	Name	Value
123	AGREE_INTIAL_INSTALL_DT	2013-03-01

We designate the parameters according to the Atomic Context.

```
SELECT
KEY AS SUBS_ID
MAX(WHEN NAME='AGREE_INTIAL_INSTALL_DT' THEN NAME ELSE NULL END)CD_NAME
MAX(WHEN NAME='' THEN VALUE ELSE NULL END)CD_START_DATE
MAX(WHEN NAME='' THEN VALUE ELSE NULL END)CD_END_DATE
MAX(WHEN NAME='' THEN VALUE ELSE NULL END)UOM
MAX(WHEN NAME='AGREE_INTIAL_INSTALL_DT' THEN VALUE ELSE NULL END)CD_VAL
FROM NV_TABLE
GROUP BY
KEY
```

When we have loaded out staging table for Subscription Conditions with both of the atomic contexts it looks like this.

SUBS_ID	CD_NAME	CD_START_DATE	CD_END_DATE	CD_VAL	UOM
123	BINDING PERIOD	2014-06-01	2017-05-31	24	Month
123	AGREE_INTIAL_INSTALL_DT			2013-03-01	

Figure 13: DDW.SUBS_ID

By applying the neutralizing the source and have a specific structure in the target, we can with one CD job load any attribute that has been designated to an Focal Condition Descriptor and this holds true for all jobs in the Focal pattern.

The Detailed Data Warehouse Layer (DDW)

The DDW layer is the persistent raw data layer where business key integration is done.

The Focal jobs to move data from STG to DDW has two main purposes.

1. Integrate data on business key(s)
2. Delta logic

Once again, since the Focal jobs knows both their source and target (the tables are following the focal pattern) we can move all data with the use of 10 jobs.

Business Data Warehouse Layer (BDW)

The BDW layer is persistent and is the final Focal model layer and its purpose is to hold the fully integrated data too be used for analytical requirements.

Once again, since the Focal pattern is applied in DDW as well as BDW we can load the BDW layer with only 10 jobs.

The main purpose of the BDW jobs are to semantically integrate data from different sources. This is done through the use of the Semantic model (see picture Focal Architectural Layers).

The Semantic Model

The semantic model is persistent and keeps history. It stores the information model of the Data Warehouse and it is used for many different things. One of the main purpose of the semantic model is to hold the local attributes and the integration attributes and the mapping between these. The Semantic Model is also implemented according to the Focal modelling pattern.

Semantic integration

The BDW jobs integrate data according to the semantic model. A simplified transposed view of the Semantic model and the mapping between local attributes and central attributes can look like this

Source Attribute	Source definition	Target definition	Target Attribute

Figure 14: Transposed view of the semantic model

The Source attribute, is the name of the attribute in the source system and the Target attribute is the name of the attribute from a central information model.

Let's use the example of the source attribute – AGREE_INITIAL_INSTALL_DT which integrates into the central attribute SUBS_START_DATE (The start date of the subscription).

Source Attribute	Source definition	Target definition	Target Attribute
AGREE_INITIAL_INSTALL_DT	…	…	SUBS_START_DATE

The BDW job just access a table pattern and searches for mapped attributes in the DDW table to load them into the BDW layer – semantically integrated. Remember how the DDW.SUBS_CD table was loaded.

SUBS_ID	CD_NAME	CD_START_DATE	CD_END_DATE	CD_VAL	UOM
123	BINDING PERIOD	2014-06-01	2017-05-31	24	Month
123	AGREE_INTIAL_INSTALL_DT			2013-03-01	

Figure 15: DDW.SUBS_CD Table

Now with use of our mapping logic, where only AGREE_INITIAL_INSTALL_DT has been mapped.

SQL example

```
SELECT
DCD.SUBS_KEY,
SEM.TGT_ATTRIBUTE,
DCD.CD_START_DATE,
DCD.CD_END_DATE,
DCD.CD_VAL,
DCD.UOM
FROM DDW.SUBS_CD DCD
JOIN SEMANTIC_MODLE SEM
ON DCD.CD_NAME=SEM.SOURCE_ATTRIBUTE
```

The SQL result loaded into the BDW.SUBS_CD table will look like this

SUBS_ID	CD_NAME	CD_START_DATE	CD_END_DATE	CD_VAL	UOM
123	SUBS_START_DATE			2013-03-01	

Figure 16: BDW.SUBS_CD Table

The new row in the BDW layer has been semantically integrated. Now add a new source system with its own start date attribute into the DDW.SUBS_CD table and its mapping and see what happens.

SUBS_ID	CD_NAME	CD_START_DATE	CD_END_DATE	CD_VAL	UOM
123	BINDING PERIOD	2014-06-01	2017-05-31	24	Month
123	AGREE_INTIAL_INSTALL_DT			2013-03-01	
432	STR_DATE_IN_NET			2014-08-01	

Figure 17: DDW.SUBS_CD Table -with new attribute

As seen in, the DDW.SUBS_CD table has a new row in it with a new Attribute from a new system – STR_DATE_IN_NET. In the mapping process it is identified that this means the start date to the subscription, so it's added into the semantic model.

Source Attribute	Source definition	Target definition	Target Attribute
AGREE_INITIAL_INSTALL_DT	…	…	SUBS_START_DATE
STR_DATE_IN_NET	…	…	SUBS_START_DATE

Figure 18: Semantic View - new mapping

As seen in the semantic mapping the new attribute also stands for SUBS_START_DATE. Then the BDW_CD job

```
SELECT
DCD.SUBS_KEY,
SEM.TGT_ATTRIBUTE,
DCD.CD_START_DATE,
DCD.CD_END_DATE,
DCD.CD_VAL,
DCD.UOM
FROM DDW.SUBS_CD DCD
```

```
JOIN SEMANTIC_MODLE SEM
ON DCD.CD_NAME=SEM.SOURCE_ATTRIBUTE
```

And run the result in the BDW.SUBS_CD will be.

SUBS_ID	CD_NAME	CD_START_DATE	CD_END_DATE	CD_VAL	UOM
123	SUBS_START_DATE			2013-03-01	
432	SUBS_START_DATE			2014-08-01	

Figure 19: BDW.SUBS_CD with new row

The different sources has been semantically integrated without any code or table changes. As soon as a new attribute enters the Data Warehouse and it is mapped to a central attribute, it gets integrated and ready for reporting without any changes in code.

Enterprise Data Mart Layer (EDM)

Now we want to give the user access to the data for reporting. If we move back to the time when we only had one source of subscription and only one subscription. This example we build a Subscription Dimension with one attribute – SUBS_START_DT.

Access the BDW.SUBS_CD and select the attribute that you want to lift to the Dimension.

```
SELECT
F.SUBS_ID,
C.SUBS_START_DATE,
…
FROM DDW.SUBS_FOCAL F
LEFT JOIN
(SELECT
 G.SUBS_ID
 G.CD_VAL AS SUBS_START_DATE
FROM BDW.SUBS_CD G
WHERE CD_NAME='SUBS_START_DATE'
)C
ON F.SUBS_ID=C.SUBS_ID
```

With this SQL pattern we can add a new left join for any new attribute we want to add to the dimension. The important part here is that when the new source enters our Data Warehouse and the BDW.SUBS_CD gets populated, the code above will pick it up, and no new code is needed to be added.

SUBS_ID	SUBS_START_DATE	…
123	2013-03-01	

432	2014-08-01	

Figure 20: SUBS_DIM Table

The big picture

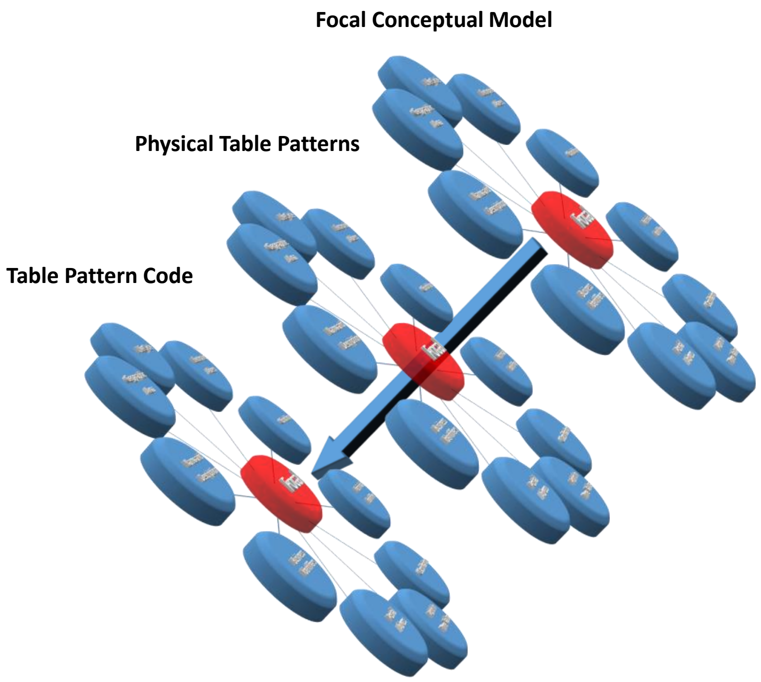
All these details might make it hard to understand how this has anything to do with automation of Data Warehouse. So let's look at it from a step by step, what needs to be done to ingest and integrate data into the Focal Data Warehouse and make it ready for reporting.

1. Do the information modelling process
2. Apply the result of the information modelling processes as parameters to the staging job(s)
3. Load the data into STG and DDW
4. Apply the semantic mapping exercise in the semantic model (source→target)
5. Load the BDW layer with the use of the semantic mapping
6. **Up to here we have not seen any new/changed code. Data just flows in and gets integrated**
7. Load the Dimension/Fact – once the code for this is built – we can load any system using the same code since the BDW layer is system agnostic.

The two steps that can't get automated are the information modelling process and the semantic mapping exercise.

Conclusion

The pattern based design implemented based on the information modelling process is an extremely powerful way of building a fully integrated Data Warehouses.


Focal Conceptual Model
Physical Table Patterns
Table Pattern Code