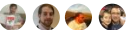


Enterprise BI in Azure with SQL Data Warehouse

11/06/2018 • 11 minutes to read • Contributors 

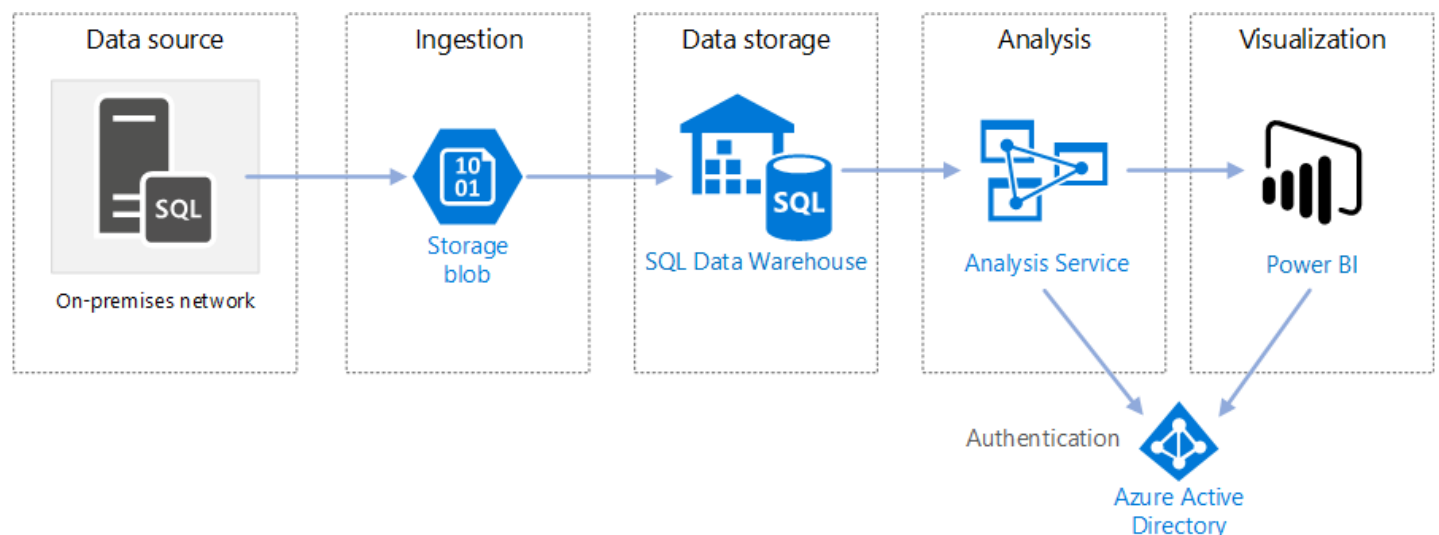
In this article

[Architecture](#)
[Data pipeline](#)
[Scalability considerations](#)
[Security considerations](#)
[Deploy the solution](#)
[Next steps](#)
[Related resources](#)

This reference architecture implements an [extract, load, and transform \(ELT\)](#) pipeline that moves data from an on-premises SQL Server database into SQL Data Warehouse and transforms the data for analysis.



A reference implementation for this architecture is available on [GitHub](#).



Scenario: An organization has a large OLTP data set stored in a SQL Server database on premises. The organization wants to use SQL Data Warehouse to perform analysis using Power BI.

This reference architecture is designed for one-time or on-demand jobs. If you need to move data on a continuing basis (hourly or daily), we recommend using Azure Data Factory to define an automated workflow. For a reference architecture that uses Data Factory, see [Automated enterprise BI with SQL Data Warehouse and Azure Data Factory](#).

Architecture

The architecture consists of the following components.

Data source

SQL Server. The source data is located in a SQL Server database on premises. To simulate the on-premises environment, the deployment scripts for this architecture provision a VM in Azure with SQL Server installed. The [Wide World Importers OLTP sample database](#) is used as the source data.

Ingestion and data storage

Blob Storage. Blob storage is used as a staging area to copy the data before loading it into SQL Data Warehouse.

Azure SQL Data Warehouse. [SQL Data Warehouse](#) is a distributed system designed to perform analytics on large data. It supports massive parallel processing (MPP), which makes it suitable for running high-performance analytics.

Analysis and reporting

Azure Analysis Services. [Analysis Services](#) is a fully managed service that provides data modeling capabilities. Use Analysis Services to create a semantic model that users can query. Analysis Services is especially useful in a BI dashboard scenario. In this architecture, Analysis Services reads data from the data warehouse to process the semantic model, and efficiently serves dashboard queries. It also supports elastic concurrency, by scaling out replicas for faster query processing.

Currently, Azure Analysis Services supports tabular models but not multidimensional models. Tabular models use relational modeling constructs (tables and columns), whereas multidimensional models use OLAP modeling constructs (cubes, dimensions, and measures). If you require multidimensional models, use SQL Server Analysis Services (SSAS). For more information, see [Comparing tabular and multidimensional solutions](#).

Power BI. Power BI is a suite of business analytics tools to analyze data for business insights. In this architecture, it queries the semantic model stored in Analysis Services.

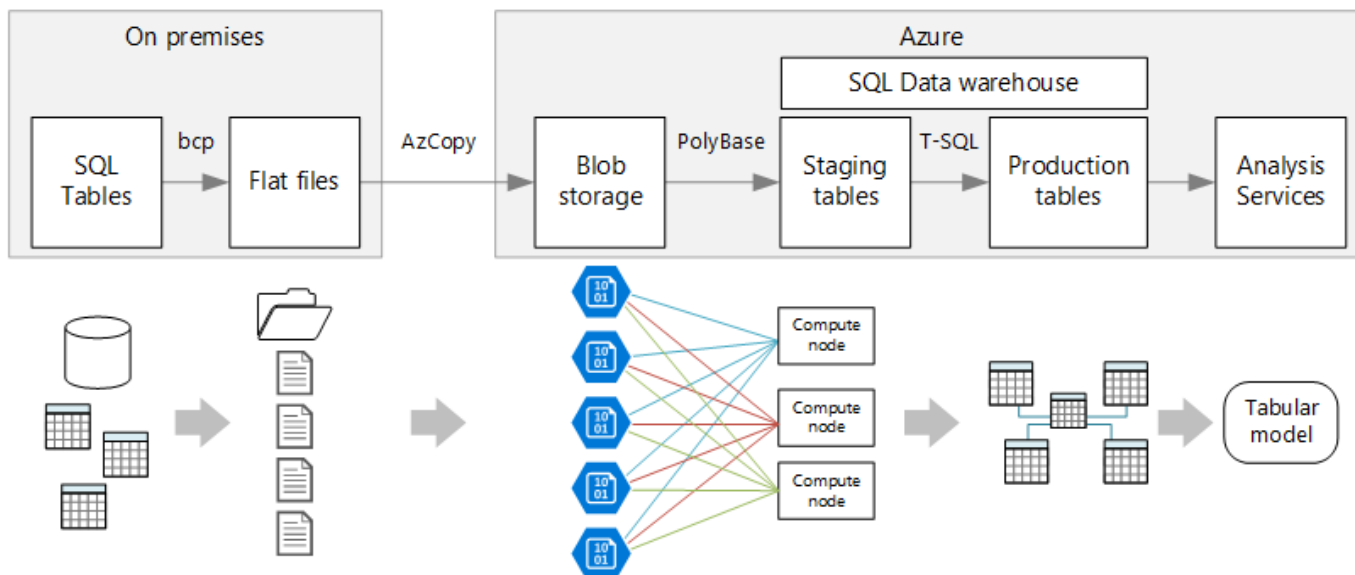
Authentication

Azure Active Directory (Azure AD) authenticates users who connect to the Analysis Services server through Power BI.

Data pipeline

This reference architecture uses the [WorldWideImporters](#) sample database as a data source. The data pipeline has the following stages:

1. Export the data from SQL Server to flat files (bcp utility).
2. Copy the flat files to Azure Blob Storage (AzCopy).
3. Load the data into SQL Data Warehouse (PolyBase).
4. Transform the data into a star schema (T-SQL).
5. Load a semantic model into Analysis Services (SQL Server Data Tools).



For steps 1 – 3, consider using Redgate Data Platform Studio. Data Platform Studio applies the most appropriate compatibility fixes and optimizations, so it's the quickest way to get started with SQL Data Warehouse. For more information, see [Load data with Redgate Data Platform Studio](#).

The next sections describe these stages in more detail.

Export data from SQL Server

The [bcp](#) (bulk copy program) utility is a fast way to create flat text files from SQL tables. In this step, you select the columns that you want to export, but don't transform the data. Any data transformations should happen in SQL Data Warehouse.

Recommendations:

If possible, schedule data extraction during off-peak hours, to minimize resource contention in the production environment.

Avoid running bcp on the database server. Instead, run it from another machine. Write the files to a local drive. Ensure that you have sufficient I/O resources to handle the concurrent writes. For best performance, export the files to dedicated fast storage drives.

You can speed up the network transfer by saving the exported data in Gzip compressed format. However, loading compressed files into the warehouse is slower than loading uncompressed files, so there is a tradeoff between faster network transfer versus faster loading. If you decide to use Gzip compression, don't create a single Gzip file. Instead, split the data into multiple compressed files.

Copy flat files into blob storage

The [AzCopy](#) utility is designed for high-performance copying of data into Azure blob storage.

Recommendations:

Create the storage account in a region near the location of the source data. Deploy the storage account and the SQL Data Warehouse instance in the same region.

Don't run AzCopy on the same machine that runs your production workloads, because the CPU and I/O consumption can interfere with the production workload.

Test the upload first to see what the upload speed is like. You can use the /NC option in AzCopy to specify the number of concurrent copy operations. Start with the default value, then experiment with this setting to tune the performance. In a low-bandwidth environment, too many concurrent operations can overwhelm the network connection and prevent the operations from completing successfully.

AzCopy moves data to storage over the public internet. If this isn't fast enough, consider setting up an [ExpressRoute](#) circuit. ExpressRoute is a service that routes your data through a dedicated private connection to Azure. Another option, if your network connection is too slow, is to physically ship the data on disk to an Azure datacenter. For more information, see [Transferring data to and from Azure](#).

During a copy operation, AzCopy creates a temporary journal file, which enables AzCopy to restart the operation if it gets interrupted (for example, due to a network error). Make sure there is enough disk space to store the journal files. You can use the /Z option to specify where the journal files are written.

Load data into SQL Data Warehouse

Use [PolyBase](#) to load the files from blob storage into the data warehouse. PolyBase is designed to leverage the MPP (Massively Parallel Processing) architecture of SQL Data Warehouse, which makes it the fastest way to load data into

Loading the data is a two-step process:

1. Create a set of external tables for the data. An external table is a table definition that points to data stored outside of the warehouse — in this case, the flat files in blob storage. This step does not move any data into the warehouse.
2. Create staging tables, and load the data into the staging tables. This step copies the data into the warehouse.

Recommendations:

Consider SQL Data Warehouse when you have large amounts of data (more than 1 TB) and are running an analytics workload that will benefit from parallelism. SQL Data Warehouse is not a good fit for OLTP workloads or smaller data sets (< 250GB). For data sets less than 250GB, consider Azure SQL Database or SQL Server. For more information, see [Data warehousing](#).

Create the staging tables as heap tables, which are not indexed. The queries that create the production tables will result in a full table scan, so there is no reason to index the staging tables.

PolyBase automatically takes advantage of parallelism in the warehouse. The load performance scales as you increase DWUs. For best performance, use a single load operation. There is no performance benefit to breaking the input data into chunks and running multiple concurrent loads.

PolyBase can read Gzip compressed files. However, only a single reader is used per compressed file, because uncompressing the file is a single-threaded operation. Therefore, avoid loading a single large compressed file. Instead, split the data into multiple compressed files, in order to take advantage of parallelism.

Be aware of the following limitations:

- PolyBase supports a maximum column size of `varchar(8000)`, `nvarchar(4000)`, or `varbinary(8000)`. If you have data that exceeds these limits, one option is to break the data up into chunks when you export it, and then reassemble the chunks after import.
- PolyBase uses a fixed row terminator of `\n` or newline. This can cause problems if newline characters appear in the source data.
- Your source data schema might contain data types that are not supported in SQL Data Warehouse.

To work around these limitations, you can create a stored procedure that performs the necessary conversions. Reference this stored procedure when you run `bcp`. Alternatively, [Redgate Data Platform Studio](#) automatically converts data types that aren't supported in SQL Data Warehouse.

For more information, see the following articles:

- [Best practices for loading data into Azure SQL Data Warehouse](#).
- [Migrate your schemas to SQL Data Warehouse](#)
- [Guidance for defining data types for tables in SQL Data Warehouse](#)

Transform the data

Transform the data and move it into production tables. In this step, the data is transformed into a star schema with dimension tables and fact tables, suitable for semantic modeling.

Create the production tables with clustered columnstore indexes, which offer the best overall query performance. Columnstore indexes are optimized for queries that scan many records. Columnstore indexes don't perform as well for singleton lookups (that is, looking up a single row). If you need to perform frequent singleton lookups, you can add a non-clustered index to a table. Singleton lookups can run significantly faster using a non-clustered index. However,

singleton lookups are typically less common in data warehouse scenarios than OLTP workloads. For more information, see [Indexing tables in SQL Data Warehouse](#).

ⓘ Note

Clustered columnstore tables do not support `varchar(max)`, `nvarchar(max)`, or `varbinary(max)` data types. In that case, consider a heap or clustered index. You might put those columns into a separate table.

Because the sample database is not very large, we created replicated tables with no partitions. For production workloads, using distributed tables is likely to improve query performance. See [Guidance for designing distributed tables in Azure SQL Data Warehouse](#). Our example scripts run the queries using a static [resource class](#).

Load the semantic model

Load the data into a tabular model in Azure Analysis Services. In this step, you create a semantic data model by using SQL Server Data Tools (SSDT). You can also create a model by importing it from a Power BI Desktop file. Because SQL Data Warehouse does not support foreign keys, you must add the relationships to the semantic model, so that you can join across tables.

Use Power BI to visualize the data

Power BI supports two options for connecting to Azure Analysis Services:

- Import. The data is imported into the Power BI model.
- Live Connection. Data is pulled directly from Analysis Services.

We recommend Live Connection because it doesn't require copying data into the Power BI model. Also, using DirectQuery ensures that results are always consistent with the latest source data. For more information, see [Connect with Power BI](#).

Recommendations:

Avoid running BI dashboard queries directly against the data warehouse. BI dashboards require very low response times, which direct queries against the warehouse may be unable to satisfy. Also, refreshing the dashboard will count against the number of concurrent queries, which could impact performance.

Azure Analysis Services is designed to handle the query requirements of a BI dashboard, so the recommended practice is to query Analysis Services from Power BI.

Scalability considerations

SQL Data Warehouse

With SQL Data Warehouse, you can scale out your compute resources on demand. The query engine optimizes queries for parallel processing based on the number of compute nodes, and moves data between nodes as necessary. For more information, see [Manage compute in Azure SQL Data Warehouse](#).

Analysis Services

For production workloads, we recommend the Standard Tier for Azure Analysis Services, because it supports partitioning and DirectQuery. Within a tier, the instance size determines the memory and processing power. Processing power is measured in Query Processing Units (QPUs). Monitor your QPU usage to select the appropriate size. For more information, see [Monitor server metrics](#).

Under high load, query performance can become degraded due to query concurrency. You can scale out Analysis Services by creating a pool of replicas to process queries, so that more queries can be performed concurrently. The work of processing the data model always happens on the primary server. By default, the primary server also handles queries. Optionally, you can designate the primary server to run processing exclusively, so that the query pool handles all queries. If you have high processing requirements, you should separate the processing from the query pool. If you have high query loads, and relatively light processing, you can include the primary server in the query pool. For more information, see [Azure Analysis Services scale-out](#).

To reduce the amount of unnecessary processing, consider using partitions to divide the tabular model into logical parts. Each partition can be processed separately. For more information, see [Partitions](#).

Security considerations

IP whitelisting of Analysis Services clients

Consider using the Analysis Services firewall feature to whitelist client IP addresses. If enabled, the firewall blocks all client connections other than those specified in the firewall rules. The default rules whitelist the Power BI service, but you can disable this rule if desired. For more information, see [Hardening Azure Analysis Services with the new firewall capability](#).

Authorization

Azure Analysis Services uses Azure Active Directory (Azure AD) to authenticate users who connect to an Analysis Services server. You can restrict what data a particular user is able to view, by creating roles and then assigning Azure AD users or groups to those roles. For each role, you can:

- Protect tables or individual columns.
- Protect individual rows based on filter expressions.

For more information, see [Manage database roles and users](#).

Deploy the solution

To the deploy and run the reference implementation, follow the steps in the [GitHub readme](#). It deploys the following:

- A Windows VM to simulate an on-premises database server. It includes SQL Server 2017 and related tools, along with Power BI Desktop.
- An Azure storage account that provides Blob storage to hold data exported from the SQL Server database.
- An Azure SQL Data Warehouse instance.
- An Azure Analysis Services instance.

Next steps

- Use Azure Data Factory to automate the ELT pipeline. See [Automated enterprise BI with SQL Data Warehouse and Azure Data Factory](#).

Related resources

You may want to review the following [Azure example scenarios](#) that demonstrate specific solutions using some of the same technologies:

- [Data warehousing and analytics for sales and marketing](#)
- [Hybrid ETL with existing on-premises SSIS and Azure Data Factory](#)

