# Model Driven Software Engineering for Data Warehousing — Part 2: Data Warehousing

The more you know about the past, the better prepared you are for the future.

— Theodore Roosevelt

This article is the second part of a series of articles in which I want to give an overview of how I think Model Driven Software Engineering (MDSE) can be used for data warehousing.

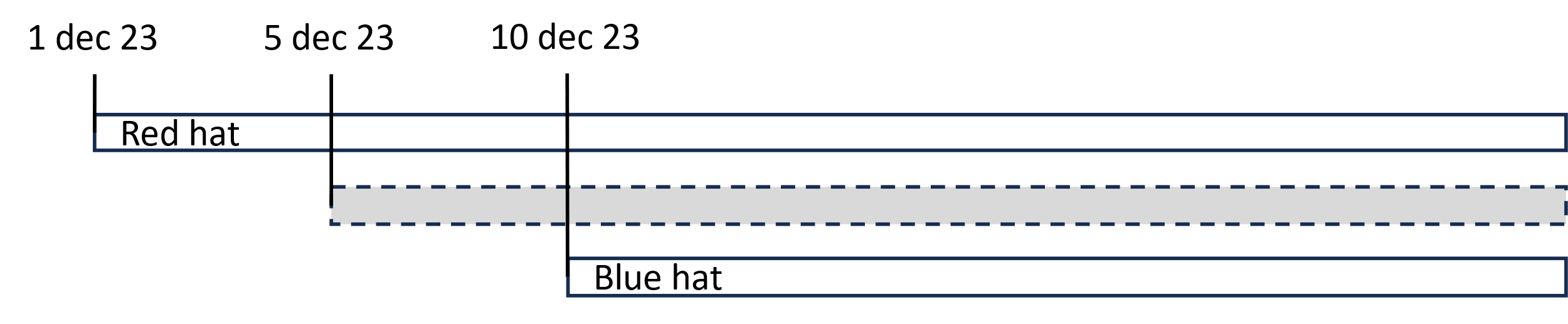
This article is about todo:DESCRIPTION.

Last but not least the topics described in this article are explained using an example.

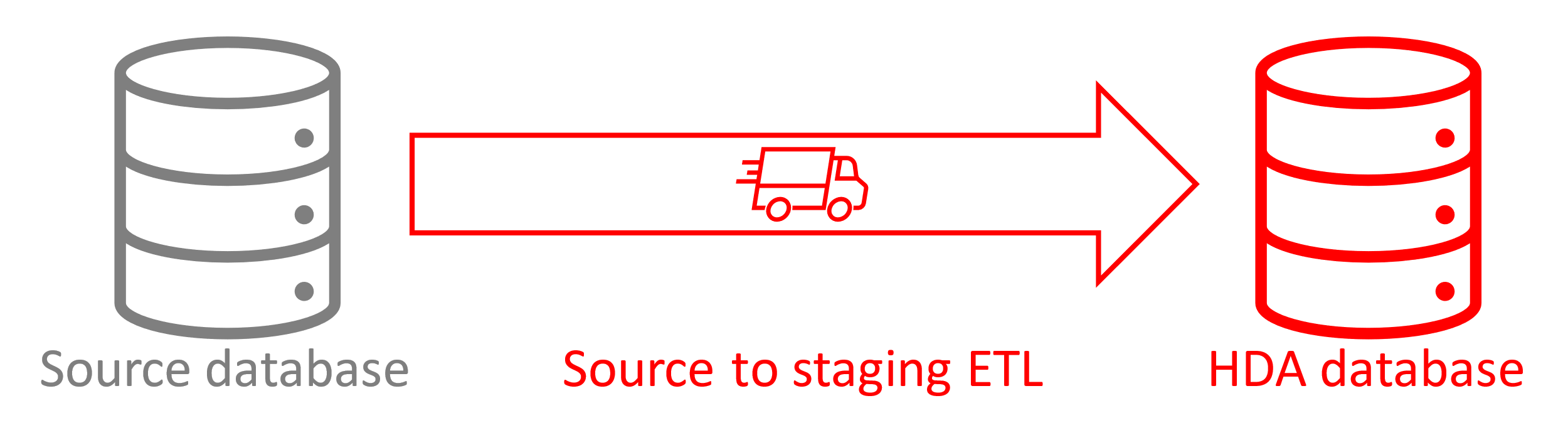
todo:INTRODUCTION

## A Brief Introduction into History

In this article we



## Generating Data Warehouse Artifacts

Now, we also want to start to work on our actual data warehouse.

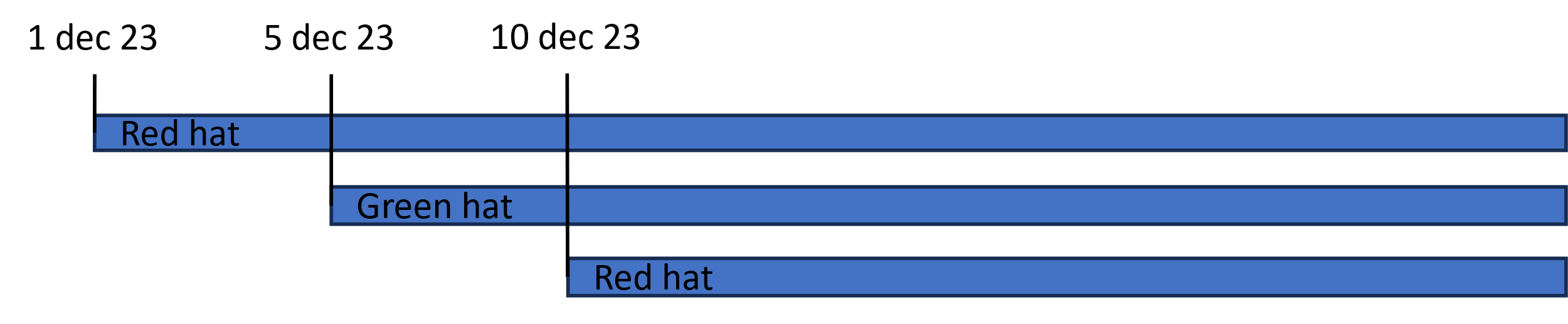
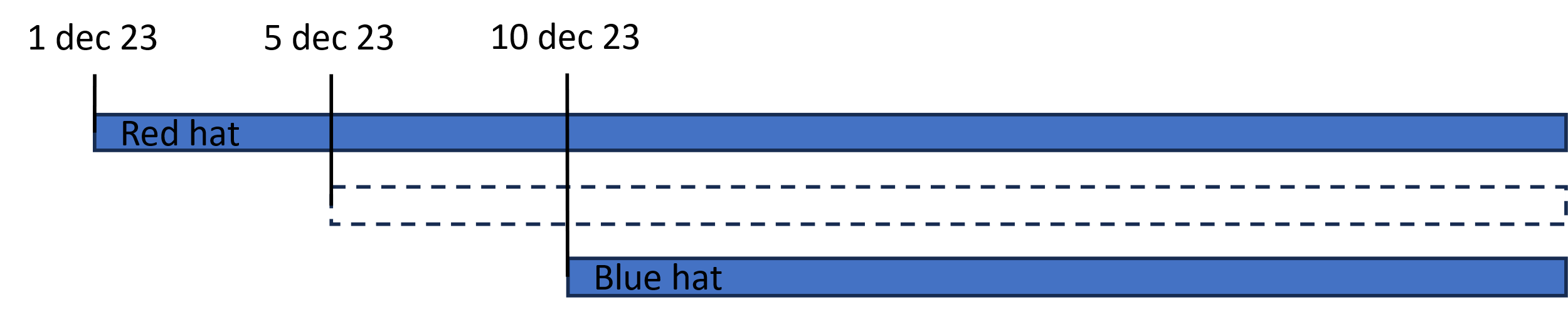
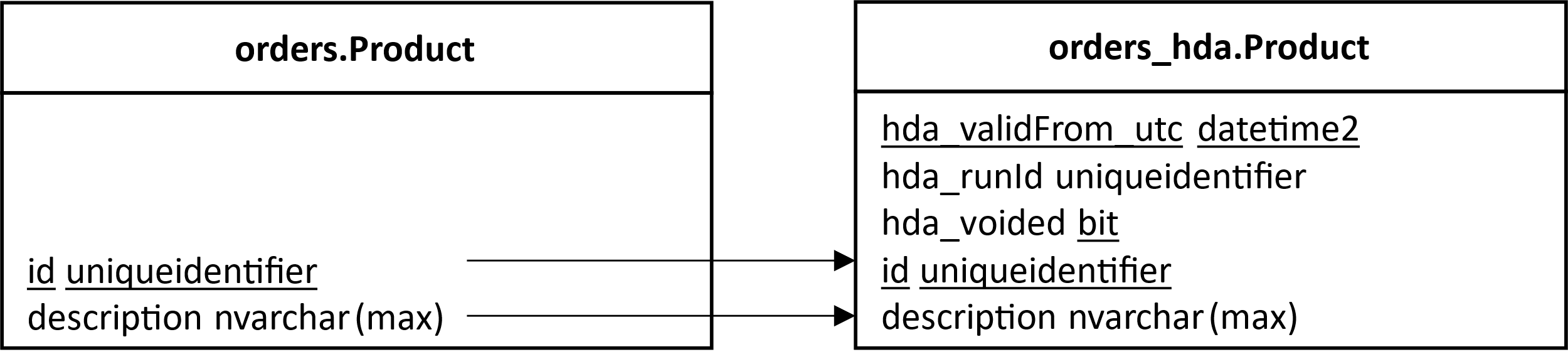
To do this, first make sure we can actually store the data in HDA table we transform our previous database model into HDA database model and generated DDL code. Below a very simple Product table from an order system is shown. The left most table is the source table, we’ll assume this already exists (it was generated with knowledge from my previous article!).

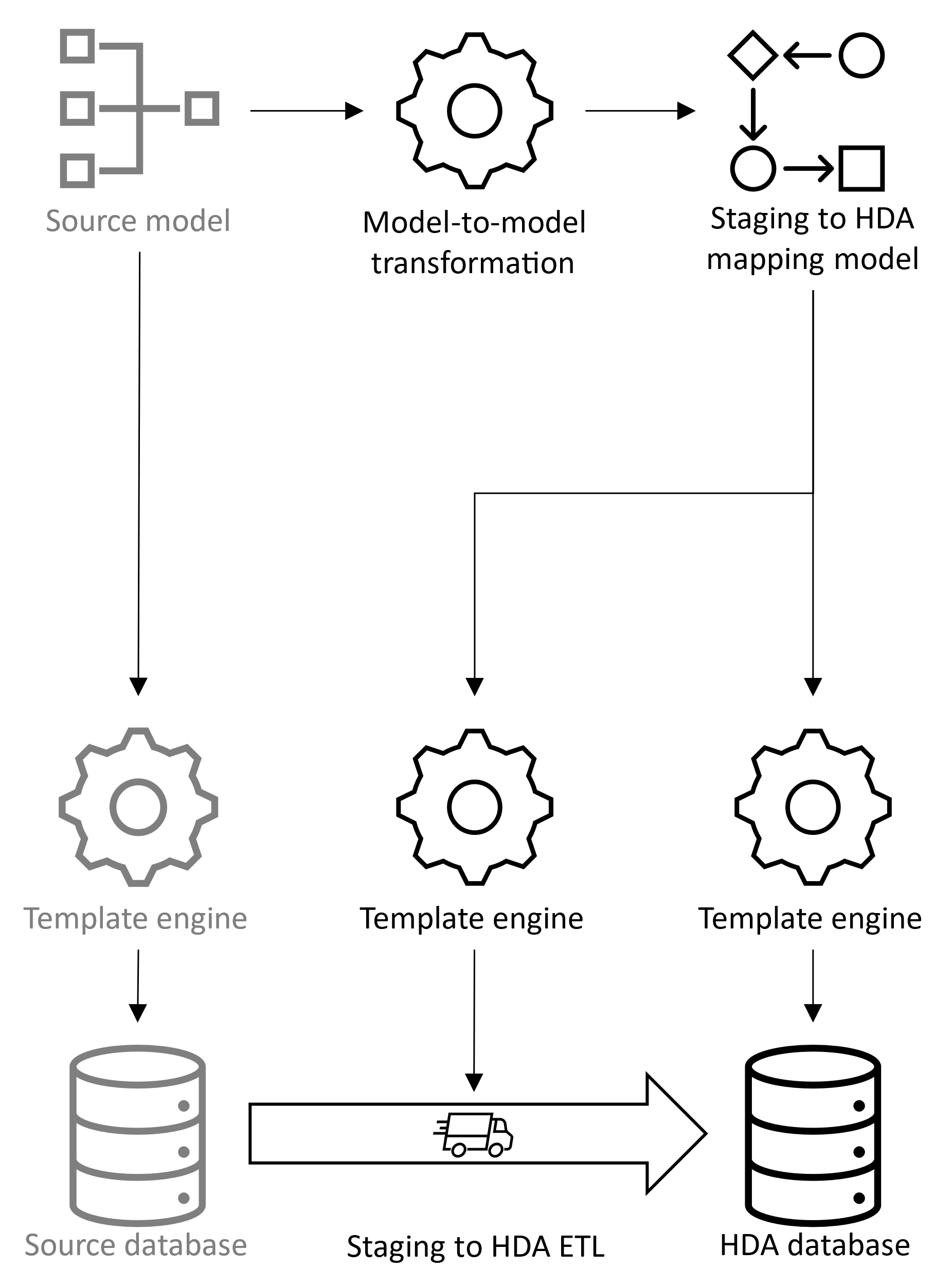
Because our table is so simple, it can is not possible to track changes to products in our table through time. But, we would really like to keep track of these changes, so we can report on changes in the description, and if we increase complexity of our table, about the impact on sales or even to predict future sales! It is time to develop what we at Nippur call a *Historical Data Archive (HAD)*! It does exactly what we want, it stores the source data together with information about when things changed. An example of the HDA table for our product table is shown on the right hand side of the image below.

We added two columns to the table, a hda\_validFrom\_utc column indicating the exact date and time source data changed, a had\_runId column that we can used to couple to a logging table to keep track of every bit of processing data one could wish for. Moreover, we want to know when a product was removed from our source table, that’s why I added to had\_voided column. And, of course we need to store the actual source data.

There is also a bit of logic behind the loading of the HDA table.

* Store only changes (key is original key + hda\_validFrom\_utc)
* Picking a point in time (PIT)

The image above depicts the process of model generation and code generation that we follow. In my previous article I described to process of generating database code from a source model, so let’s assume that this is a given. This article focus more on increasing the added value of the generation process, because as stated before, generation becomes much more beneficial when the amount of generated stuff increases.

## Architectural considerations

Introducing a staging model, it was left out due to simplicity

* Variation on types of source systems
* Fast loading of data to target environment
* Limit strain on source systems

End dating of HDA rows

Differences between HDA and temporal tables