INTEGRATION OF SAP ERP DATA INTO A COMMON DATA MODEL

Author: Julius von Ketelhodt, initions GmbH
Date: 26.11.2020

1. Introduction

In the past, data were often stored in differing formats across an enterprise, leading to data integration challenges. To unify data formats, Microsoft, SAP and Adobe have agreed to pursue an Open Data Initiative (https://info.microsoft.com/Open-Data-Initiative.html). Part of this initiative is to develop a Common Data Model (https://github.com/microsoft/CDM).

The purpose of the Common Data Model (CDM) is to store information in a unified shape, which consists of data in CSV or Parquet format, along with describing metadata JSON files. In the Microsoft landscape the CDM is already supported and implemented in different products and services such as Dynamics 365, PowerApps and Power BI, which leverage the Common Data Service. Many standard entities have been defined which offer structural and semantic consistency and simplify data integration into CDM structures. These benefits are often missed when extracting plain SAP tables on their own. The standard CDM entities can be browsed here: https://microsoft.github.io/CDM.

However, there is no one-click solution offered by SAP to integrate ERP or BW data into a CDM structure and thereby conveniently combine it with services and other CDM-based data, such as Dynamics 365. In this blog post I close this gap and outline the necessary steps to (1) extract SAP ERP data into a CDM format and (2) consume this data conveniently with machine learning services such as Databricks, data warehousing applications such as Synapse Analytics and Power Platform applications, such as Power BI for data visualisation (Figure 1.1).

In the extraction process for SAP objects I choose to use the Operational Data Provisioning (ODP) framework (https://blogs.sap.com/2017/07/20/operational-data-provisioning-odp-faq/). The ODP framework bundles many advantages for extraction scenarios such as their availability by default, a delta load enablement, a harmonized extraction layer over multiple applications (ERP / BW), and the extraction of objects instead of plain tables. Please contact initions for further details on SAP ODP extraction scenarios.

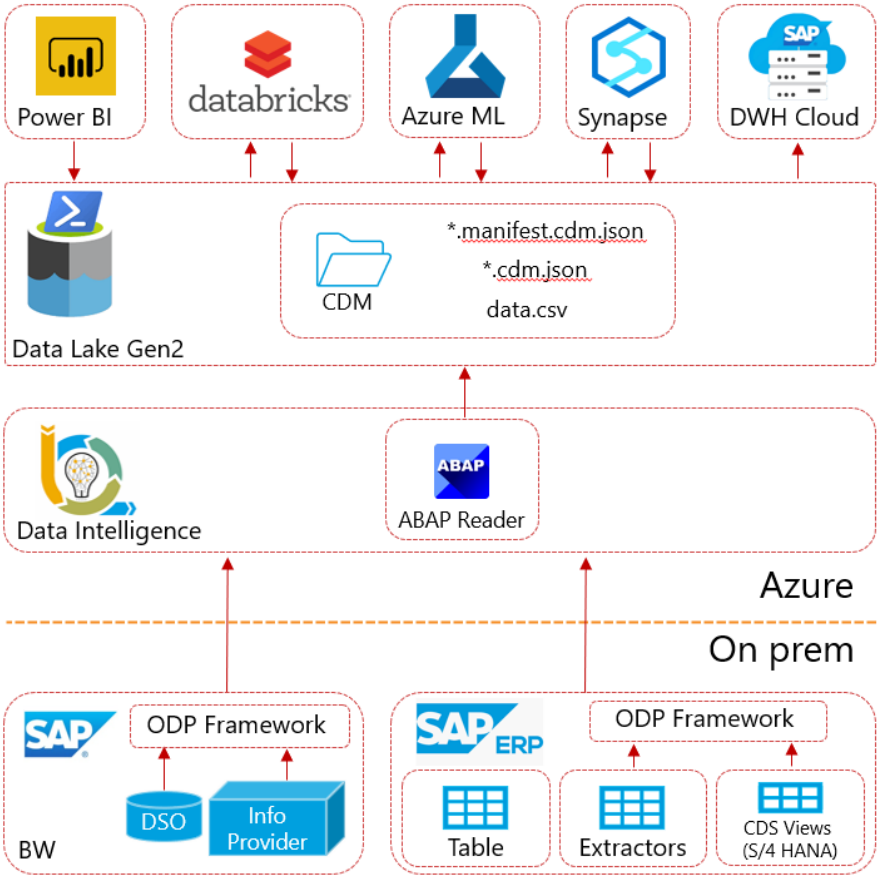


Figure 1.1: Dataflow proposition to extract SAP ERP (or BW) data and write via ODP framework into CDM format on an Azure Data Lake Gen 2. This ensures convenient consumption by a variety of tools, such as Power BI, Databricks, Azure ML and Synapse Analytics. On premises ERP systems can also include SAP on Azure (IaaS). Please note that consumption by SAP Data Warehouse Cloud is only supported for CSV files and not the entire CDM structure.

2. Extraction of ERP Data using SAP Data Intelligence

SAP Data Intelligence (https://www.sap.com/products/data-intelligence.html) is the next generation data integration tool from SAP and already widely used as a machine learning and big data tool (Getting Started: https://blogs.sap.com/2019/08/14/sap-data-intelligence-create-your-first-ml-scenario/). The installation of Data Intelligence is possible on an Azure Kubernetes cluster and upon completion, the tool is accessible via a browser (Figure 2.1).

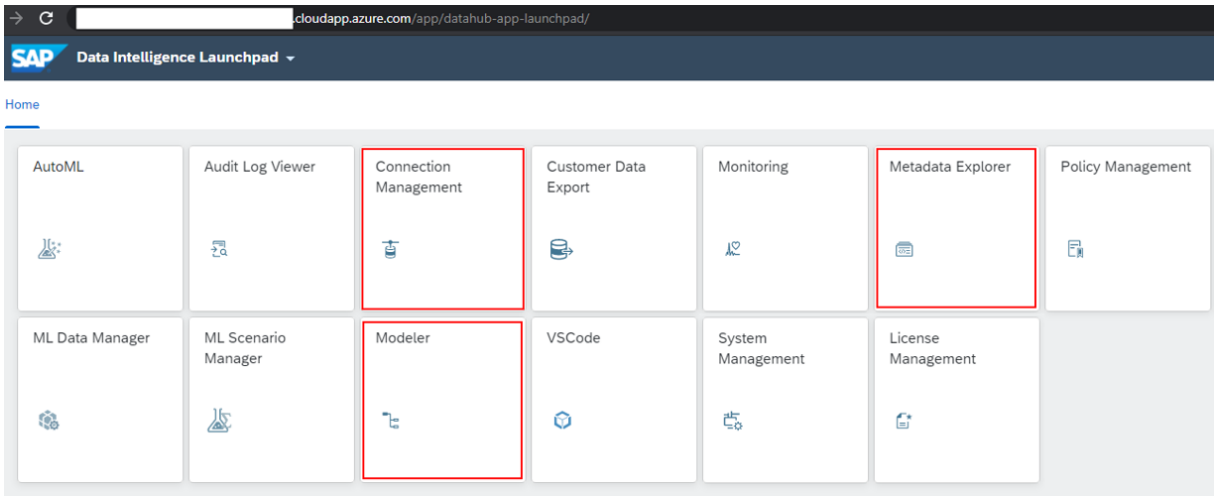


Figure 2.1: SAP Data Intelligence launchpad with the Connection Manager, the Metadata Explorer, and the Modeler.

The ERP data extraction is accomplished by (1) creating a connection to an ERP or BW system, (2) extracting the data by using standard ODP extractors, and (3) writing the data into CDM format on an Azure Data Lake Gen 2.

2.1. Add connections

The first step is to enable communication with your SAP ERP system, the source, and with an Azure Data Lake Gen 2, the destination. These two connections can be created in the Connection Manager. An example of creating an ABAP connection via RFC to the ERP system is shown in Figure 2.2.

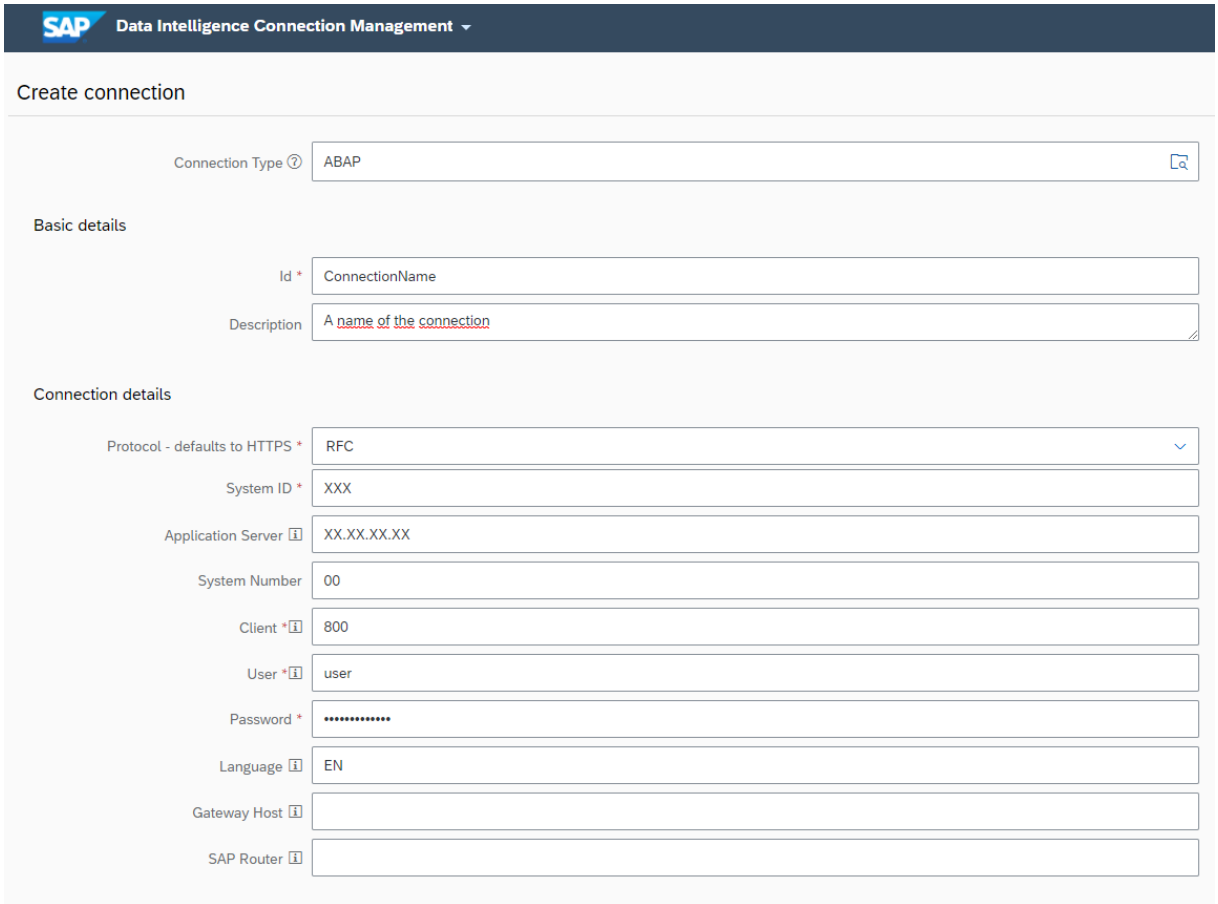


Figure 2.2: Example of establishing an ABAP connection to an SAP ERP system.

Establishing a connection to an Azure Data Lake Gen 2 is analogous (Connection Type ADL_V2).

2.2. Preparations of the SAP ERP system

In order to use SAP Data Intelligence operators and functionalities, the ERP system needs to be updated by applying a few SAP Notes. This process is dependent on a few factors, such as the ERP version (S/4 HANA or classic ERP with Netweaver) and installed support packages.

2.3. Metadata catalogue

Once a connection is established, and pre-requisites installed on the ERP side, the ODP extractors can be viewed with the Metadata Explorer. Chose Browse Connections, select your ABAP ERP connection, and then the ODP objects (BW and SAPI), Tables and Views will be accessible (Figure 2.3).

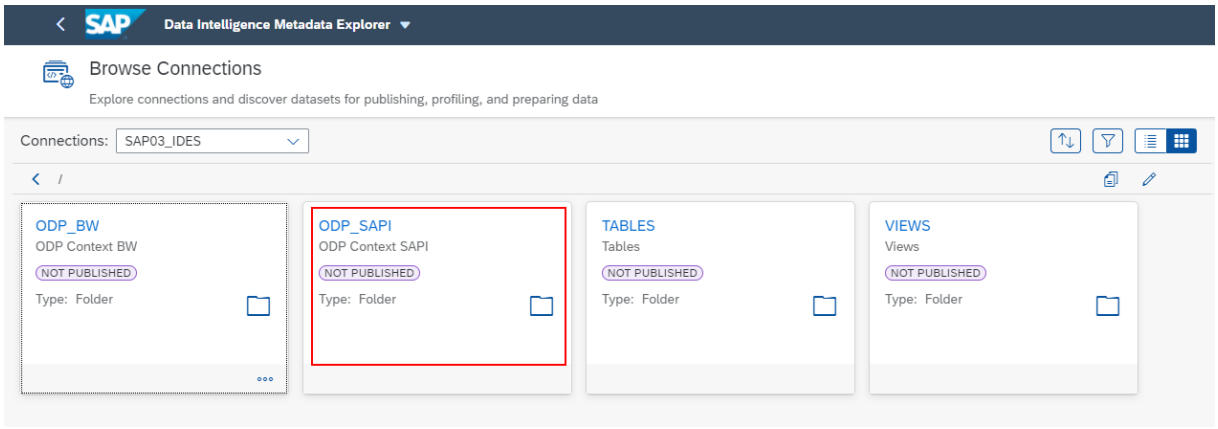


Figure 2.3: Overview of ERP objects in the Metadata Explorer.

Drill down to an ODP SAPI object of choice that you would like to extract. In this example I extract material data (MARA and related tables) by accessing the two ODP objects 0MATERIAL_ATTR and 0MATERIAL_TEXT. The ODP object metadata alongside a data preview is shown in Figure 2.4 and Figure 2.5.

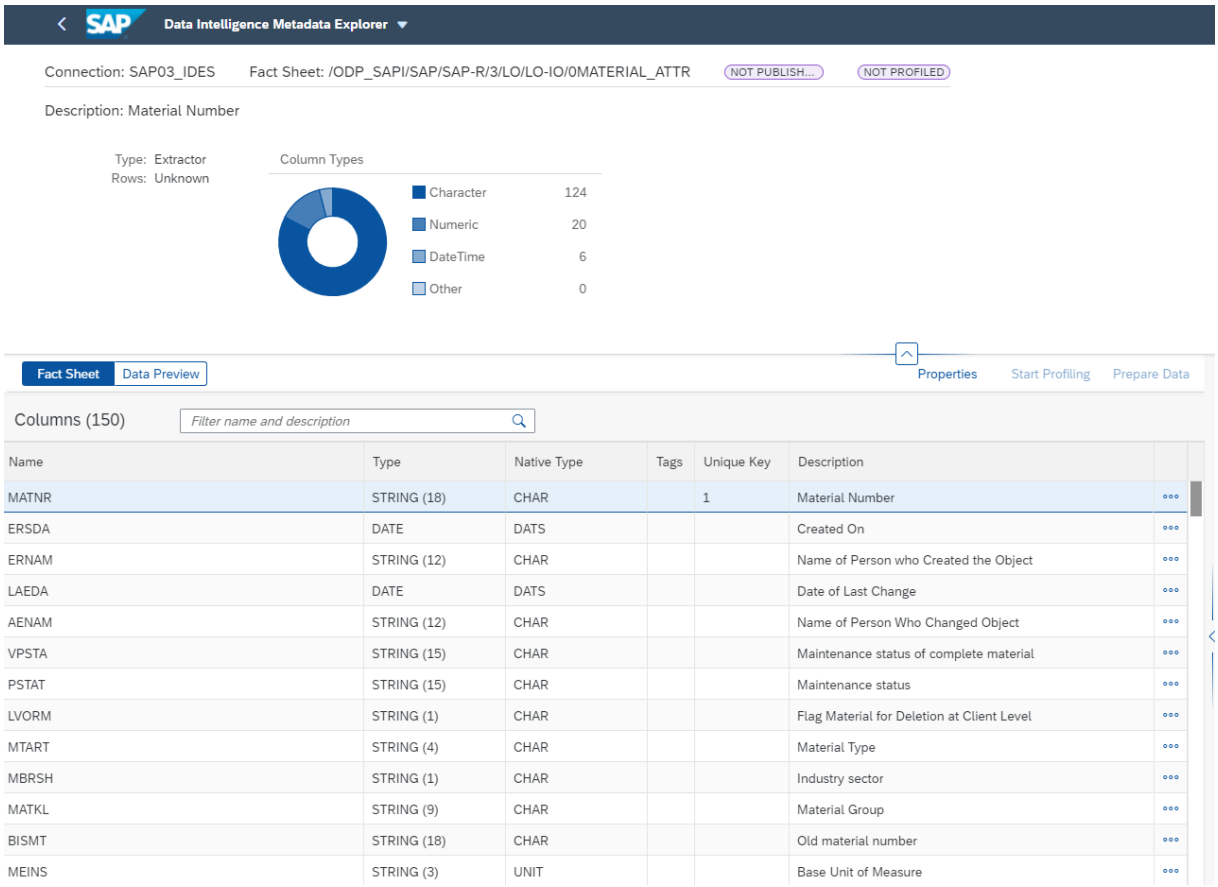


Figure 2.4: Viewing the columns and distribution of column data types for ODP Object 0MATERIAL_ATTR in the Metadata Explorer

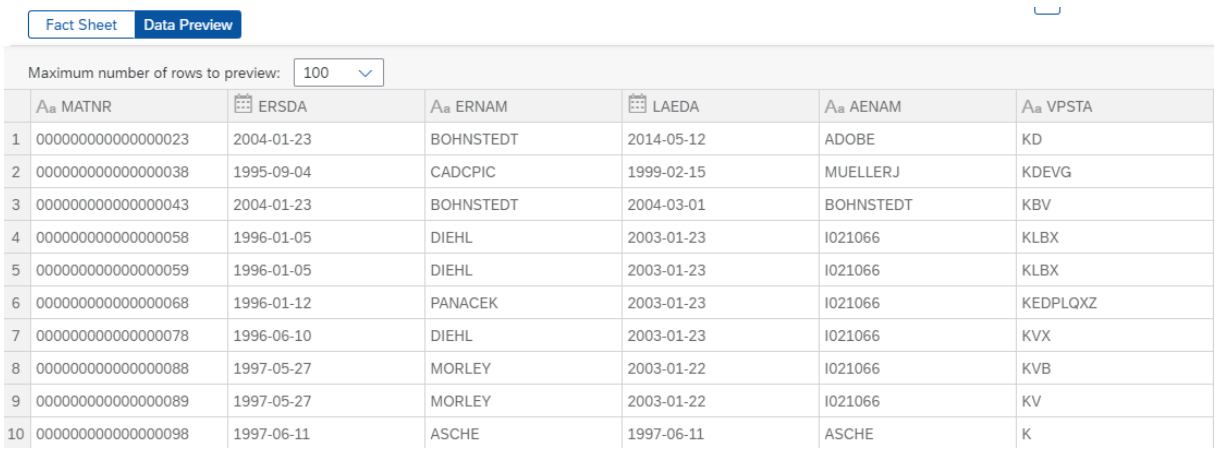


Figure 2.5: Data preview of the ODP Object 0MATERIAL_ATTR in the Metadata Explorer.

2.4. ODP extraction and CDM creation

The workflow of generating CDM entities from ODP objects is realised with several different operators in the Data Intelligence Modeler. This workflow (or graph in SAP DI terminology) is shown in Figure 2.6. Here different operators are used to (1) read data from a parameterised ODP object (ABAP ODP Reader), (2) transform the data from type ABAP to string (ABAP converter), (3) Map the data into CDM format (Please get in touch with initions to learn further details about the Python script), (4) write the CDM data (Write CSV) and describing metadata (Write JSON) to the Azure Data Lake and (5) finally terminate the workflow. This workflow is executed in a parameterised fashion for each ODP object that is loaded into the CDM. In addition, the delta capabilities of the ODP extractors

enables a scheduled execution to keep the CDM up to date. Part of the CDM manifest alongside the metadata for the ODP object 0MATERIAL_ATTR is shown in Figure 2.7 a) and b).

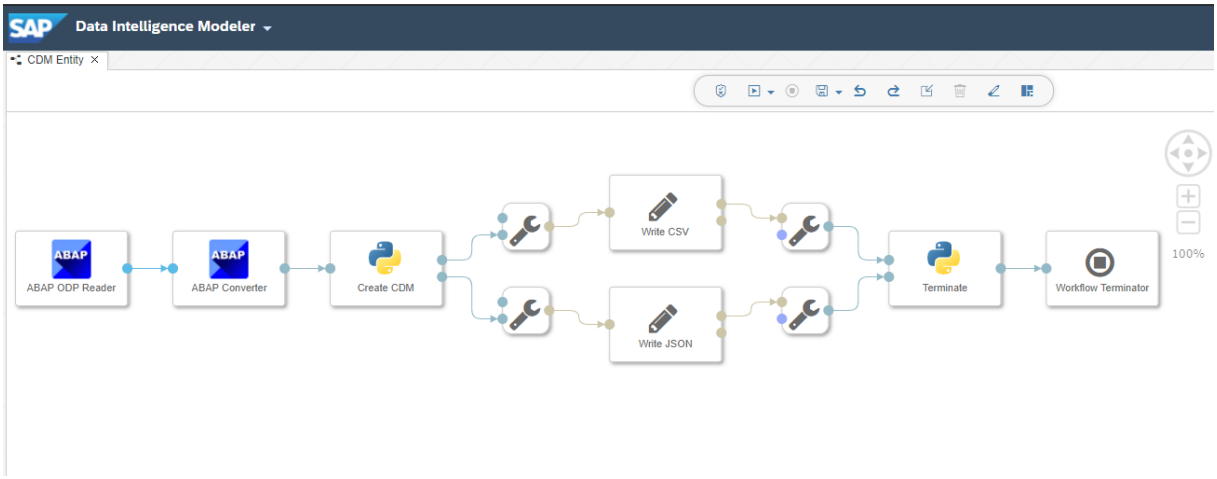


Figure 2.6: Workflow for creating a CDM entity in the Data Intelligence Modeler.

Figure 2.7: The CDM metadata files that were genereted by the Data Intelligence graph are shown, with the manifest (a) and the metadata for the ODP object 0MATERIAL_ATTR now stored as entity MATERIALATTR (b).

3. Consuming the CDM

Multiple services support the CDM format. These include different tools such as Databricks for machine learning scenarios, Synapse Analytics for data warehousing purposes and Power Platform applications such as Power BI for data visualisations. An introduction to loading CDM files into these

tools is described subsequently. In Figure 1.1 the SAP Data Warehouse Cloud was mentioned as a consumer. This only applies to the CSV files. For further information on SAP DWH Cloud, please see this blog post: [Loading the data from Microsoft Azure into SAP HANA Cloud, Data Lake | SAP Blogs]

3.1. Databricks

Databricks has evolved into a preferred tool for data scientists, since it offers a convenient environment, easily integrates with the Microsoft stack and is available on Azure. By importing data in the CDM format to Databricks offers the SAP semantics and structures conveniently to data scientists who are otherwise not fluent in the SAP domain. Reading data in the CDM format is possible by using a Spark-CDM-Connector library, which can be installed to clusters in Databricks. The GitHub repository can be found here: https://github.com/Azure/spark-cdm-connector.

To load a CDM entity, create a cluster (e.g. Databricks Runtime Version 6.6, since Spark 3.0 is not yet supported). Then install the Spark-CDM Maven repository, as shown in Figure 3.1.

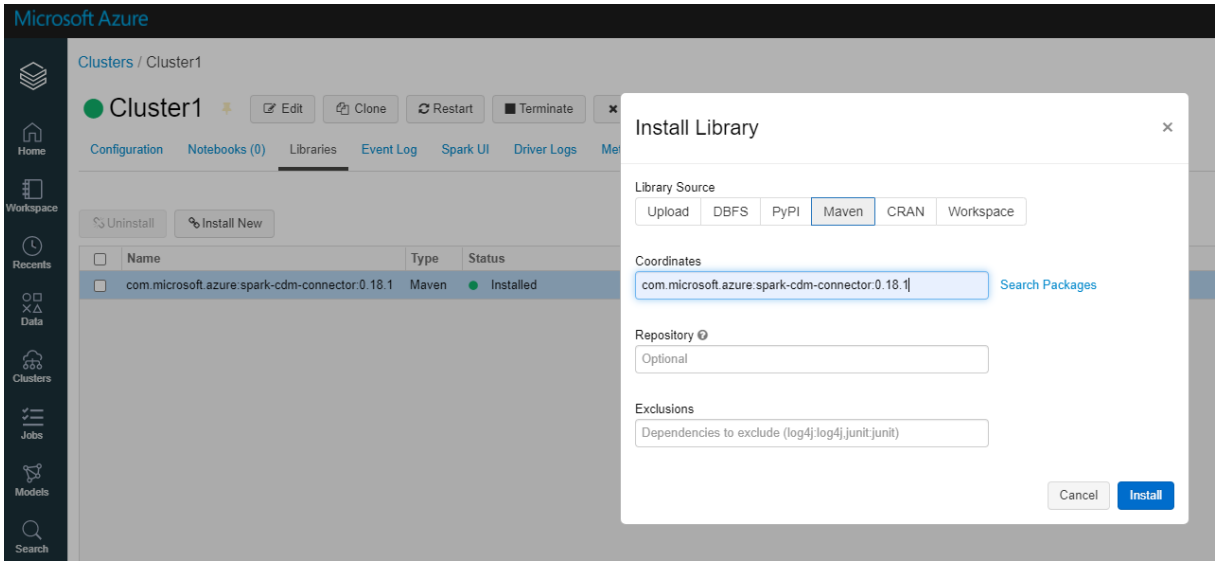


Figure 3.1: Adding the Spark-CDM-Connector library in the form of a Maven repository to a Databricks cluster.

After connecting to your Datalake, the data can be read into a data frame using the following Python commands:

```
readDf = (spark.read.format("com.microsoft.cdm")
  .option("storage", storageAccountName)
  .option("manifestPath", container + "/Path/default.manifest.cdm.json")
  .option("entity", "MATERIALATTR")
  .option("appId", appid)
  .option("appKey", appkey)
  .option("tenantId", tenantid)
  .load())

readDf.select("*").show()
```

3.2. Azure Synapse

The CDM format can recently be accessed by dataflows in Azure Data Factories. Since Data Factory capabilities are included in Synapse Workspaces, this functionality can be used to load CDM data into

a table of a Synapse Database in the form of a SQL Pool. A minimal dataflow example is shown in Figure 3.2.

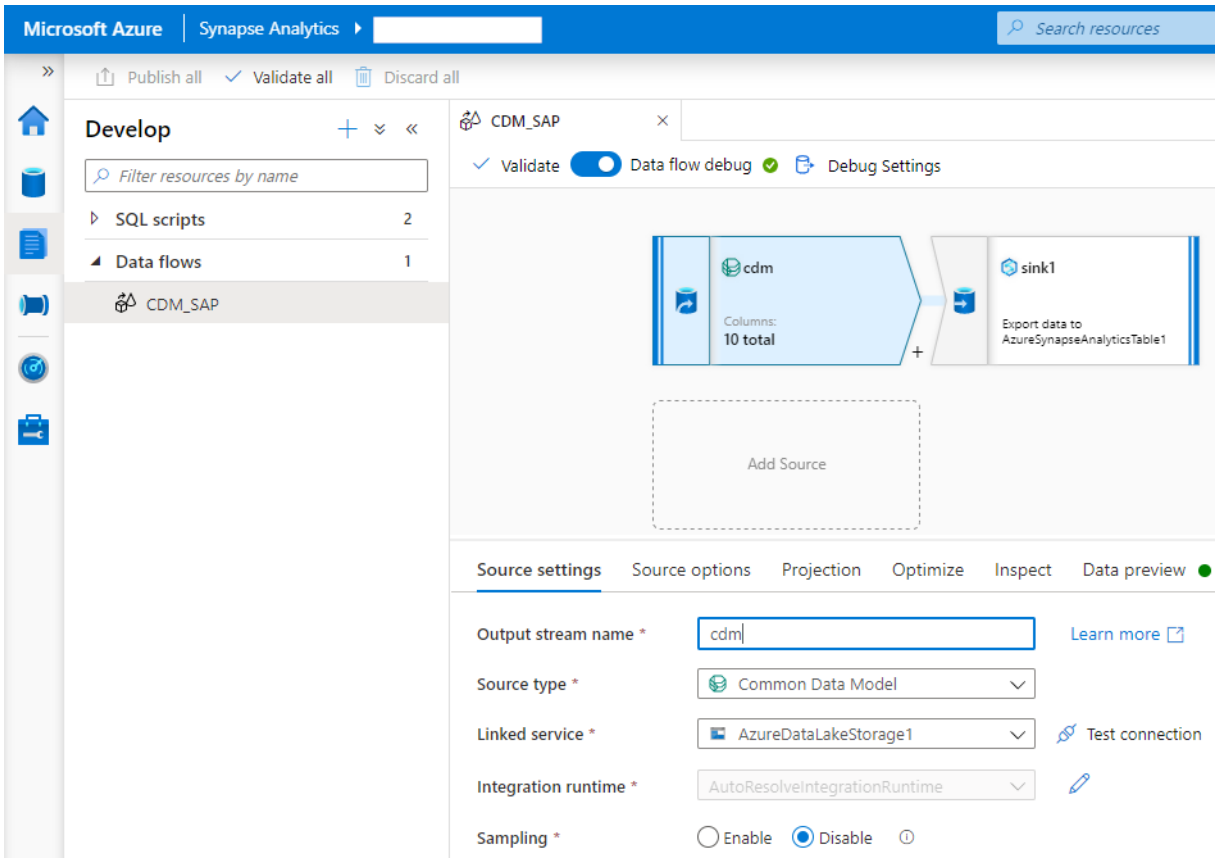


Figure 3.2: Loading the CDM into a Synapse data warehouse table.

3.3. Power BI

The functionality of consuming a CDM is currently in preview (October 2020 version) in Power BI. To load a CDM, select Get Data and choose Azure Data Lake Storage Gen2 (Figure 3.3). Subsequently, enter the URL of the Data Lake (https://<datalakeName>.dfs.core.windows.net) and choose CDM Folder View. Upon entering your Data Lake credentials, the CDM entities can be selected for import (Figure 3.4).

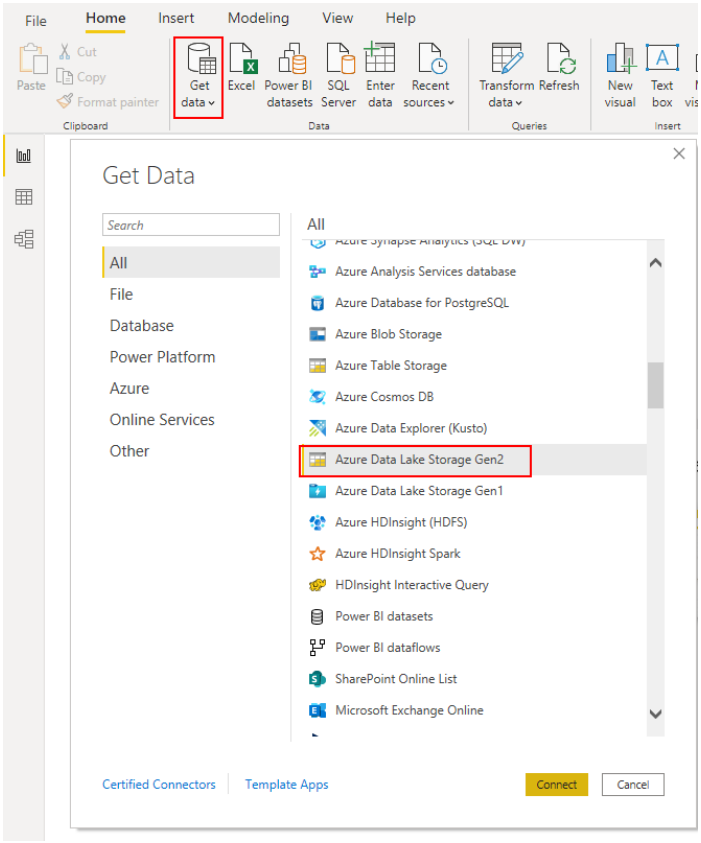


Figure 3.3: Connecting the Azure Data Lake Gen 2 where the CDM is stored to Power BI.

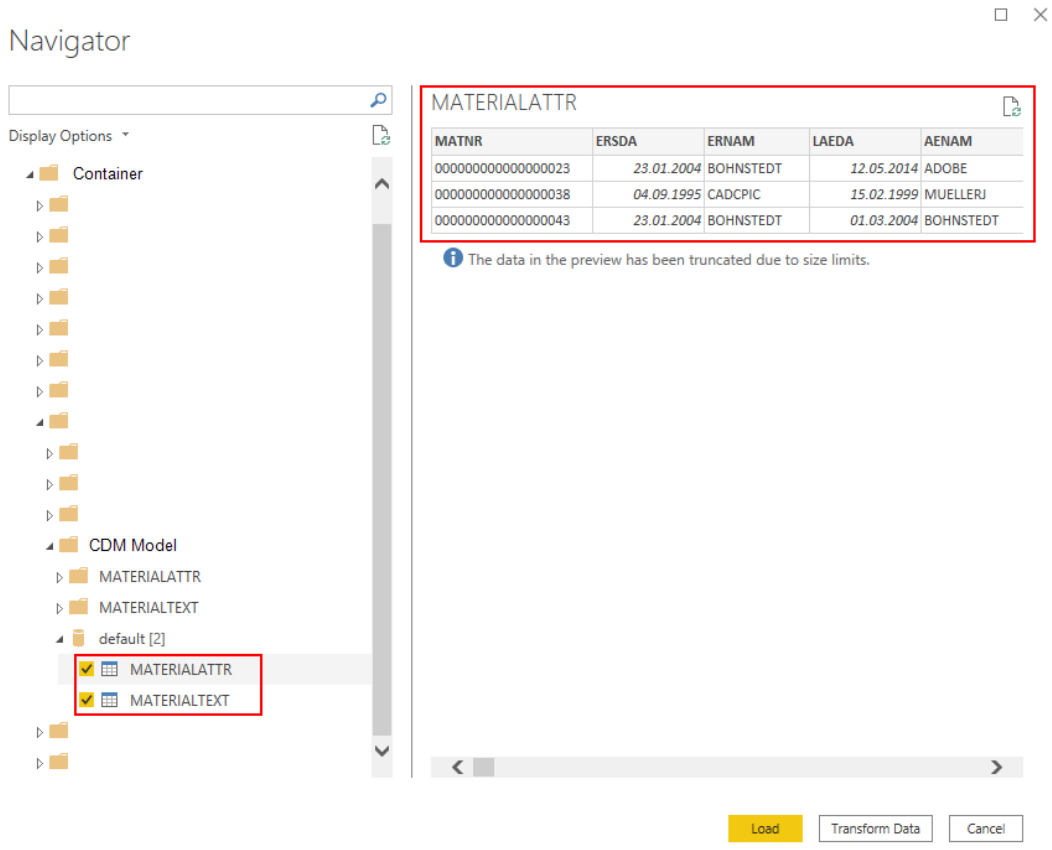
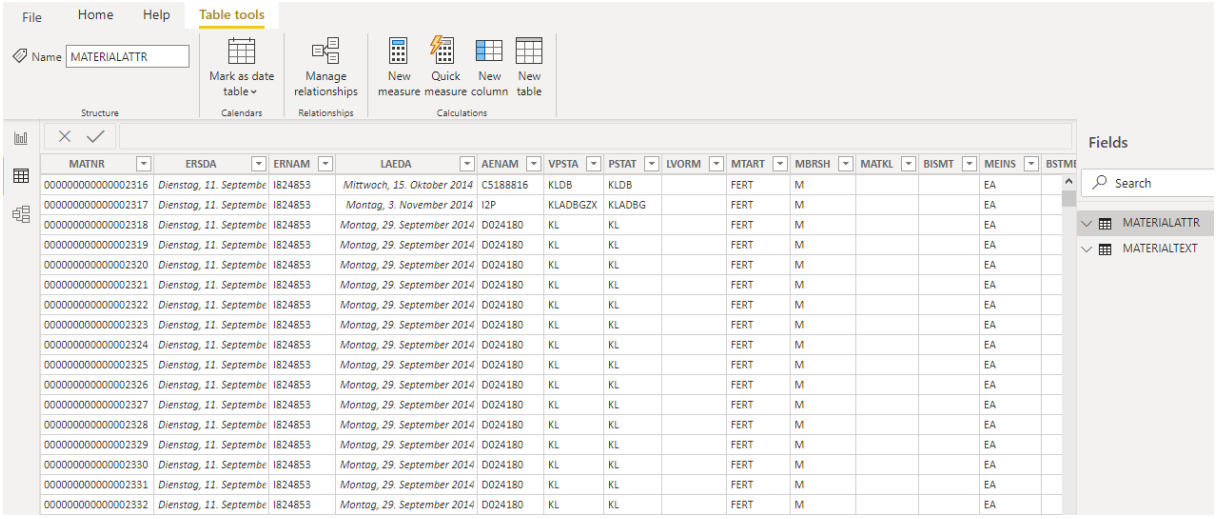


Figure 3.4: Selecting the CDM entities (left) and previewing the data (right) prior to loading to Power BI.

Now the CDM data are imported to Power BI with the correct data types and ready for visualisation or integration with additional data (Figure 3.5).



MATNR	ERSDA	ERNAM	LAEDA	AENAM	VPSTA	PSTAT	LVORM	MTART	MBRSH	MATKL	BISMT	MEINS	BSTME
000000000000002316	Dienstag, 11. Septembe	I824853	Mittwoch, 15. Oktober 2014	C5188816	KLDB	KLDB		FERT	M			EA	
000000000000002317	Dienstag, 11. Septembe	I824853	Montag, 3. November 2014	I2P	KLADBGZX	KLADBG		FERT	M			EA	
000000000000002318	Dienstag, 11. Septembe	I824853	Montag, 29. September 2014	D024180	KL	KL		FERT	M			EA	
000000000000002319	Dienstag, 11. Septembe	I824853	Montag, 29. September 2014	D024180	KL	KL		FERT	M			EA	
000000000000002320	Dienstag, 11. Septembe	I824853	Montag, 29. September 2014	D024180	KL	KL		FERT	M			EA	
000000000000002321	Dienstag, 11. Septembe	I824853	Montag, 29. September 2014	D024180	KL	KL		FERT	M			EA	
000000000000002322	Dienstag, 11. Septembe	I824853	Montag, 29. September 2014	D024180	KL	KL		FERT	M			EA	
000000000000002323	Dienstag, 11. Septembe	I824853	Montag, 29. September 2014	D024180	KL	KL		FERT	M			EA	
000000000000002324	Dienstag, 11. Septembe	I824853	Montag, 29. September 2014	D024180	KL	KL		FERT	M			EA	
000000000000002325	Dienstag, 11. Septembe	I824853	Montag, 29. September 2014	D024180	KL	KL		FERT	M			EA	
000000000000002326	Dienstag, 11. Septembe	I824853	Montag, 29. September 2014	D024180	KL	KL		FERT	M			EA	
000000000000002327	Dienstag, 11. Septembe	I824853	Montag, 29. September 2014	D024180	KL	KL		FERT	M			EA	
000000000000002328	Dienstag, 11. Septembe	I824853	Montag, 29. September 2014	D024180	KL	KL		FERT	M			EA	
000000000000002329	Dienstag, 11. Septembe	I824853	Montag, 29. September 2014	D024180	KL	KL		FERT	M			EA	
000000000000002330	Dienstag, 11. Septembe	I824853	Montag, 29. September 2014	D024180	KL	KL		FERT	M			EA	
000000000000002331	Dienstag, 11. Septembe	I824853	Montag, 29. September 2014	D024180	KL	KL		FERT	M			EA	
000000000000002332	Dienstag, 11. Septembe	I824853	Montag, 29. September 2014	D024180	KL	KL		FERT	M			EA	

Figure 3.5: Result of loaded CDM entities in Power BI.

4. Conclusion and Outlook

The described tools are currently undergoing development and many of the CDM connectors are only available in preview mode. Therefore, there are still a few limitations that will be resolved in future releases.

For example, SAP Data Intelligence, traditionally used for machine learning scenarios, is undergoing development towards an ETL tool with diverse data loading capabilities. For example, a current limitation is that CSV files have to use a comma as a delimiter.

The CDM format does allow for relationships to be defined between entities. However, the relationships must be single key (mapping one column from Table A to a column in Table B). In SAP systems it is common to have composite keys. The CDM model is currently being extended by this capability (https://github.com/microsoft/CDM/issues/192).

Despite these limitations, the advantages of CDM can already be tapped, thereby minimising data integration challenges. In this blog I demonstrated how this can be achieved by exporting SAP data into a CDM format and reuse it over multiple tools like Power BI, Databricks and Synapse. With a continuous development and refinement of these tools, there is more to come, such as generating SAP specific CDM folder structures and joining SAP and Dynamics data models.