Microsoft

Dataflows in Power BI

White Paper

Amir Netz, Technical Fellow

Published: November 6, 2018

Table of Contents

Table of Figures

Table of Tables

Introduction

Data preparation is considered the most difficult, expensive and time-consuming task of any Analytics and BI project. Most experts estimate the cost of the data preparation phase to be as much as 60%-80% of the time and cost of a typical project. In addition, the fragility of the ETL (**E**xtract, **T**ransform, **L**oad) processes often inflates the costs and complexity of the project throughout its lifetime.

For significant projects with large-scale data and complex cleansing and integration challenges, specialized expertise that is typically reserved for data warehousing professionals is required. That skill set is rare and expensive. Moreover, the data in such projects is often fragmented and dispersed across data sources, lacking structural or semantic consistency, and requiring complex system integrations to bring all the data together in an aligned and cohesive manner.

The new dataflows capability in Power BI can change the time, cost, and expertise required for data preparation in a fundamental way.

Dataflows in Power BI is a new extensive capability that allows business analysts and BI professionals to ingest, cleanse, transform, integrate, enrich and schematize data from a large set of transactional and observational sources. It handles the most complex data preparation challenges for users through its revolutionary model-driven calculation engine, cutting the cost, time and expertise required for such activities to a fraction of what they otherwise would be.

This document covers the basic concept and principles of the new dataflows capability in Power BI and is aimed at the following audiences:

- Business analysts and BI professionals who need to tackle medium and large-scale data preparation challenges in their projects
- Enterprise IT professionals who want to understand the architectural foundation of dataflows, and how they can integrate with and leverage them

Background: The Self-Service Revolution

A decade ago, a massive change occurred in the Business Intelligence industry. A new generation of tools aimed at business analysts started to compete with traditional enterprise BI technologies that were driven by IT professionals. BI tools back then were known for their complexity and the expertise required to master them. IT professionals had to learn dimensional modeling, MDX and semantic model design as they created multi-dimensional cubes in tools like Business Objects, SQL Server Analysis Services and Oracle Essbase.

Today, emerging self-service BI tools are aimed at allowing business analysts, most without any formal training in data modeling and OLAP technologies, to create their own BI models. To do that, a radically different approach to BI had to be invented.

At Microsoft we introduced PowerPivot – a BI modeling tool for Excel users. With PowerPivot we departed from the foundations that shaped SQL Server Analysis Service for 15 years – the multi-dimensional model, the MDX language and the MOLAP storage technology. Instead, PowerPivot

introduced a new and much simplified data model – the tabular model; an Excel like formula language – DAX; and brand-new storage and query processing technology – the VertiPaq in-memory database.

A decade later, these technological innovations allow tens of millions of Excel and Power BI users to enjoy BI technologies, working with massive amounts of data, using large scale sophisticated models, all without any formal BI training.

Microsoft wasn't alone in the transition from the IT-centric, complex BI products to simplified self-service offerings. Today the entire BI market is led by products that excel at the self-service capabilities.

While the BI market went through that fundamental transition to self-service, it mostly skipped the ETL market.

Until now.

Enter Dataflows

Dataflows are new artifacts in Power BI that sit side-by-side with datasets, dashboards and reports in a Power BI workspace. Unlike datasets and reports that are created in Power BI Desktop, dataflows are created and managed in the Power BI service, directly in the browser.

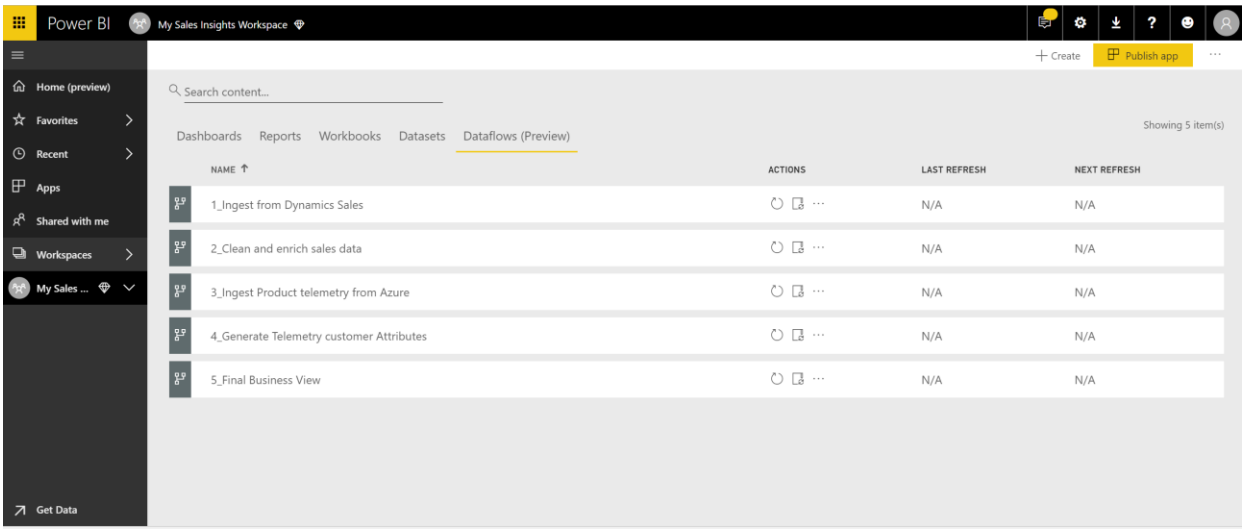

Figure 1: Power BI service

Power BI provided data prep capabilities as part of the dataset definition in Power BI Desktop since its launch. Power BI Desktop users use Power Query to connect to data, ingest it, and transform it before it landed in the dataset.

While Power Query in Power BI Desktop is a powerful tool, it also has limitations. First, the data ingested is locked in the Power BI dataset. There is no way to reuse the data or the logic in multiple datasets, or to connect to it with other data processing technologies such as Azure Databricks. Second, managing complex transforms at scale is difficult with Power BI Desktop. The transforms are not designed for incremental updates (while Power BI datasets can be incrementally updated, the same doesn't apply to the transforms). Third, all the data needs to be ingested through Power Query connectors, which makes for challenging work with large volumes of observational data.

For those reasons, for large analytical projects the data preparation phase is often performed outside the Power BI dataset, typically executed by professionals using a set of ETL tools on top of a relational data warehouse database or a big data lake.

The new Power BI dataflows capability is designed to overcome existing limitations of data prep in Power BI Desktop and allow large-scale, complex and reusable data preparation to be done directly within the Power BI service.

Business analysts will typically connect to the dataflows data from Power BI Desktop and use it to create BI datasets the same way they use SQL Server tables or other sources of data. Therefore, in terms of information hierarchy of the Power BI artifacts, dataflows are at bottom of the stack, preparing data to feed the datasets, which in turn are serving the various visualization artifacts of Power BI – dashboards and reports.

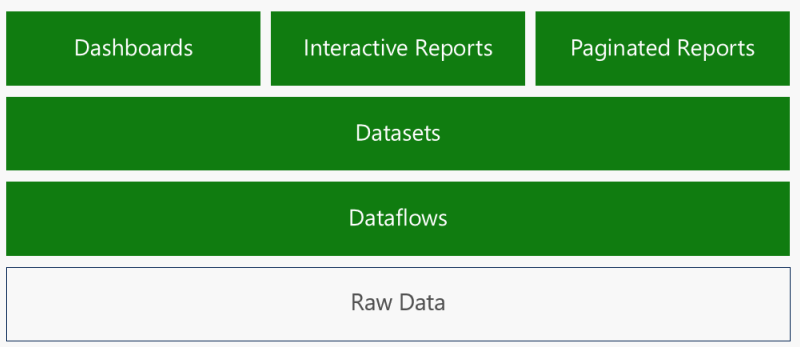

Figure 2: Power BI information hierarchy

"Dataflows are to ETL what Excel is to Programming"

The traditional technology for data prep is called ETL (**E**xtract, **T**ransform, **L**oad). ETL products have been available for decades and are the core toolkit of the data warehouse professional. ETL tools allow the professional to connect and extract data from many sources, perform a rich set of transformation on the data to cleanse, integrate and enrich it, and then load the data to a data warehouse or more recently, to a data lake.

Handling a one-time extraction of a table, performing some transformation, and storing the data somewhere may not seem terribly difficult. In fact, many Power BI and Excel users are going through that process when they use Power Query. But building a complete system that handles dozens of tables, from many sources, with complex cleansing, integration and enrichment logic, and keeping them all up to date in an incremental manner is much harder. This is where professional ETL tools and data warehousing experts come into play.

ETL Orchestration is Like Programming

Data warehousing experts create ETL packages that handle various extract and transformation tasks. They then create complex orchestration logic to ensure that the ETL packages are executed at the right time, in the right order, under the right logical conditions. That latter part – the orchestration – is where most of the challenges exist. While each piece of logic may be sound, executing the correct package at the wrong time could cause catastrophic consequences. For example, an ETL package may be responsible for clearing the content of a daily sales table in the database. If it executes before the

content of that table has completed successfully to update the full historical sales table, data will be lost!

A significant ETL project often includes hundreds and even thousands of ETL packages, each of which must be chained together in complex orchestration logic. Building this logic is very much the same as coding a program flow in any procedural programming language. As with any computer program, it requires highly trained professionals that can conceptualize and create a program flow, handling all the branches and exceptions, and is similarly rife with bugs that must be weeded out from the system over time.

Dataflows are for Self-Service ETL by Business Analysts

As with the self-service revolution in BI, enabling self-service ETL requires a fundamental rethinking of the basic conceptual model, and the underlying supporting technologies.

As such, dataflows in Power BI were designed around five core principles:

- **Intuitive and familiar authoring**: dataflows are created using Power Query, a powerful and friendly transformation tool already familiar to tens of millions of Excel and Power BI users.
- **Auto Orchestration**: dataflows handle, automatically, the full orchestration of the transformations without any direction from the author.
- **Big data in a data lake**: dataflows are designed to work with massive amounts of data, both transactional and observational, stored in the Azure Data Lake Storage (gen 2) and are designed to be accessed by Azure data services.
- **Common data model**: dataflows support the Common Data Model (CDM) – a set of standard business entities such as Account, Product, Lead and Opportunity. Dataflows enable easy mapping between any data in any shape and into CDM standard entities.
- **Complete native integration with the rest of the Power BI system:** dataflows are as native an artifact of Power BI as datasets, dashboards and reports. They share workspaces with the other artifacts, the ALM.

Dataflows are like Excel

While millions of IT professionals can code, hundreds of millions of business analysts can use Excel. Business analysts can create incredibly sophisticated Excel workbooks, sometimes encapsulating mission-critical logic responsible for billions of dollars in Wall Street transactions, all without writing a single line of code.

The secret of Excel is two-fold: first – a highly intuitive user experience where users work in a visual grid environment, getting instant feedback for every operation performed. Second and most importantly – the Excel calculation engine is where the real magic lies. The calculation engine takes away the need for users to orchestrate the computations. Instead, users simply author logical expression in cells and Excel automatically determines if, when, how, and in what order those formulas need to be executed.

Since Excel has been part of our lives for decades, we typically take the magic it performs for granted: whenever the value of a cell changes in the worksheet, Excel automatically updates all other cells, and does it correctly every time. Excel performs this magic by analyzing the formulas in each cell, figuring out which other cells the formula refers to, uses all that information to build a dependency graph of all the

cells in the workbook, and then uses that graph to orchestrate the calculations in exactly the correct order, without fail.

The automatic orchestration of the calculations based on the dependency graph is what allows non-programmers to build these sophisticated spreadsheets.

"Dataflows are like Excel" is not just a figure of speech. The dataflows system is intentionally designed to follow the Excel conceptual and computational approach. The Excel analogy of dataflows is very strong and is key to understanding the inner workings of the system.

Dataflow Entities have a Formula

Dataflows define a collection of entities. Entities look like tables and, for simplicity, you can think of them as such (in this paper we use "entities" and "tables" interchangeably).

Each entity in the dataflow has a formula associated with it. The formula defines how the entity gets computed. Conceptually, it is a direct analog to each Excel cell having a formula that defines how it is computed.

The formulas used for entities are much more sophisticated and more powerful than those used in Excel formulas. The formulas for entities can have operators that do the following:

- Connect to data sources and ingest tables of data
- Create new computed columns in the tables
- Join tables together
- Filter and aggregate the tables
- Union tables
- Pivot a table
- Train and apply AI models on the tables
- And much more!

As with Excel cells, each dataflow entity has exactly one formula associated with it. That formula defines how the table gets computed.

The incredibly powerful formula language used for defining the entities is the *M* language. Therefore, these table formulas are called *M expressions*.

The Magic of the M Expressions

Learning the new M expressions language may seem like a significant burden and high bar for most data analysts. While M is incredibly powerful, it is certainly more complex than the Excel formula language.

But if you are among the millions of active users of Power BI or Excel, then you are likely already authoring complex M expressions without typing a single character, and without knowing anything about the syntax of the language!

The magic happens in the Power Query user interface. Power Query is a friendly tool that allows business analysts to read data from a large range of data sources, and to transform the data in incredibly powerful and flexible ways. Power Query supports more than 80 built-in data sources, as well as a custom connector SDK with a rich ecosystem. The Power Query user interface offers dozens of ways to compute and transform the data directly using the Power Query ribbon and dialogs. Tens of millions of

users have used Power Query, creating billions of tables, all through its friendly user interface. What may not be obvious to all these users is that each operation in the Power Query UI gets automatically translated to an M language function.

In fact, you can examine the M expressions created by Power Query by selecting the **Advanced Editor** option in the **View** ribbon of Power Query. The M expression may seem a bit scary, but remember – you never had to type any of it! For you and most other users, the M expression is automatically generated by the Power Query user experience, and you never have to worry about writing it or even examining it. In fact most Power Query users will never need to master the syntax of M.

When authoring dataflows, you have all the magic of M and Power Query within the Power BI service, in a **Power Query Online** experience.

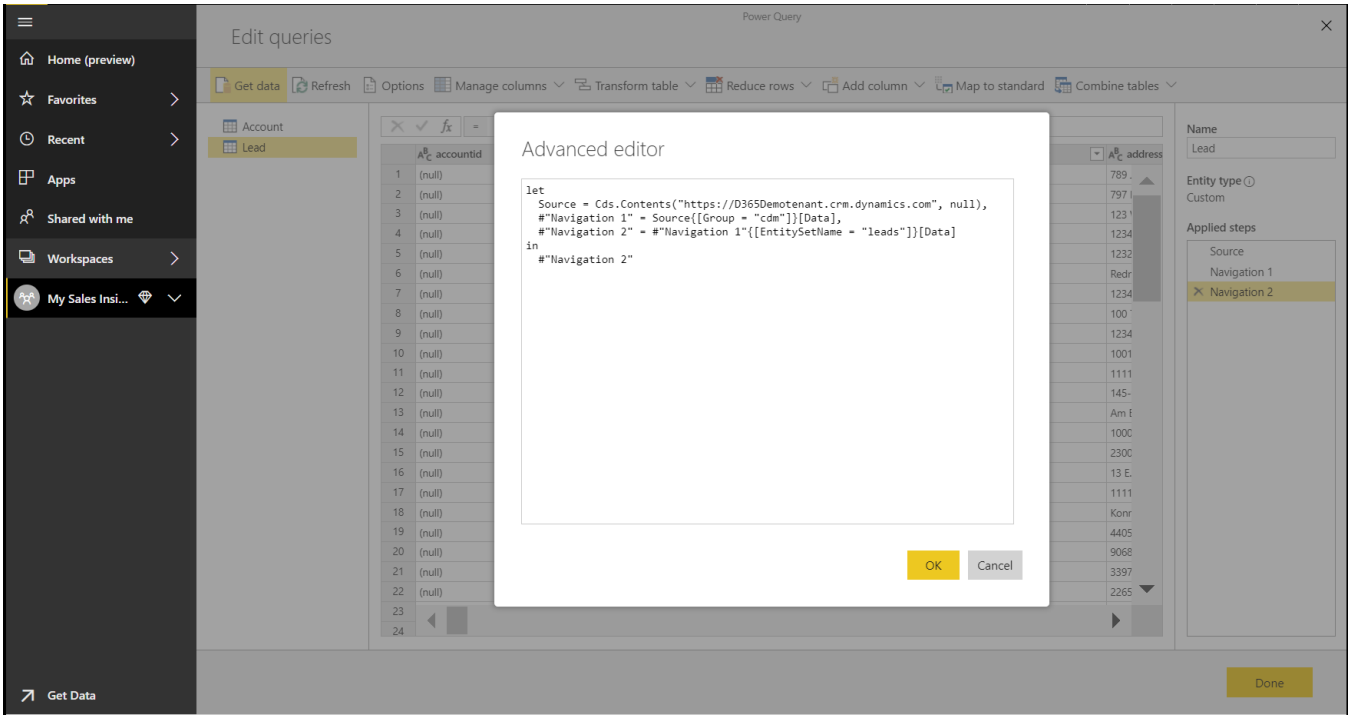

Figure 3: Power Query editor

The Dataflow Calculation Engine

We've established that every dataflow entity has a formula (an M expression) associated with it. In many cases, that M expression refers to other entities in the dataflow – for example, the M expression of an entity will be joining two other tables together or summarizing another table.

Just like the Excel calculation engine, dataflows have their own calculation engine. Like Excel, the dataflow calculation engine analyzes the M expression of each entity, finds the references to entities used in that expression, and uses the information to build a dependency graph between the entities.

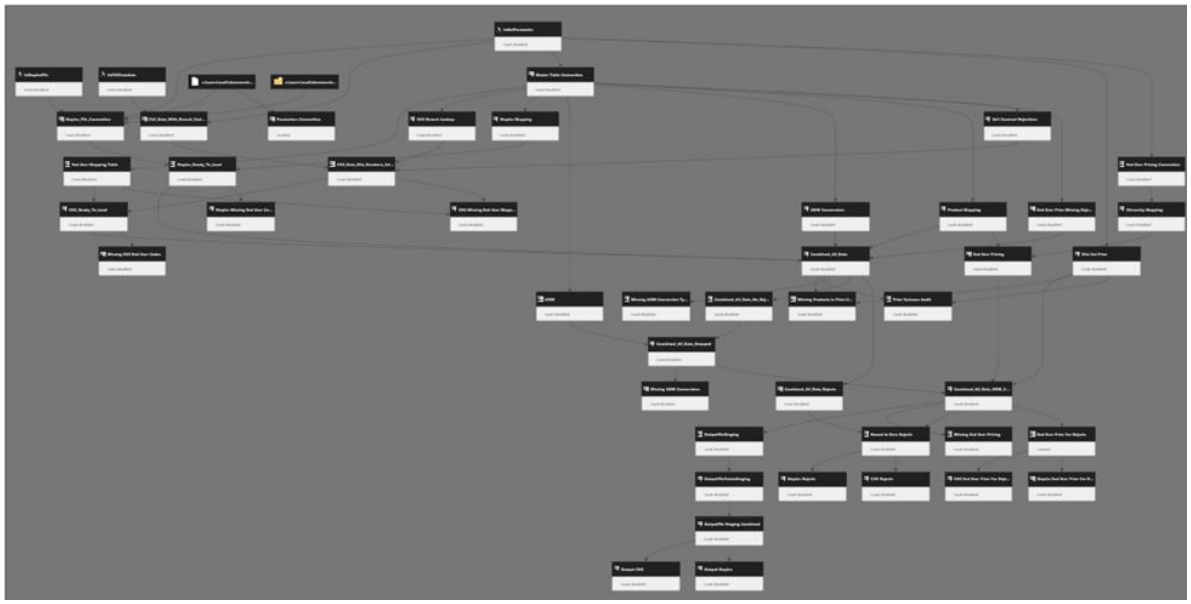

Figure 4: Dependency graph

Some of the entities in the dataflow only ingest data from outside data sources, while other entities are computed based on other tables defined within the dataflow. Using the dependency graph, the dataflow calculation engine determines which entities should be computed first, which are to follow, which set of computations can be done in parallel and which expressions should wait for other entities to complete their own computation. All this engine logic ensures the full correctness and consistency of the tables.

The dataflow calculation engine works pretty much the same as the Excel calculation engine. And just like Excel, it can react intelligently to any change in one of the tables. If any of the entities' data changes, the dataflow engine triggers the chain reaction of computations for any entities that depend on the entity data that changed.

The dataflow calculation engine relieves the business analyst from having to worry about orchestrating all the ingestions and transformations that must take place during the data preparation process. And just like with Excel, all the analyst must do is set the proper transformation logic for each table, and then sit back and let the calculation engine manage all orchestration of the computations.

Multiple Dataflows

For projects that ingest and transform dozens of entities or more, multiple dataflows are typically utilized. These multiple dataflows are created and managed in a single Power BI workspace.

Why would an analyst need to use multiple dataflows? For the same reason someone uses multiple Excel sheets in a workbook: it allows for better organization. In fact, the analogy of multiple dataflows in a workspace are like multiple sheets in an Excel workbook works quite well.

When encountering a large and complex data prep project, it's easier to organize the transformation process into natural table groups, or stages. For example, some dataflows will be dedicated to the data ingestion process, whereas other dataflows will deal only with cleansing and integration. Others still can deal with enrichments and AI scoring, and so forth.

There are no strict rules on the logic that should be used to group transforms. Project authors can organize the process into any set of dataflows that makes sense to them.

Multiple dataflows are great for convenient grouping and encapsulation of related logic. However, even when broken into multiple dataflow, they all continue work in concert as a single unit to generate the desired data. To connect the dataflows, entities in one dataflow can refer to entities computed in other dataflows, the same way that cells in one Excel sheet can refer to cells in another sheet.

The following diagram shows five dataflows working in concert to ingest, clean and enrich data to provide a final business view. Each dataflow ingests, transforms and creates a set of entities. The black circles with numbers depict the count of entities from the upstream dataflow that are used (referenced) in the downstream data flow.

Note that the diagram below is an early prototype for a tool Microsoft intends to add to Power BI several months after the public preview of dataflows. It is not available yet in the Power BI service.

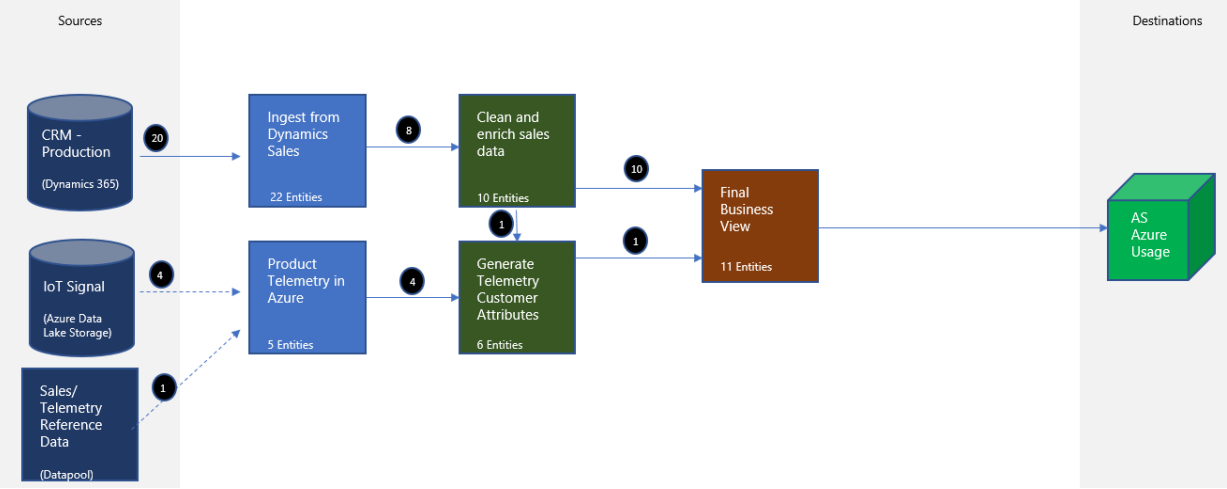

Figure 5: Conceptual Power BI visual dataflows editing tool

While dataflows allow for easy organization and encapsulation of data prep logic, the dataflow calculation engine analyzes all dataflows in a workspace together, generating a single complete dependency graph that spans all entities in all dataflows.

Therefore, whenever any data in any dataflow is updated, a recalculation process is initiated to update all downstream calculated tables that are affected by the change, spanning all dataflows in the workspace.

One more important reason for using multiple dataflows is the ability to set different refresh schedules for each. Some data sources may require a more frequent refresh than others. An author can break the dataflows based on the data sources they ingest, then set a separate schedule for each. In the example shown in the previous image, the dataflow *Ingest from Dynamics Sales* can be scheduled for 30-minute refresh intervals, while the dataflow *Product Telemetry from Azure* could be scheduled for twice a day.

Transaction Consistency

When all dataflows in a single workspace are performing their calculations, the dataflow engine ensures full transactional consistency. So while individual tables across the dataflow are computed at various

stages in the process, only once all computations complete successfully are the new values available for consumption. In the world of databases, the process of making all changes available together is known as *committing a transaction,* which ensures complete data consistency across all the tables so there is never a situation where some tables show new data and others still show old data.

The transactional system of dataflows is robust. If a failure occurs in the middle of processing any entity, the system picks up from the point it was interrupted and continues the computation from there to completion.

Complete Data Lake Consistency

Dataflows can refer to entities owned by dataflows managed in other workspaces, so the dependency graph managed by the engine actually spans all dataflows in all of the workspaces. Therefore, a single change to data in a single table in one dataflow could trigger a chain reaction of recalculation across a set of dataflows in many workspaces.

This ensures data consistency is automatically maintained across the entire data estate, without being cordoned off by boundaries of project ownership or workspace organization.

That said, the transaction consistency system is also designed to ensure the absolute robustness of each single workspace or application; the transaction commits at the boundary of the workspace. Only once the changes to the dataflows of a workspace commits do any other dataflows in other workspaces that depend on the changed data receive a change notification, and then initiate their own computation transaction.

In many ways, the ability to refer to data from other workspaces is akin to cells in an Excel workbook referring to data from another Excel workbook file: the recalculation transaction stays within the workbook, the same as the recalc of the dataflows is contained within the workspace.

This approach ensures that when the authors of dataflows in one workspace allow authors from other workspaces to reuse their data, they can rest assured that those other dataflows cannot hinder the performance or robustness of their own original dataflows. Therefore, even if the data is used in other workspaces, the computation of the originating workspace will complete in its own transaction, independent of any downstream workspaces.

Triggering the Recalculation of the Dataflows

In Excel, the most common recalculation occurs when user to types a new value to a cell, or changes a formula in one of the cells. In both these cases, manual editing triggers the recalculation.

Dataflows are not really designed for interactive cell editing. Instead, they are optimized for a regular, repetitive process of data updates originating from external sources. As such, the most common way for a dataflow recalculation to be triggered is for new data to be ingested.

There are three ways where new data can trigger a recalculation:

1. **Manual refresh of data**: An author can trigger data refresh by manually initiating a dataflow refresh.
2. **Scheduled refresh**: This is the most common way refresh and recompute data on a consistent basis. The scheduling facilities are identical to those for dataset refresh, including incremental

updates (more on that later in this paper). Each dataflow has its own refresh schedule, allowing great flexibility in handling data ingestion from sources with different data change velocity.

3. **Externally referenced data changed**: A dataflow can also refer to data that is not owned by the workspace. The external data can be entities in other workspaces, or data in so-called "CDM folders" in the data lake (more on that later in this paper). In either case, the dataflow that refers to the external data proactively looks for changes in the data. Whenever such changes are detected in the data lake, a recalculation is triggered automatically.

Incremental Updates

Authors can define incremental data ingestion for entities in the dataflow. The incremental update system is designed to be identical to incremental updates for Power BI datasets.

With incremental updates, dataflows can ingest only new data, or changed data, instead of re-ingesting all data from the source with each refresh.

Configuring incremental refresh is straightforward in the refresh options settings, but does require a bit of reading and some design and planning. The rewards for setting such a configuration are significant. Dataflows configured for incremental updates tend to complete their data refresh and computation in a fraction of the time, and a fraction of the resource consumption, compared to the default full-refresh method.

You can learn more about incremental refresh in the documentation article that describes incremental refresh with dataflows.

Dataflows and the Data Lake

Dataflows were designed from the ground up to work with massive amounts of data, both transactional and observational, stored in the ADLSg2 (Azure Data Lake Storage gen 2), and to be accessible by multiple Azure data services and applications.

Azure data lake provides Power BI dataflows with an enormously scalable storage facility for data. ADLSg2 can be thought of as an enormous disk, with incredible scale and inexpensive storage costs. ADLSg2 also supports the HDFS (Hadoop File System) standard, which enables the rich Hadoop ecosystem including Hive, Spark and MapReduce, to operate on it.

The Power BI Internal Data Storage vs "Bring Your Own Storage"

Many users of dataflows will never be aware of the underlying data lake. By default, Power BI uses ADLSg2 internally where it stores all data processed by dataflows created by the users. Every Premium node (P1 and above) gets 100TB of internal Power BI storage without any additional cost.

Using the internal data storage of Power BI is a simple option for many small organizations, but in many Enterprises, a Power BI administrator may choose to move the data from internal data storage of Power BI to a data lake that is owned by the organization.

When choosing the latter option, Power BI performs a one-time, one-directional move of all the data from the internal Power BI store to the tenant-provided ADLSg2 storage. From that point forward, all the dataflow processing and storage is performed on the organization's tenant-owned ADLSg2.

Open Approach

By moving the data to the tenant's storage account ("BYOSA – Bring Your Own Storage Account") the IT organization can access that data by using a host of Azure data services, which allows data engineers and scientists a rich toolset with which to process and enrich the data.

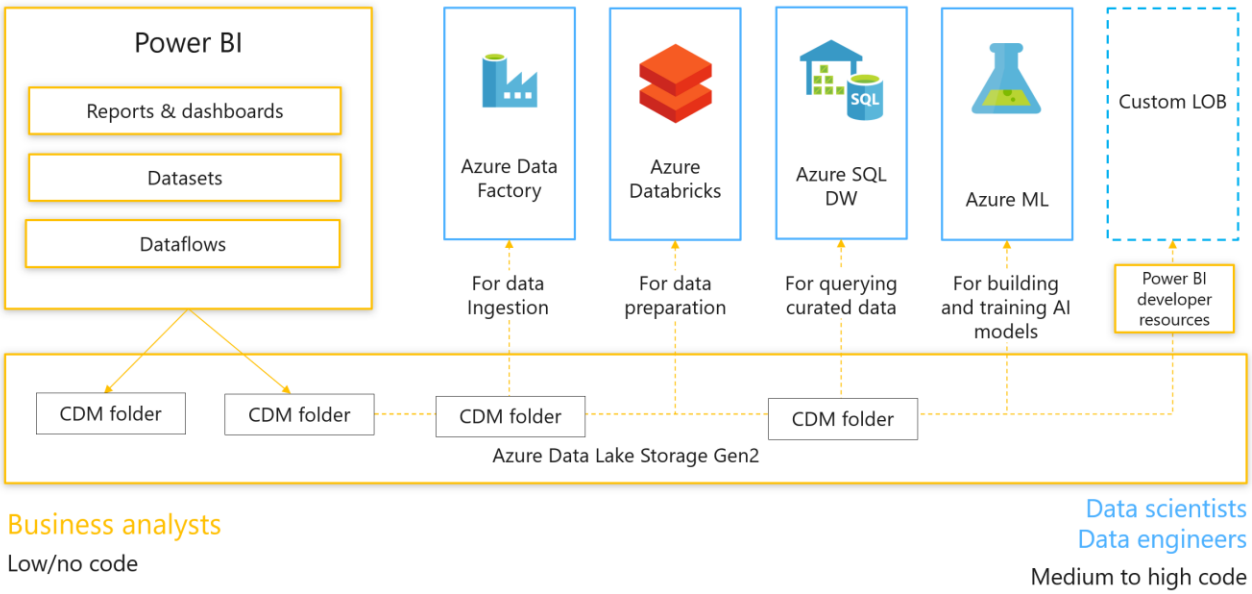

Figure 6: Open approach with Power BI

The ability to create a collection of data services and applications that are processing and exchanging data through ADLSg2 is a key feature of Power BI. It creates a strong and direct link between the business-driven data assets created by business analysts through the dataflows, with the strong technical capabilities of the IT pros who use advanced data services to provide depth and solve difficult challenges.

It also allows Power BI to easily work with large volumes of observational data that is directly deposited to the data lake by various IoT, logs and telemetry applications. Since Power BI sits directly above ADLSg2, this observational data can be mounted directly and used in dataflows without requiring any data movement.

To allow easy data exchange, Power BI stores the data in ADLSg2 in a standard format (CSV files) and with a rich metadata file describing the data structures and semantics in special folders called *CDM folders*. In a nutshell, the Common Data Model (CDM) unifies data in a well-known schema with semantic consistency across applications. You can learn more about CDM and CDM folders in the appendix.

Power BI dataflows are not the only capability that creates CDM folders. Any app and any service can create similar CDM folders. The appendix provides information about the structure and format of CDM folders.

Once a CDM folder is properly created by an application in ADLSg2, it's possible to do the following:

1. Mount that CDM folder directly to a Power BI workspace as an external dataflow
2. Use that CDM folder with any other Azure service that is compatible with dataflows, such as Databricks and ADF.

Security

All data processed by dataflows is kept secured in the Azure data lake. Since ADLSg2 is essentially a file system, Power BI uses standard ACLs (Access Control Lists) to restrict access to data files to authorized users only.

In the first release of Power BI dataflows, the access control applies to the entire dataflow (the whole CDM folder). In later releases, we expect access control to have entity-level granularity, allowing some users access to only some of the tables owned by a dataflow.

Row level security (RLS) is not currently supported by ADLSg2, so once a user is granted access to an entity, they have rights to read all the data in the table.

Because users are granted full access to the data in ADLSg2, it is highly recommended to restrict access to the dataflows' entities only to analysts whose job is to create Power BI datasets. RLS can then be implemented at the dataset level, and the rest of the users can be served from that tier.

Since the Power BI dataset has much richer metadata than dataflow entities, and since datasets also enjoy an ultra-fast query engine (not available directly on the data files in the lake), the recommended architecture is to always limit the use of dataflows to data preparation, to grant direct access to the data only to analysts who use the data to create datasets, and to serve the remaining user population only through the datasets and the reports and dashboards built on the datasets.

Dataflows: Power BI Pro vs Power BI Premium

Dataflows are available for both Power BI Pro and Power BI Premium offerings. There are differences in the capabilities of the two product offerings.

Power BI Pro allows analysts to ingest data from external sources. All Power BI connecters can be used for such access. Data ingested can be transformed upon ingestion, using standard Power Query transformations.

Power BI Premium provides access to the full power of dataflows; the dataflows recalculation engine is available only in Power BI Premium. There is no separate licensing required for dataflows, as it shares the same v-cores with the Analysis Services and Reporting Services engines. There is no need to dedicate specific v-cores to the dataflows calculation engine.

Dataflow Capability	Pro	Premium
Connectivity	All connectors to all sources	All connectors to all sources
Storage	10GB per user	100TB for P1 or greater nodes
Data ingestion	Serial ingestion of entities, making data refresh longer	Parallel ingestion of entities
Incremental updates	Not available	Available
References to entities in the same workspace	Not available	Available, allowing the creation of complex data prep processes using multiple dataflows
References to entities across workspaces	Not available	Available, allowing full data consistency across the whole data estate
Calculation engine	Not available, since entities cannot refer to other entities, computed entities cannot be created	Available, allowing computed entities for complex data prep projects with multiple cleansing and enrichment steps
Refresh rates	Up to 8 times a day	Up to 48 times a day

Table 1: Power BI dataflows capability (Pro & Premium)

Roadmap

This whitepaper is released with the public preview of the dataflow. Over the coming months we expect to see dataflows enhanced with regular weekly updates. Here is what you can expect:

Power Query for the Web Enhancements

The primary experience for defining entities in dataflows is through Power Query for the Web (PQW) that runs in the browser. In the initial release, PQW has a subset of the capabilities of Power Query that appear in Power BI Desktop.

The functional and experience gaps between the PQW and Power Query will be closed gradually over the coming months.

Similarly, connectivity in PQW to various sources is limited compared to what is available in the Power BI Desktop version of Power Query. We anticipate these gaps will also close in the coming months.

Once the gaps are closed, most new Power Query features used by dataflows will be released simultaneously for both Power BI Desktop and PQW.

Dataflows Experiences Enhancements

Beyond Power Query, Microsoft expects to release more refined and visual dataflow experiences. These experiences will allow users to get a better visual sense of the data prep process design and execution.

For example, Microsoft expects a visual dataflow editor that may look similar to the following image:

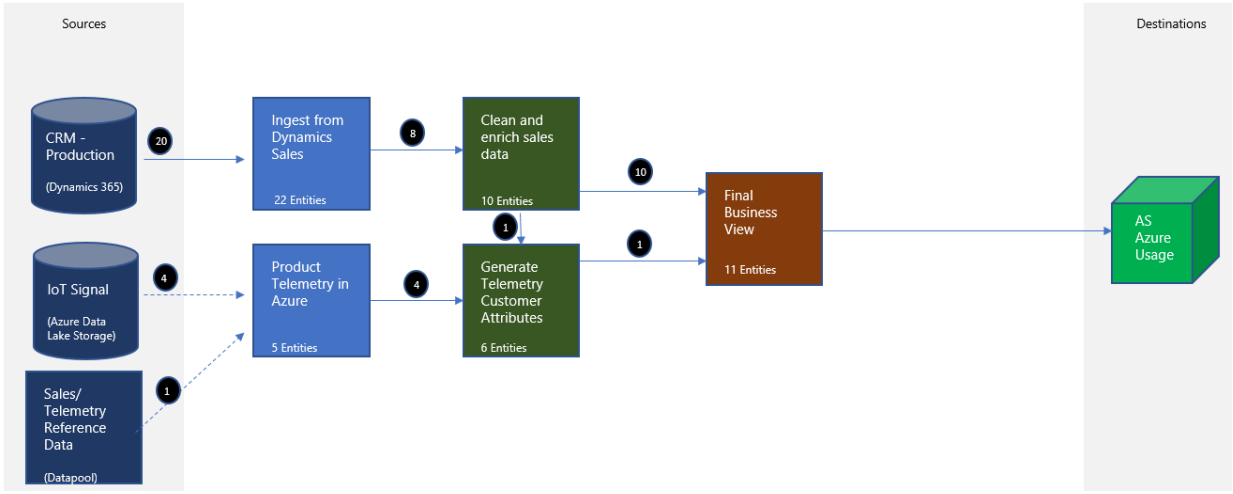

Figure 7: Conceptual Power BI visual dataflows editing tool

New tools for debugging and monitoring the progress of dataflow computations will also be available.

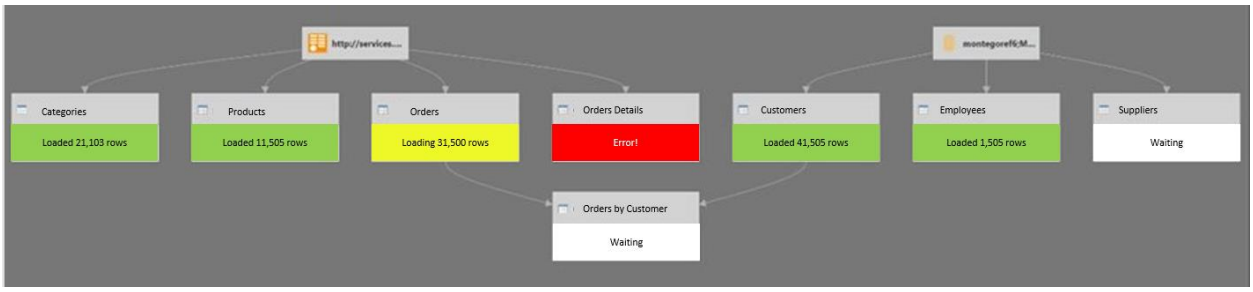

Figure 8: Conceptual debugging and monitoring tool for computations in Power BI dataflows

Calculation Engine Enhancements

You can expect to see gradual enhancements in the performance, scale and latency of the dataflows calculation engine.

In the initial release, the execution of dataflow transforms is performed by the M engine. Microsoft expect significant enhancements in this space to allow larger data to be processed, much faster and more efficiently.

Appendix: CDM Folders – the Storage Organization in the Azure Data Lake

About the Azure Data Lake

Azure Data Lake Storage (generation 2) is Microsoft's solution for *big data* storage – it's where organizations accumulate and amass any data that may have intelligence/analytical value. At the highest level, Azure Data Lake is an organized storage container (with both object and hierarchical access models). It enables best-in-class analytics performance along with Blob storage's tiering and data lifecycle management capabilities, plus the fundamental availability, security (including AAD integration and ACL support) and durability capabilities of Azure Storage.

About the Common Data Model

Microsoft has created the **Common Data Model (CDM)**, in partnership with an industry ecosystem that defines a mechanism for storing data in an Azure Data Lake that permits application-level interoperability.

More specifically, the goal is to permit multiple data producers and data consumers to **easily interoperate**, over raw storage of an Azure Data Lake Storage (ADLSg2) account (without necessarily requiring additional cloud services for indexing/cataloging contents) to accelerate value from data by minimizing time and complexity to insight and intelligence.

The CDM accomplishes this by bringing structural consistency and semantic meaning to the data stored in a CDM-compliant Azure Data Lake. The primary reason that CDM and CDM Standard Entities exist on the intelligence/analytics side is to bring semantic meaning to data (and structural consistency, but that is secondary and in support of the out-of-box semantic model). Note that this provides an expanded definition of CDM to cover structural aspects of the data lake, in addition to individual entity definitions.

CDM Folders and Standard Format

Standardizing contents in an ADLSg2 account with data in **CDM compliant form** enables Azure services and customers to build applications that *just work* when they open an ADLSg2 account – so they can read and understand the data inside, just from how bits are laid out in the storage account. A **CDM folder** is a folder in the data lake conforming to specific, well-defined and standardized metadata structures and self-describing data. This approach facilitates effortless metadata discovery and interoperation between data producers (such as Dynamics 365 business application suite) and data consumers, such as Power BI analytics, Azure data platform services (such as Azure Machine Learning, Azure Data Factory, Azure Databricks, so on) and turn-key SaaS applications (Dynamics 365 AI for Sales, so on).

- **CDM folder** – describes a folder structure and self-describing metadata in a file called *Model.json* further describing its contents.
  The *Model.json* metadata description file contains semantic information about entity records /

attributes, links to underlying data files, and indicates compliance with CDM Standard Entities which have additional rich out-of-box semantic metadata that applications can leverage.

- **Data files** – underlying data files in well-defined structure and format.

The diagram below shows an example ADLSg2 storage account populated with CDM folders with well-defined metadata descriptions and data files:

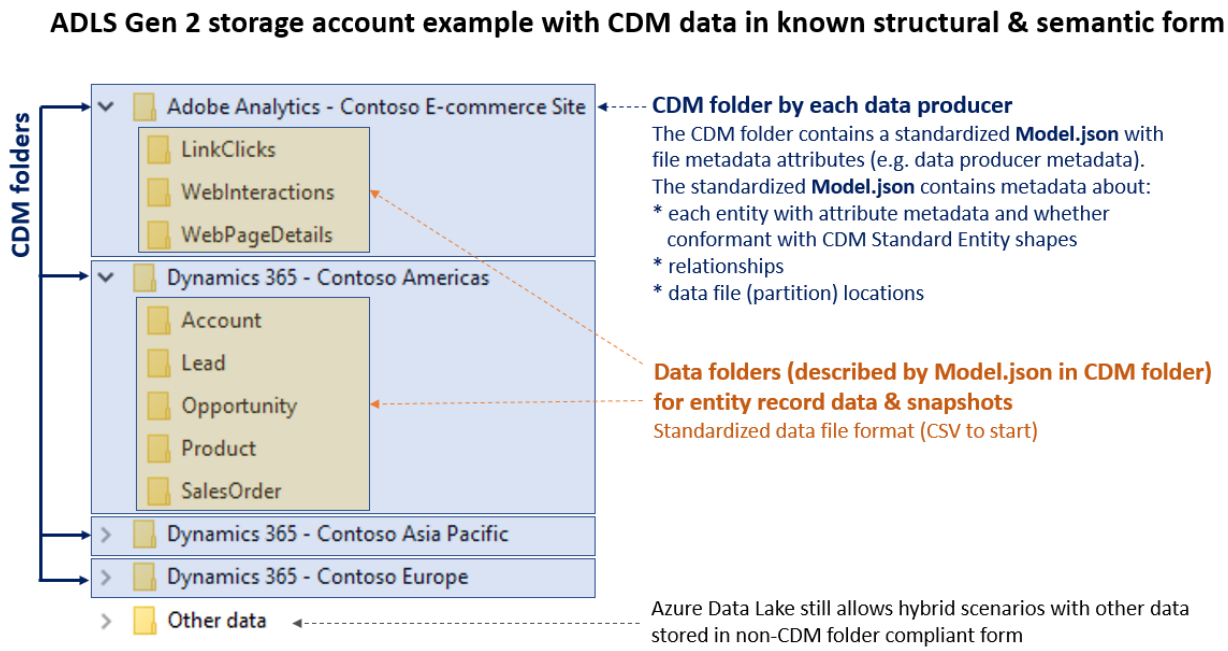

Figure 9: Example ADLSg2 storage account populated with CDM folders

The specific folder names (such as *Dynamics 365 – Contoso Americas*) shown in the diagram above are for illustration purposes to enhance readability. CDM folders (the blue boxes) and *Model.json* are mandatory, the internal subfolder structure (such as a folder per entity for data snapshots) is optional. Note that data consumer applications make decisions **based on metadata in each CDM folder** (for example, file metadata properties on *Model.json* inside a CDM folder, and *Model.json* metadata contents), **not** based on specific folder names at the root level of the storage account.

The concept of *CDM Folder* is shared across all participating services; individual services, such as Power BI or Azure Data Factory may put additional service-specific private metadata on top of the standardized metadata into a folder (for example, ETL pipeline metadata).

CDM Folder and entity metadata descriptions

Applications as data consumers make decisions based on application metadata in the metadata file in each CDM Folder. Each data producer typically owns its CDM Folder.

The CDM Folder contents is described by a standardized metadata JSON file, containing the following key categories of metadata:

- **Overall CDM Folder metadata:** applicable to all entities within the folder, such as description, last modified time, data culture.
- **Entities & attributes:** this includes attribute metadata, such as data types.
- Per entity, **compliance with CDM Standard Entity schema shapes**: For example, a given entity in the folder might be compliant with all the attributes required for a *base* Account entity (defined in the CDM Standard), as well as contain the additional attribute defined by the Dynamics 365 *Sales* application and listed in the CDM Standard as extended *Account* entity for Sales.
- **Relationships among entities**
- **Lineage**: Information on detailing where the data is originating from (for Power BI – the M expressions)
- Per entity, data file (partition) locations, and metadata about data encoding of data files

Based on this metadata, services and applications can gain deep understanding of the data programmatically without requiring user intervention. For example, Azure Databricks will be able to configure a whole set of dataframes in its notebooks automatically, without further user input, by simply pointing it to the CDM folder created by a Power BI dataflow.

Data files in CDM form

CDM bridges both transactional and analytical worlds. Many attributes of a CDM Standard Entity are declared optional, which means they can have null values, or have default values when not explicitly specified in the data file.

CDM folders describe the schema structure and semantic metadata of data files in a CDM-form compliant data lake. **CSV format** is the most ubiquitously supported format in Azure Data Lake and data lake tools in general, and CSV is generally the fastest and simplest to write for **data producers**.

Power BI dataflows store data in CDM folders as CSV files. CSV was chosen as the first data format supported in CDM compliant data lakes. In the future, the following additional data formats commonplace in the Hadoop ecosystem are planned to be directly supported:

- **Parquet** is a compressed binary columnar storage format, and generally the fastest format to read for data consumers because it is optimized for write-once-read-many scenarios.
- **Avro** stores the data definition in JSON format, making it easier to read and interpret, with the data itself stored in binary format making it compact and efficient.