

# sqibi®

[www.sqlbi.com](http://www.sqlbi.com)

Microsoft  
Partner



Gold Data Analytics  
Gold Data Platform

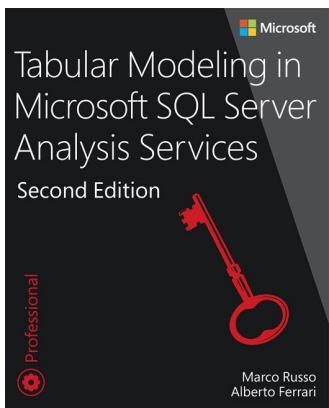
SSAS  
MAESTRO



MASTERING  
**TABULAR**

VIDEO COURSE

We write  
**Books**



We teach  
**Courses**



We provide  
**Consulting**

- Remote Consulting
- Power BI/SSAS Optimization
- BI Architectural Review
- On-Site Consulting
- Custom Training & Mentoring

We are recognized  
**BI Experts**



A 30,000 feet overview of Analysis Services Tabular

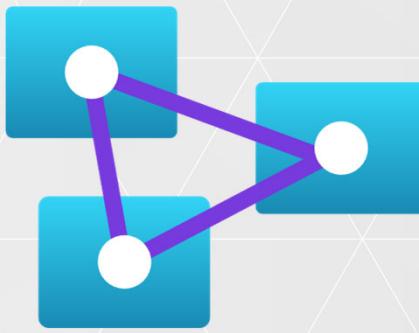
# **Analysis Services Tabular overview**



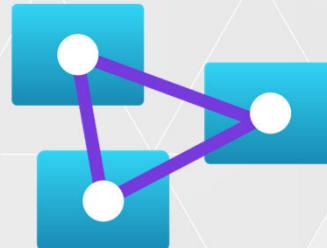
Tabular is the  
Power BI  
model



Tabular is the  
Analysis Services  
model



Enterprise-grade  
Power BI solutions  
require mastering  
Tabular concepts



# Evolution of Microsoft BI tools

Year	Server Version
2000	SQL Server Analysis Services Multidimensional (MDX)
2005	SSAS 2005 – UDM – MDX Script
2010	Power Pivot for Excel
2012	BISM – Analysis Services Tabular
2015	Power BI
Today	

# Analysis Services: where we started

On premises

AS Multidimensional

Development tools

Visual Studio

# Analysis Services: today

## Cloud

- Azure AS Tabular
- Power BI
- Power BI Premium
- Power BI Embedded
- Data gateway

## On premises

- AS Multidimensional
- AS Tabular
- Power Pivot for Excel
- Power BI Desktop

## Development tools

- Power BI Desktop
- Visual Studio
- Tabular Editor
- DAX Studio
- ALM Toolkit

## Data access

- VertiPaq
- DirectQuery over SQL
- DirectQuery over AS
- Cloud databases/services
- Power Query

## Different professionals, different needs

- Power BI author
- Business Intelligence architect
- Analysis Services Tabular developer
- Analysis Services Multidimensional developer
- Database administrator
- System administrator

## Goals of this course

- Create models in Power BI, Visual Studio, Tabular Editor
- Enrich semantic models to improve reports
- Understand how Tabular works
- How to make educated choices
- Complete features reference
- Deploy and maintain a solution

# What is not in the course

<b>DAX</b>	Introducing DAX Mastering DAX
<b>Data modeling</b>	Kimball star schema Data Modeling for Power BI
<b>Optimization techniques</b>	Optimizing DAX Optimizing Tabular
<b>Tools</b>	DAX Tools video course
<b>Design</b>	Dashboard design course

## How to get the most out of this course

- Download the sample files
- Study the lectures
- Repeat the relevant demos on your computer
- Practice on your data

# Content / Tabular overview

- Introduction to Tabular models
- Installating prerequisites
- Guided tour of Tabular
- Understanding model types

# Content / Tabular fundamentals

- Formula engine, storage engine
- Data sources for Tabular
- Tabular presentation layer
- Understanding relationships
- Handling security

## Content / Tabular in depth

- Understanding data types
- Understanding the VertiPaq engine
- Understanding DirectQuery
- Understanding DirectQuery over Analysis Services
- Aggregations
- Push datasets

# **Content / Tabular architecture**

- Hardware considerations
- Architectural choices

# Content / Managing Tabular

- Interfacing with Tabular
- Deploying Tabular models
- Processing and partitioning Tabular models
- Monitoring Tabular
- Backup and restore strategies
- Scaling out

# Customize your training

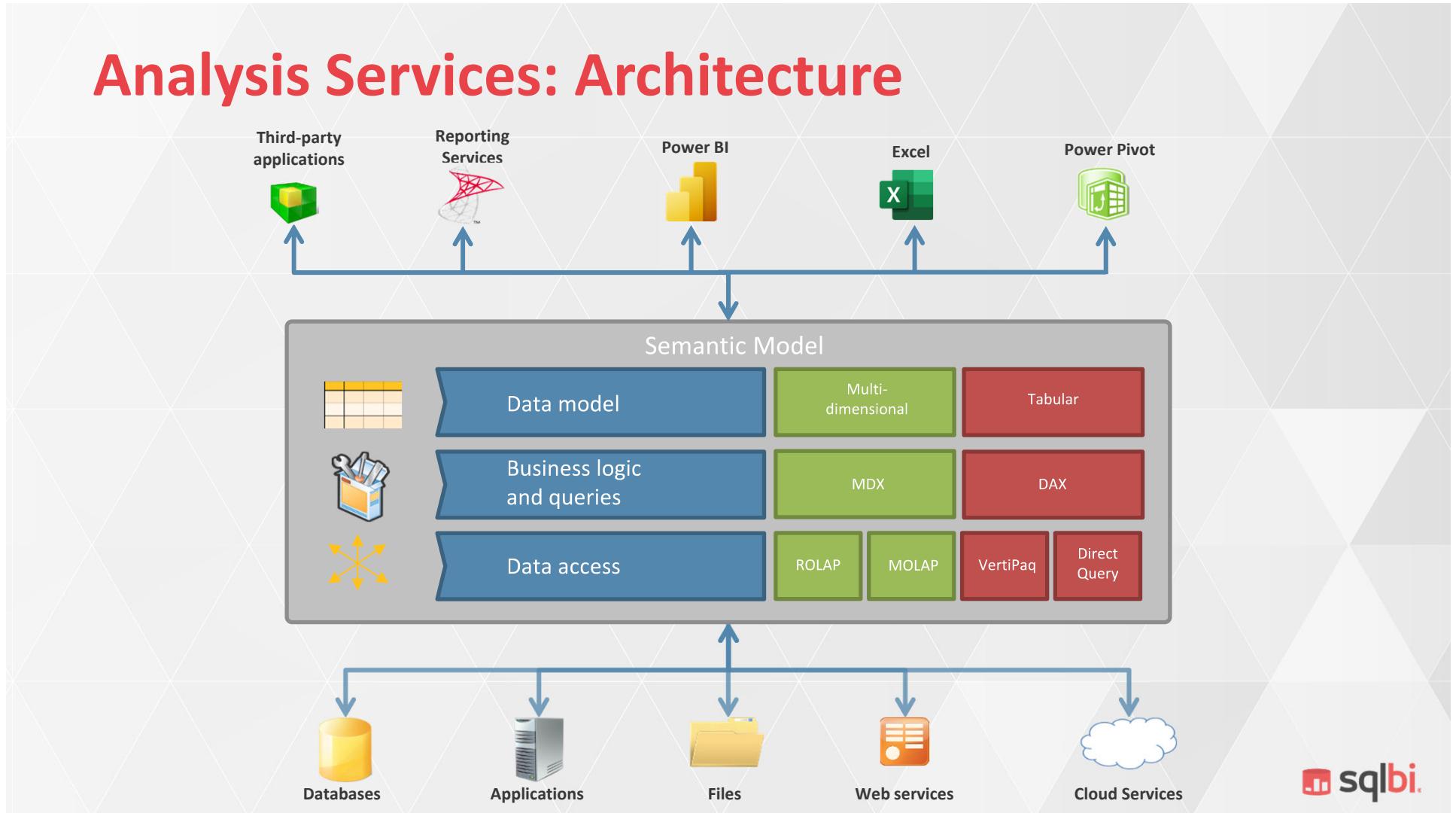
- The content is sequential
  - Watch all the modules, at least the intro
  - Check the relevance of each module for your solution
  - Different requirements and interests for different jobs
- Continuous learning
  - Use the links to deepen your knowledge on specific topics
  - Use the content as a reference while creating your model
  - Occasionally browse the modules you skipped

Overview of what a Tabular model is

# Introduction to Tabular models



# Analysis Services: Architecture



# Compatibility level

Compatibility level	Server version
1100	SQL Server 2017*, SQL Server 2016, SQL Server 2014, SQL Server 2012 SP1, SQL Server 2012
1103	SQL Server 2017*, SQL Server 2016, SQL Server 2014, SQL Server 2012 SP1
1200	Azure Analysis Services, SQL Server 2019, SQL Server 2017, SQL Server 2016
1400	Azure Analysis Services, SQL Server 2019, SQL Server 2017
1500	Power BI Premium, Azure Analysis Services, SQL Server 2019
15xx	Power BI Premium, Azure Analysis Services

# The protocol matrix

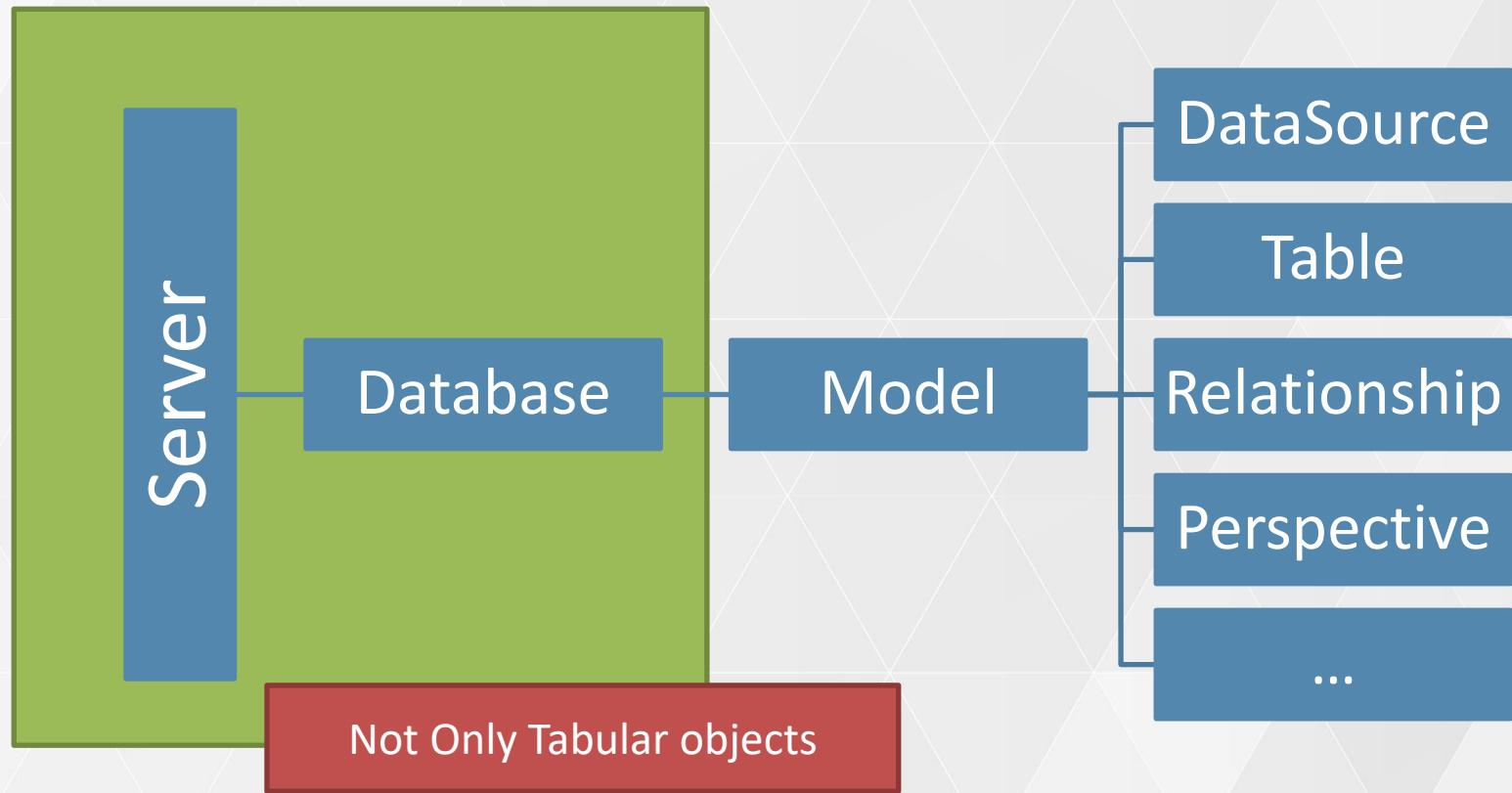
	Multidimensional	Tabular	Power BI
Communication protocol	MS-SSAS	MS-SSAS-T	MS-SSAS-T
Object definition	AMO	TOM	TOM
Scripting language	ASSL	TMSL	TMSL
Push Dataset API			Power BI API
Client libraries	ADOMD, MSOLAP	ADOMD, MSOLAP	ADOMD, MSOLAP

Here, the main focus is on TOM.  
TOM contains the description of a Tabular model.

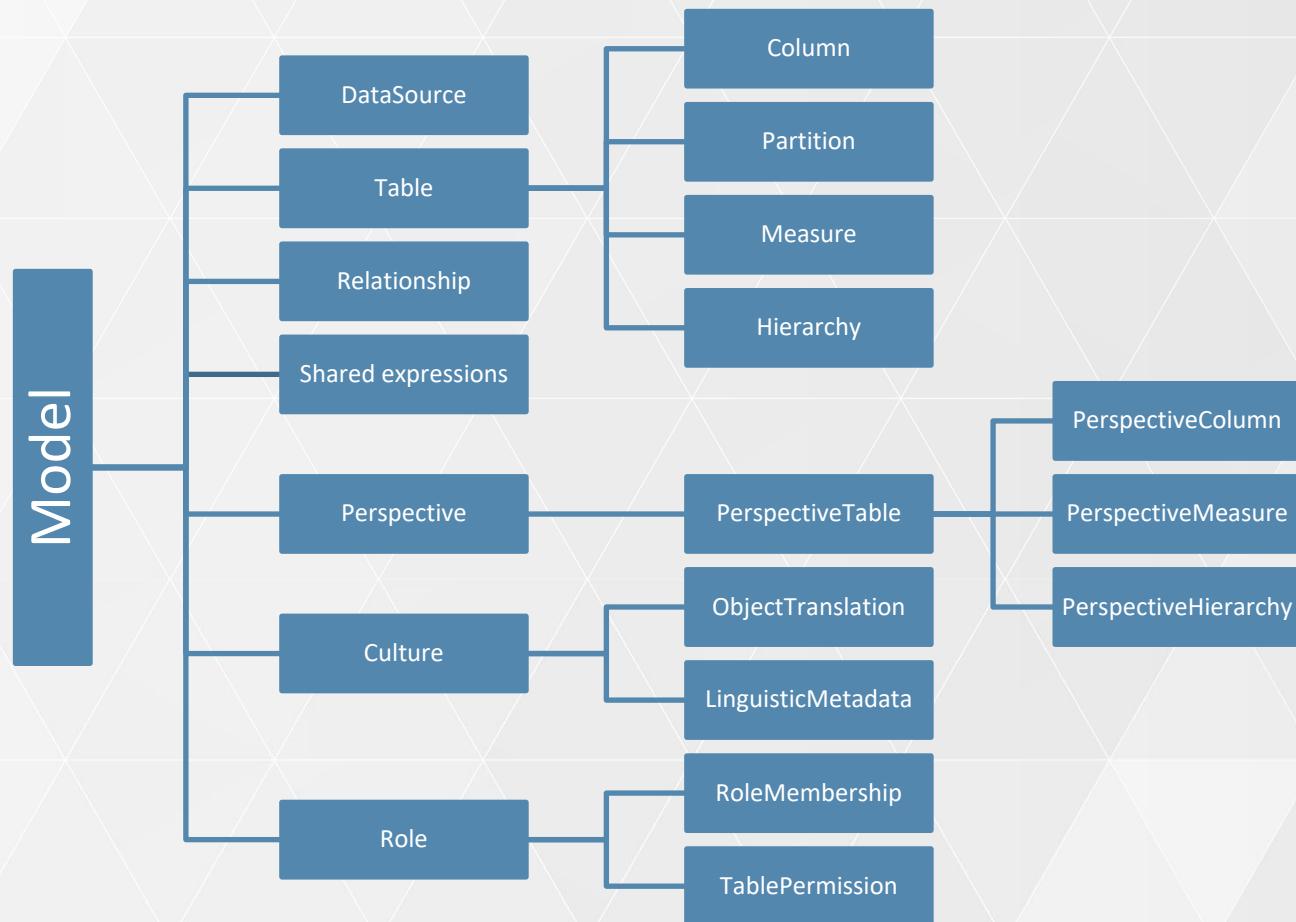
# Analysis Services protocols

- AS
  - MS SSAS Tabular Protocol (**MS-SSAS**)
    - XML for Analysis (**XMLA**)
  - ADOMD
- AS Multidimensional
  - Analysis Management Objects (**AMO**)
  - Analysis Services Scripting Language (**ASSL**)
- AS Tabular
  - MS SSAS Tabular Protocol (**MS-SSAS-T**)
  - Tabular Object Model (**TOM**)
  - Tabular Model Scripting Language (**TMSL**)
- Power BI
  - Power BI API
  - Designed to integrate Power BI features

# TOM Architecture



# Tabular model architecture



## Model collections

- **DataSources**: either load, connect, or both
- **Tables**: structure of the tables in the semantic model
- **Relationships** between tables in the model
- **Perspective**: different views of the same model
- **Culture**: translations of the model in different cultures, Q&A configuration
- **Role**: security information

# Tabular Object Model (TOM)

- Hierarchical structure of a Tabular model
- Serialized on a JSON document
- Each entity contains
  - Properties
  - Collections of children entities
  - Example: Table
    - Definition of the table, in its own properties
    - Collection of Column objects

# Model.BIM

Model.BIM is the serialization of the TOM definition of a Tabular model.  
It contains the whole model definition, in a simple text file.

```
{  
  "name": "SemanticModel",  
  "compatibilityLevel": 1500,  
  "model": {  
    "tables": [  
      {  
        "name": "Product",  
        "columns": [  
          {  
            "name": "ProductKey",  
            "dataType": "int64",  
            "sourceColumn": "ProductKey"  
          },  
          {  
            "name": "Product Code",  
            "dataType": "string",  
            "sourceColumn": "Product Code"  
          },  
        ]  
      }  
    ]  
  }  
}
```

## Tools to create a Tabular model

- Microsoft tools
  - Analysis Services Projects (Visual Studio)
  - Power BI Desktop
- Third-party tools
  - Tabular Editor
- Microsoft products provide the UI for specific tasks
- Tabular Editor exposes the full TOM hierarchy

# Architectural choices

- Where the data is stored
- Star schema / more normalized
- On-premises / cloud services
- Power Query / legacy
- VertiPaq / DirectQuery
- Using different data islands
- Composite models and aggregations
- Using calculation groups
- Calculated columns, measures, calculated tables
- Report measures / model measures
- How security is enforced
- Data latency (days, hours, minutes, real-time)
- User interaction response time required / expected
- Number of users and scalability

A short guide on installing the software needed for the training

# Install prerequisites



# Prerequisites

- Brief guide on which tools are needed for the training
- If you did not install some of the tools
  - Use the video to perform a step-by-step setup
- If you have already installed a tool
  - Use the video to verify that you followed all the steps required to follow the training
- No theory, these are just required tools, we explain the details later

# SQL Server 2019

- SQL Server 2019 – Developer edition
  - SQL Server (the relational engine)
  - SQL Server Analysis Services Tabular
- You need at least 2019 for this course:
  - Analysis Services features
  - Load the demo database

# **SQL Server Management Studio (SSMS)**

- Download and install latest version
- Keep SSMS updated to access cloud services

## Demo database

- We use one database for all the demos:
  - Contoso 100K (AS Tabular)
  - Inspired by Contoso sample by Microsoft
  - Additional data and views
- Steps
  - Create a demo alias for your server
  - Download the file and restore it
  - Grant read-access to the Analysis Services account
  - Test connectivity

# Visual Studio

- Visual Studio Community Edition
- Analysis Services Projects add-in for Visual Studio

# Power BI Desktop

- Microsoft Store version
  - New version every month, bugfix every 1-2 weeks
  - Automatic updates
- Power BI website version (manual download)
  - New version every month
  - Manual updates
- Report Server version (manual download)
  - New version every 4/6 months
  - Manual updates

## Excel

- Excel 2016 or later for PivotTables demos
- Not mandatory for the course
- Strongly suggested to test models for Excel users

## Tabular development tools

- DAX Studio
- Tabular Editor 2
- Tabular Editor 3
- Analyze in Excel
- ALM Toolkit

Build a model step by step, learning the basics of the most useful tools

# Guided tour of Tabular



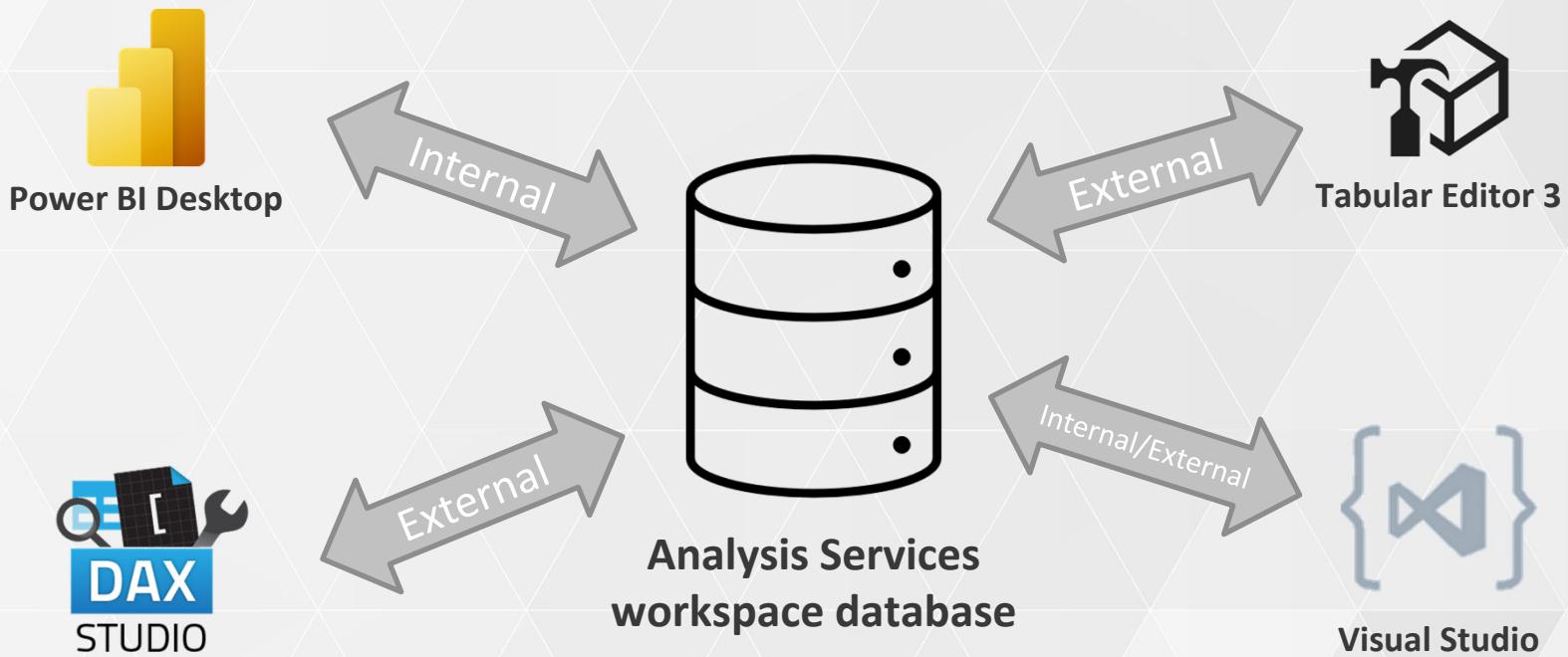
## Guided tour: the tools

- Different tools to edit a Tabular model:
  - Power BI Desktop
  - Analysis Services projects (Visual Studio)
    - Formerly known as SSDT
  - Tabular Editor 2
  - Tabular Editor 3
  - DAX Studio
- We are going to build a model with all the tools
- Introducing the basic concepts of Analysis Services

## The workspace database

- Users browse the deployed model
- Developers use a workspace database
  - Continuously edited during development
  - Always hosted by an instance of Analysis Services
- The workspace database can be
  - Internal: AS runs inside the development tool
  - External: the tool connects to an external server

# Tools connected to a workspace database



# Different tools, different workspaces

Tool	Internal workspace	External workspace	Online editing
Power BI Desktop	Yes	No	No
Visual Studio	Yes	Yes	No
Tabular Editor 2	No	No	Yes
Tabular Editor 3	No	Yes	Yes
DAX Studio	No	No	No

## Guided tour: the steps

- Load data from SQL
- Setup relationships in the model
- Create calculated columns and measures
- Create calculated tables
- Deploy the model on a server
- Build a simple report
- Run simple queries

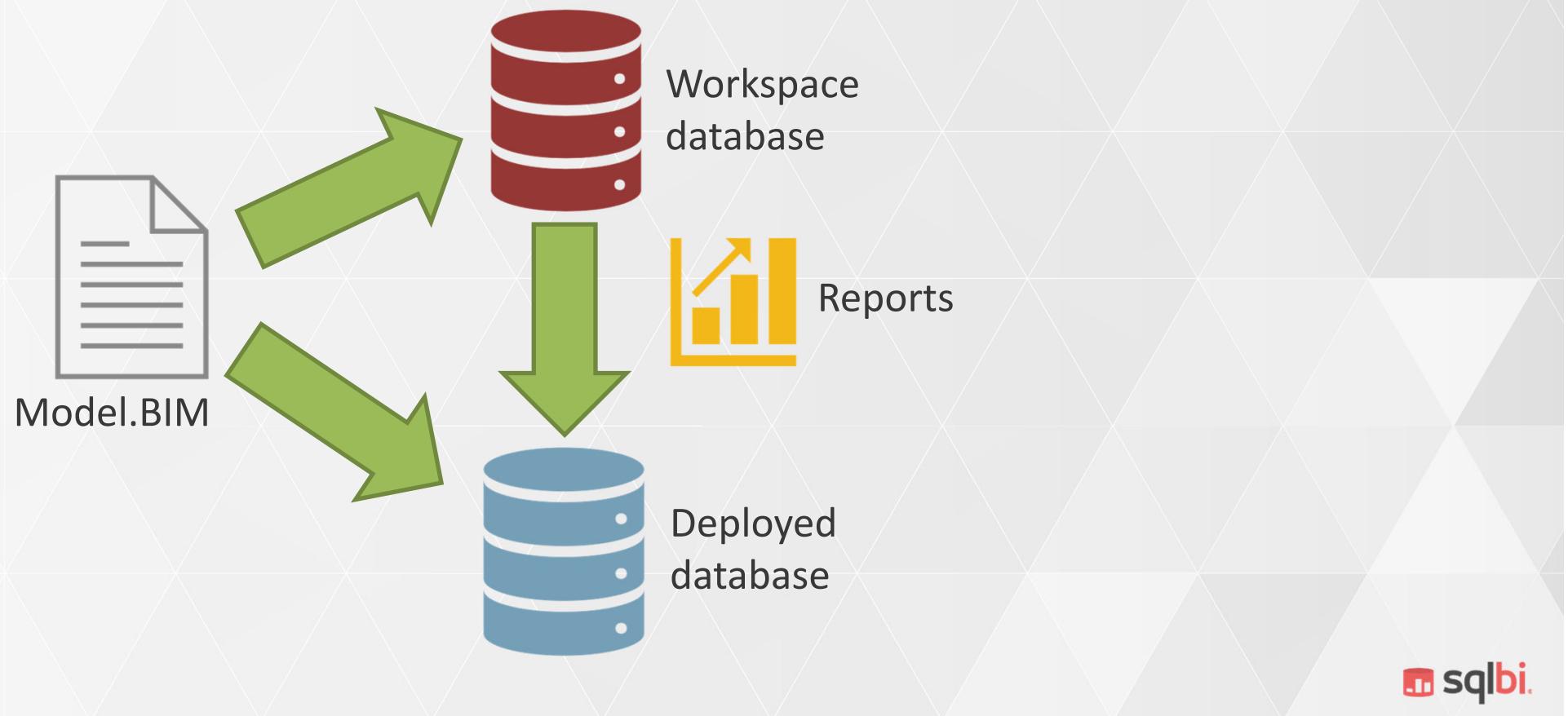
## Guided tour: how to watch the demos

- We are going to build a similar model using
  - Power BI Desktop
  - Visual Studio
  - Tabular Editor 2
  - Tabular Editor 3
- We strongly suggest you watch all the demos:
  - Even though you do not plan on using a specific tool
  - We explain concepts gradually through the four demos

# Guided tour with Power BI Desktop



# Development process in Power BI Desktop



# Introducing Power BI Desktop

- Power BI Desktop
  - Uses an internal workspace database
  - Runs a private instance of Analysis Services
- Power BI Desktop is at the same time:
  - A user interface to update the workspace database
  - A client tool to build reports using the workspace database
- A Power BI Desktop file contains
  - Model.BIM
  - JSON files with the structure of the reports
  - A complete backup of the Analysis Services database

# Calculated columns with Power BI Desktop

- Calculated columns
  - Columns added to the semantic model
  - Created by writing DAX code
  - Stored in the model
  - Computed during data refresh
- Commonly used to slice and filter with columns not already present in the data source
- Not the best option for most calculations; measures are a better choice

# Measures with Power BI Desktop

- Measures
  - Calculations added to the semantic model
  - Written using DAX
  - Computed at query time, not stored in the model
- Best option to add calculations to a semantic model

# Measures vs. calculated columns

- Measures
  - Computed at query time
  - Change their value based on the report filter
  - Do not consume memory in the model
- Calculated columns
  - Computed at process time
  - The value is unaware of any filters on the report
  - Use memory in the model

# Hierarchies with Power BI Desktop

- Hierarchies
  - Predefined navigation paths in the model
  - Commonly used to simplify data browsing
- Levels in a hierarchy can have a different name
- Strictly a visualization aid
  - Hierarchies do not improve the speed of calculations
  - Different behavior between DAX and MDX

## Connecting to the workspace database

- You can connect other tools to the workspace database
  - Useful to test, debug and analyze performance
  - Need to discover the TCP/IP port AS is listening to
- Power BI Desktop offers external tools
  - DAX Studio
  - Analyze in Excel
  - Tabular Editor
  - And many others

# Calculated tables in Power BI Desktop

- Calculated tables
  - New tables added to the model
  - Code is written in DAX
  - Stored in the model
  - Refreshed at process time
- Powerful modeling tool
  - Beware of RAM usage during processing
  - Might introduce circular dependencies
  - Mostly used on smaller models, or for prototypes

# Publishing to the Power BI service

- Power BI Desktop
  - Publish a model to the Power BI service
  - The entire model is uploaded to the cloud
  - Data refresh might or might not work, depending on the source
- Once the model is in the service
  - It can be browsed from the web
  - It can be periodically refreshed, if configured properly

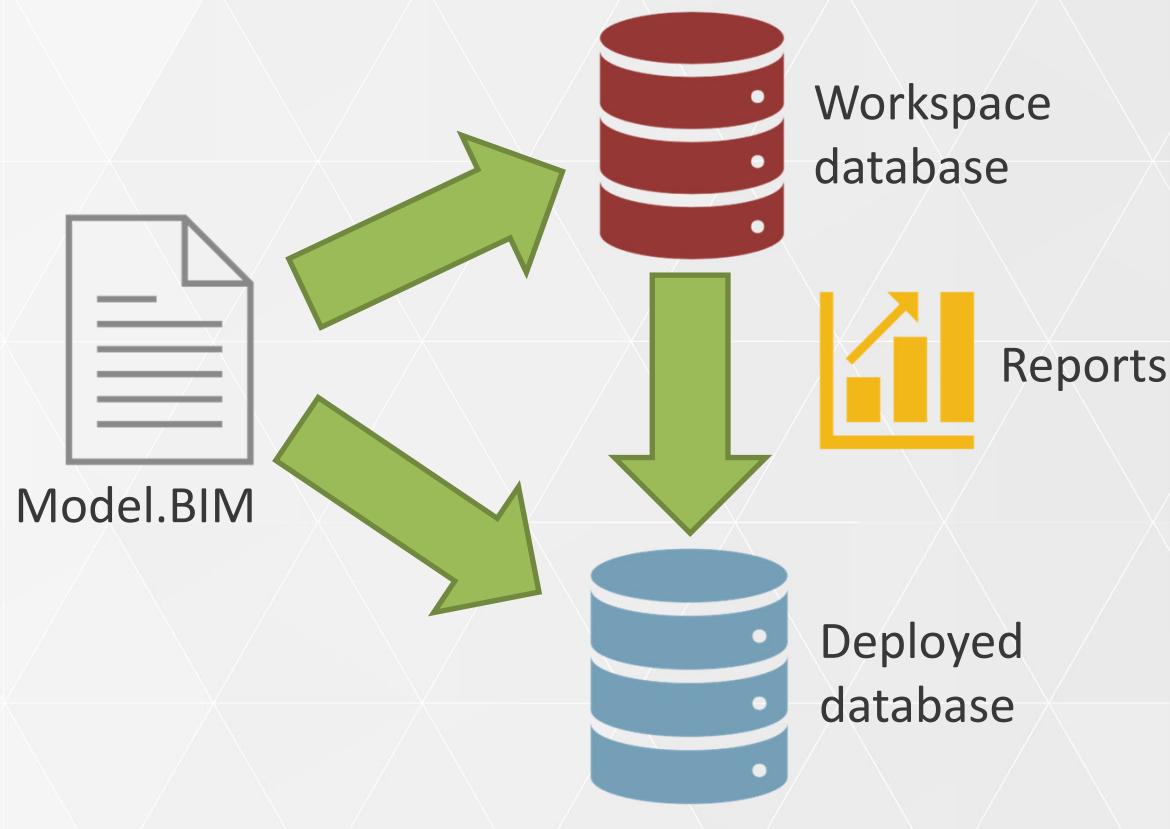
## Power BI Desktop: conclusions

- Power BI Desktop is a user interface for Analysis Services
- AS runs the workspace database
- Calculated columns, measures, calculated tables
- You can connect different tools to the workspace database
- A Power BI Desktop file can be published to Power BI Service
- A Power BI Desktop file cannot be deployed directly to Analysis Services

# Guided tour with Visual Studio



# Development process in Visual Studio



# Introducing Visual Studio

- Visual Studio is a developer tool
- Works with internal and external workspace databases
- Does not offer you a client tool. Excel or Power BI Desktop required to browse the semantic model
- Requires the Analysis Services Projects add-in
- More technical than Power BI Desktop, still not as advanced as Tabular Editor

# Choosing the workspace database

- Visual Studio can use
  - An integrated workspace, different versions of AS
  - An external workspace, requires an external AS server
- Choose the compatibility level
  - Depending on the target AS database of the solution

## Loading data: data sources and tables

- Data sources are the interface of AS with data
- A data source is required to connect to any kind of data
- There are two types of data sources
  - Structured data sources (Power Query)
  - Legacy data sources (supported databases only)
- Visual Studio edits the workspace database
- Data is loaded in the workspace database

# Introducing the Visual Studio user interface

- Data view, diagram view, source code
- Tabular model explorer
- Measure grid
- The properties panel
- DAX Editor
- Each edit in Visual Studio is reflected on the workspace server

## Building test reports

- Visual Studio does not work as a client tool
- External tools required:
  - Excel
  - DAX Studio
  - Power BI Desktop
- Excel connection is integrated in Visual Studio
- Other tools require knowledge of the TCP/IP port of AS

## Deploying the solution

- Once the project is completed, you can deploy on AS
- Deployment starts from model.bim
  - No data is transferred
  - The deployed model needs to be processed
- The deployed model can be further
  - Managed using SQL Server Management Studio
  - Browsed with any client tool (Excel, Power BI Desktop)
  - Processed periodically to refresh its content

## Using an external workspace database

- Visual Studio can use an external workspace database
- Data is stored in the workspace database
  - Need to decide what to do when you close the solution
- The workspace server performs the processing
  - Impersonation and authentication need to be set correctly

## Importing a deployed database

- Visual Studio lets you import model.bim from a database
- Useful in case the database contains changes not present in the source project
- Model.bim is recreated from the deployed solution
- Data is not imported: you must process the workspace database

## Visual Studio: conclusions

- Advanced developer tool for Tabular models
- Uses integrated or external workspace databases
- Does not provide client tool features
- Exposes many TOM features, not all of them

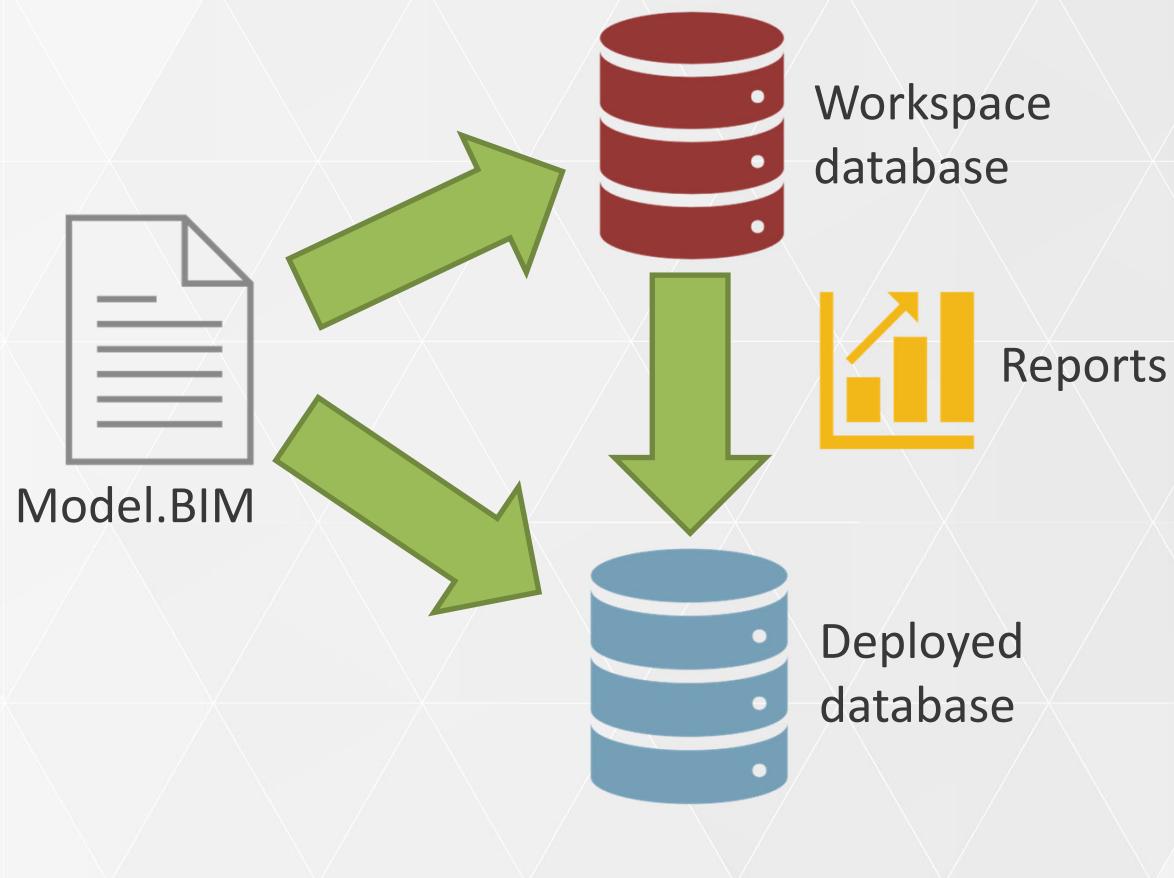
# Guided tour with Tabular Editor



# Introducing Tabular Editor

- There are two versions of Tabular Editor:
  - Tabular Editor 2: open source
  - Tabular Editor 3: paid version, premium features
- We introduce both versions here, we will mainly use Tabular Editor 3 during the training
- Tabular editor can work:
  - Connected to an external workspace database
  - Offline, editing the model.BIM file directly
- Edits are not committed immediately to the workspace

# Development process in Tabular Editor



## Creating a model with Tabular Editor 2

- Tabular Editor supports legacy data sources
- Tabular Editor 2 does not use a workspace database
  - You can edit model.bim
  - You need a server to deploy the solution
  - You can edit a deployed solution directly
  - Tabular Editor 3 supports working with an external workspace
- Tabular Editor 2 does not work as a client tool
  - Tabular Editor 3 offers you querying features too

## Different ways of working with Tabular Editor

- Disconnected mode uses Model.BIM
  - The one used by Visual Studio, for example
  - Or any model file previously saved
  - Visual Studio does not recognize concurrent edits
    - Visual Studio needs to be closed during editing
    - It might be necessary to delete the workspace database
- Connected to a database
  - Opens a deployed database directly
  - Any edit is saved on the database

## Advantages of disconnected edits

- Copy & paste of entities in TOM
- Full undo / redo of edits
- Offline editing provides a smoother experience
- You need multiple tools open at once
  - Tabular Editor 2
  - SQL Server Management Studio
  - DAX Studio
  - Excel

# Introducing Tabular Editor 3

- Improving on Tabular Editor 2
- All the features of Tabular Editor 2, plus:
  - Connection to an external workspace database
  - Very powerful DAX editor, with DAX scripting
  - Advanced user interface
  - Ability to work as a client tool
  - Diagram window
- Many other advanced features available, we discover them during the training

## Editing a model with Tabular Editor 3

- Tabular Editor 3 can work with a workspace database
  - It does not have an integrated workspace server
  - It can control an external workspace server
  - It does not delete the workspace database
  - It lets you preview the data in the workspace database
- We continue editing the model previously created with Tabular Editor 2

## Tabular Editor 3 as a client tool

- Tabular Editor 3 integrates querying features
  - DAX Queries
  - Pivot grids
- Useful to test and debug your measures on the fly
- Requires a workspace database

# DAX Editor and DAX Scripting

- The DAX editor in Tabular Editor 3 is very powerful
  - The DAX editor understands the code
  - Intelligent syntax highlighting
  - Errors are spotted immediately
  - Context-aware auto-complete
- DAX Scripting
  - Edit multiple measures in a single window
  - Apply edits to all the selected measures

## Using the diagram window

- Tabular Editor 3 includes a diagram view
- Diagrams are saved as external files
  - Diagrams are not part of TOM
  - No integration with Visual Studio or Power BI Desktop diagrams
- Useful feature in a multi-monitor environment

## Power BI Desktop integration

- Tabular Editor can be an external tool in Power BI Desktop
  - Both Tabular Editor 2 and Tabular Editor 3
- Lets you edit the entire model.bim architecture on a Power BI Desktop file
- Power BI Desktop acts as the workspace database

## Tabular Editor: conclusions

- Both are powerful offline/online editors
- Tabular Editor 3 is the most advanced editing tool available
- Tabular Editor exposes the full TOM hierarchy

## Guided tour: conclusions

- Developing Tabular models requires different tools
- All the tools can be used at the same time
  - Power BI Desktop
  - SQL Server Management Studio
  - DAX Studio
  - Tabular Editor 2
  - Tabular Editor 3
- Use the right tool for the job

The underlying storage technology affects the semantic model

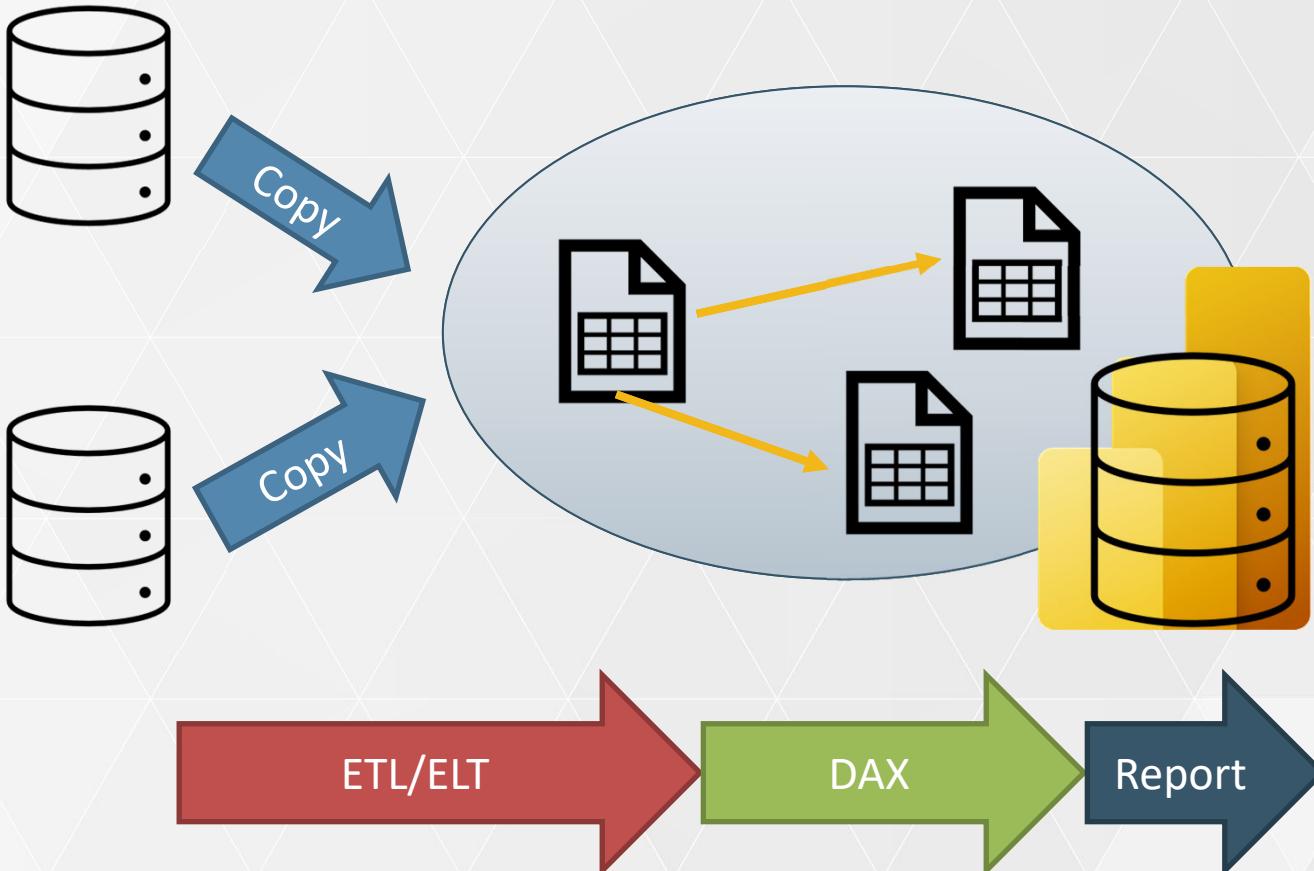
# Understanding model types



## AS storage technologies

- VertiPaq
  - Data is imported in the VertiPaq engine
- DirectQuery for relational sources
  - Data resides in relational data sources
  - Use SQL as a query language
- DirectQuery for Analysis Services
  - Data resides on another AS instance
- Mixing the architectures leads to several possible scenarios

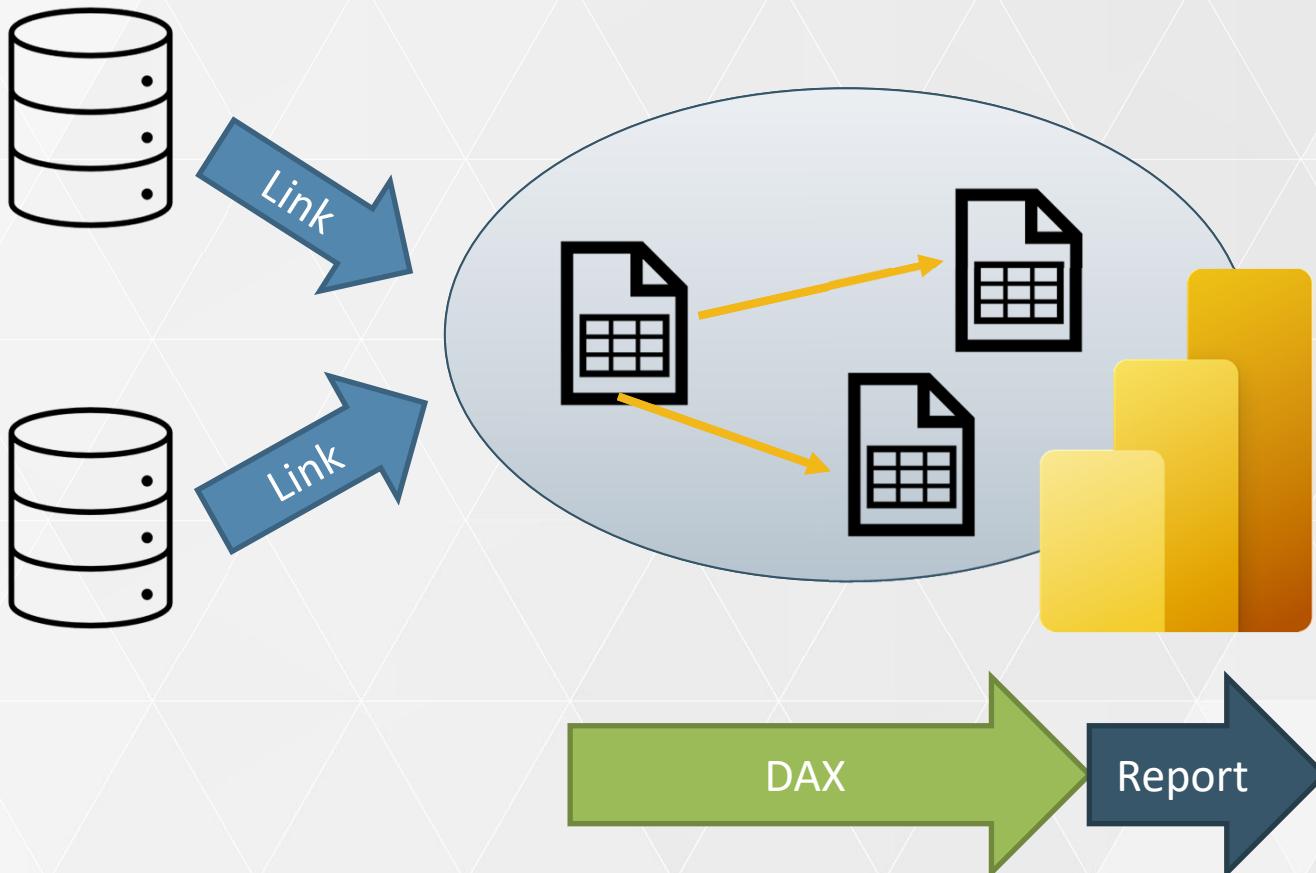
## VertiPaq model



Data is loaded from the sources into a single VertiPaq model.

The VertiPaq model contains the business logic (measures, relationships, entities).

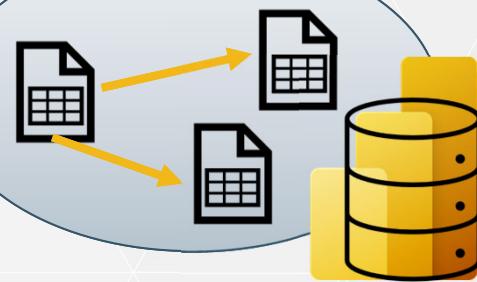
## DirectQuery model



Data remains in the data source, and it is linked in the model.

The AS DirectQuery model contains the business logic (measures, relationships, entities).

## Live connection to AS

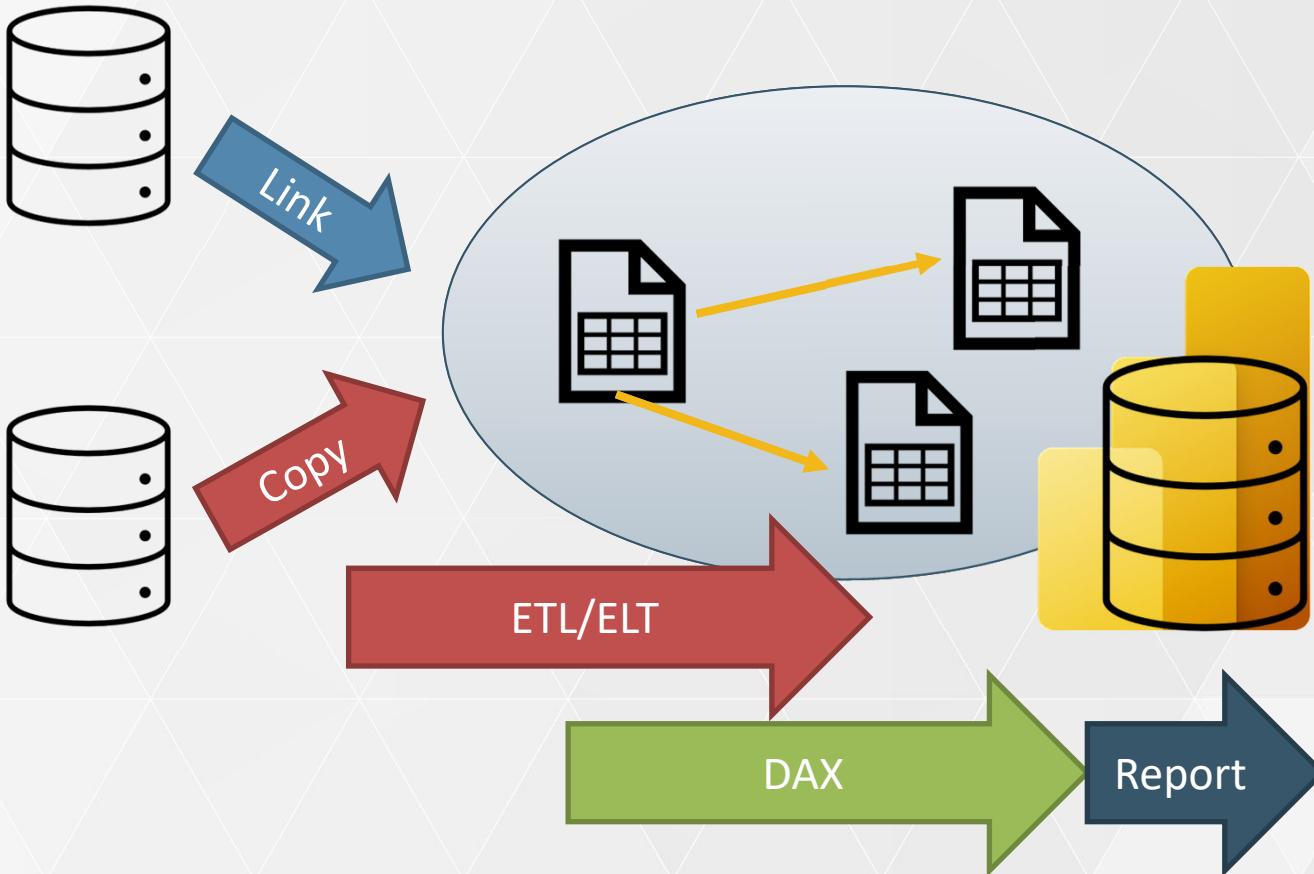


The main business logic is stored in the AS model.



Data is stored in the remote AS; the report links to the AS database and can extend it with local DAX measures.

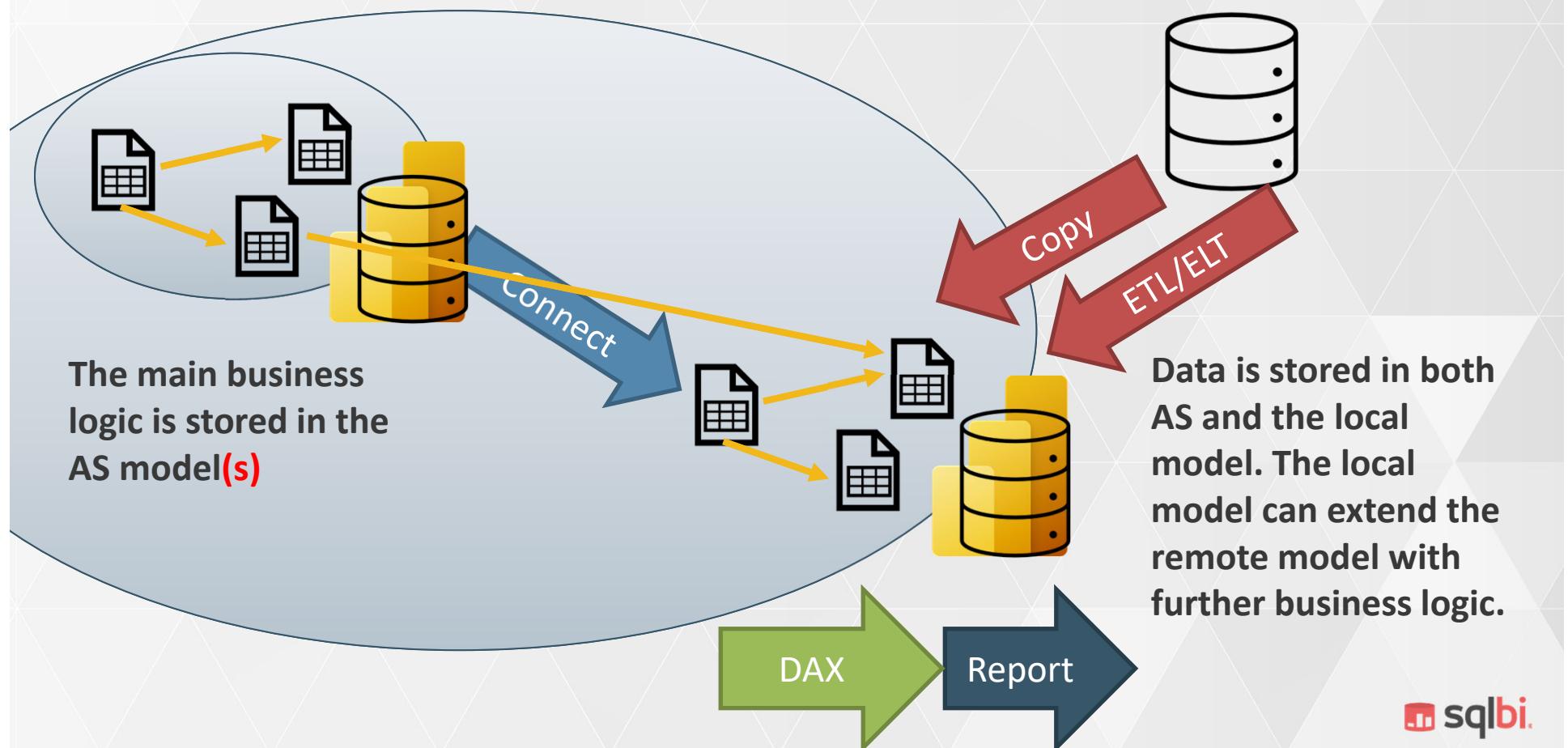
## Composite model (SQL)



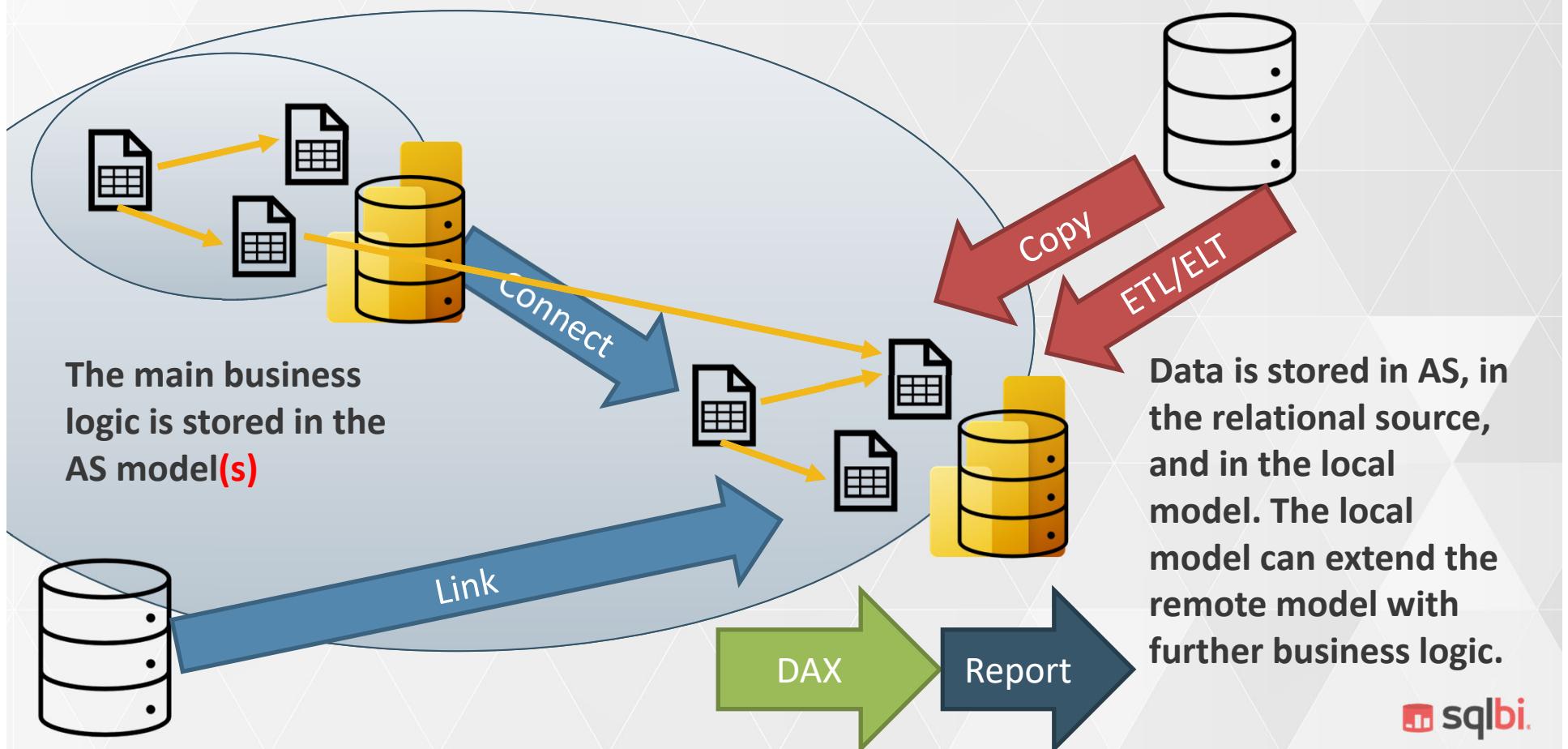
Data is stored in the relational data source **and** in the local VertiPaq storage.

The AS model contains the business logic and the VertiPaq tables.

## Composite model (AS)



# Composite model (AS and SQL)



# Model types

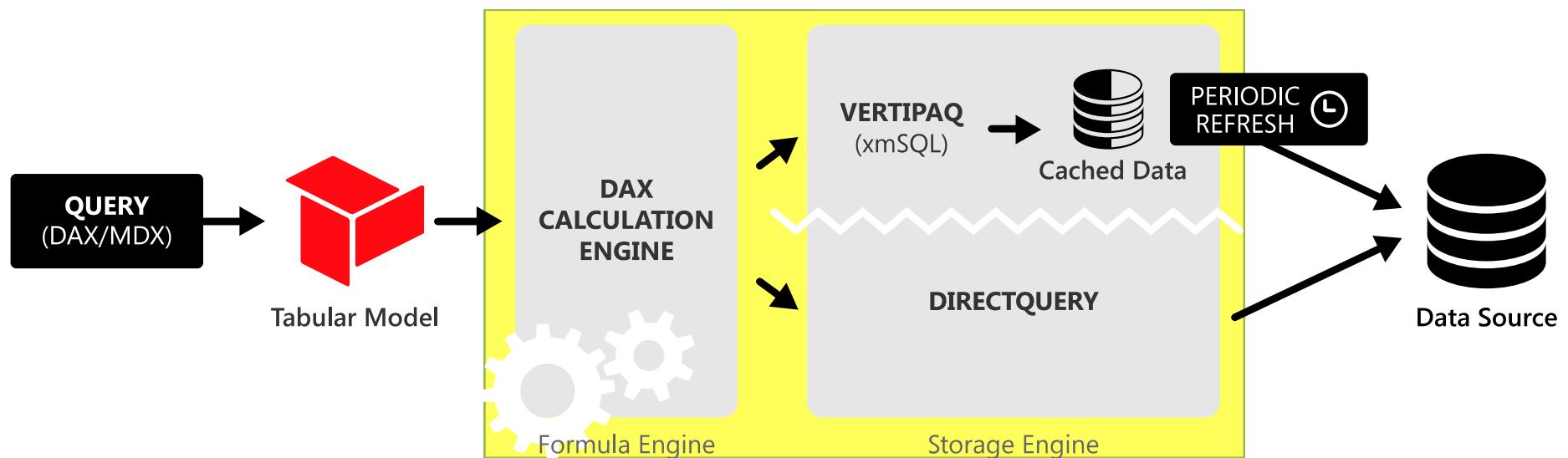
- Import model
  - All tables are stored in VertiPaq
- DirectQuery model
  - All tables are from a single DirectQuery island
- Composite model
  - Some tables in VertiPaq and some in DirectQuery
  - Multiple DirectQuery islands
  - Official name:
    - DirectQuery for Power BI datasets and Azure Analysis Services
- The model type affects multiple choices and performance

An introduction to the two engines running inside Tabular

# Formula engine & Storage engine



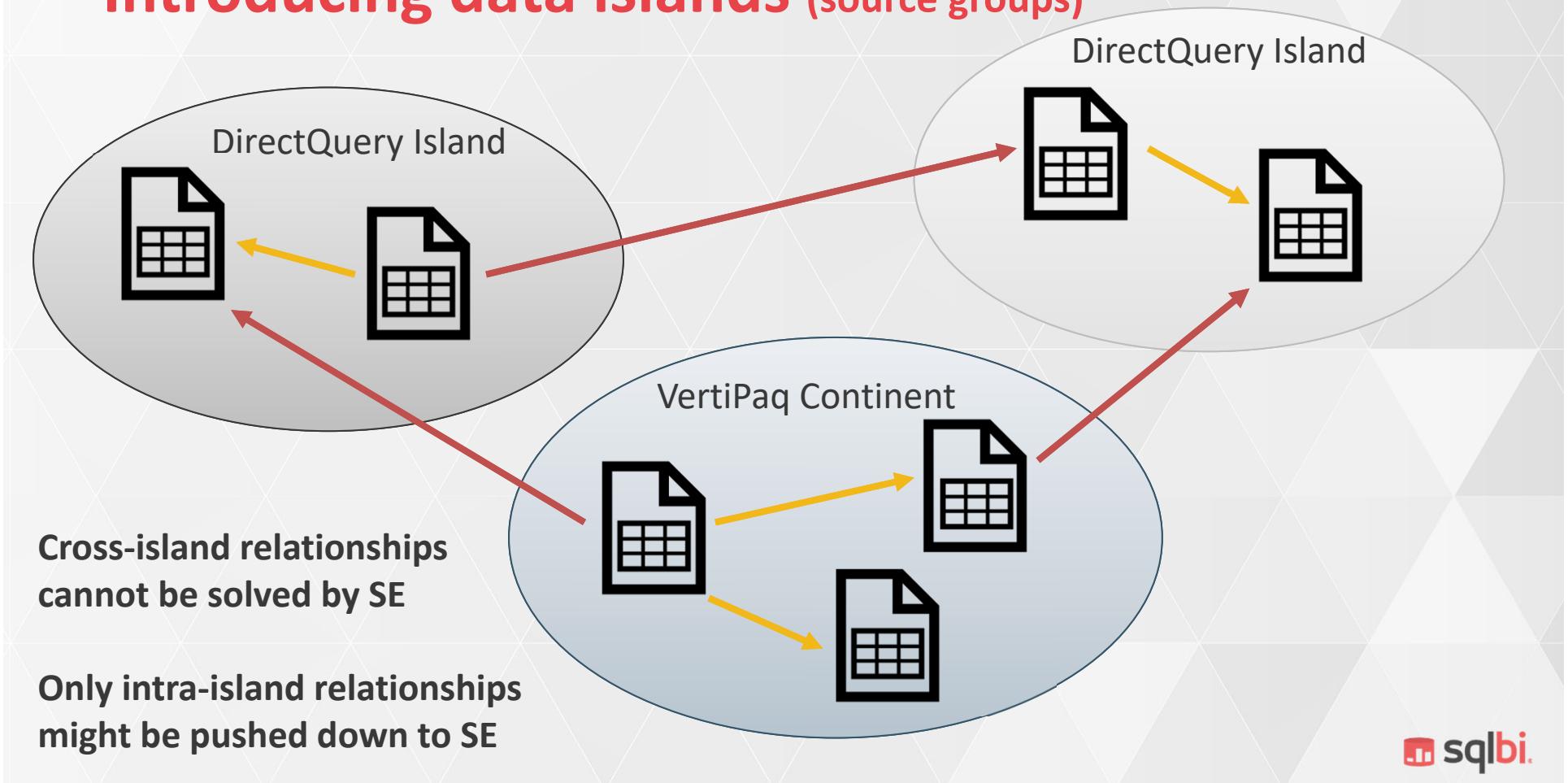
# Tabular's two engines



# Formula engine vs Storage engine

- One formula engine, multiple storage engines:
  - VertiPaq, for imported tables
  - DirectQuery over relational sources (SQL)
  - DirectQuery over Analysis Services (AS)
- The Formula Engine (FE) computes any DAX expression
- FE retrieves data from the Storage Engine (SE)
  - FE makes request to SE
  - SE can also perform some basic calculations
  - SE cannot communicate with other SE

# Introducing data islands (source groups)



# Query resolution requires multiple steps

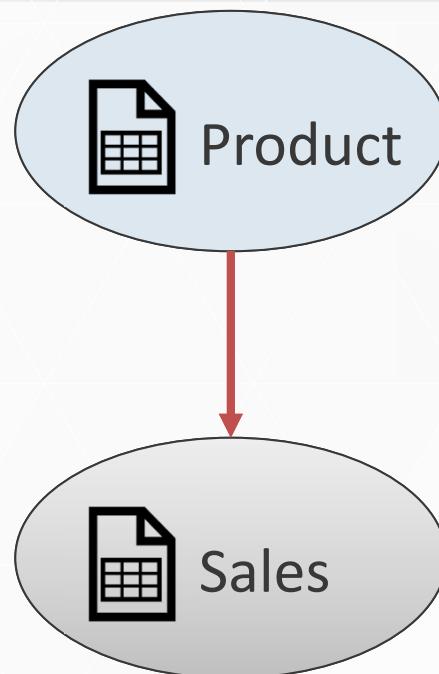
Some expressions can be computed using a single SE query, others require multiple steps.

In this example, SUMMARIZECOLUMNNS gathers data from different islands and merges the results.

DISTINCTCOUNT ( 'Product'[Brand] )

SUM ( Sales[Quantity] )

SUMMARIZECOLUMNNS (  
    'Product'[Brand],  
    "@Qty", SUM ( Sales[Quantity] )  
)



## Query execution steps

- Data is gathered by SE and placed in datacaches
- FE works on datacaches to compute the result
- Size of datacaches directly impacts performance
- FE has no way of connecting to the data sources. Every access to a data source needs to go through the SE
- Aggregations executed by the SE reduce the size of datacaches and push calculations down where the data resides

## Basic design principles

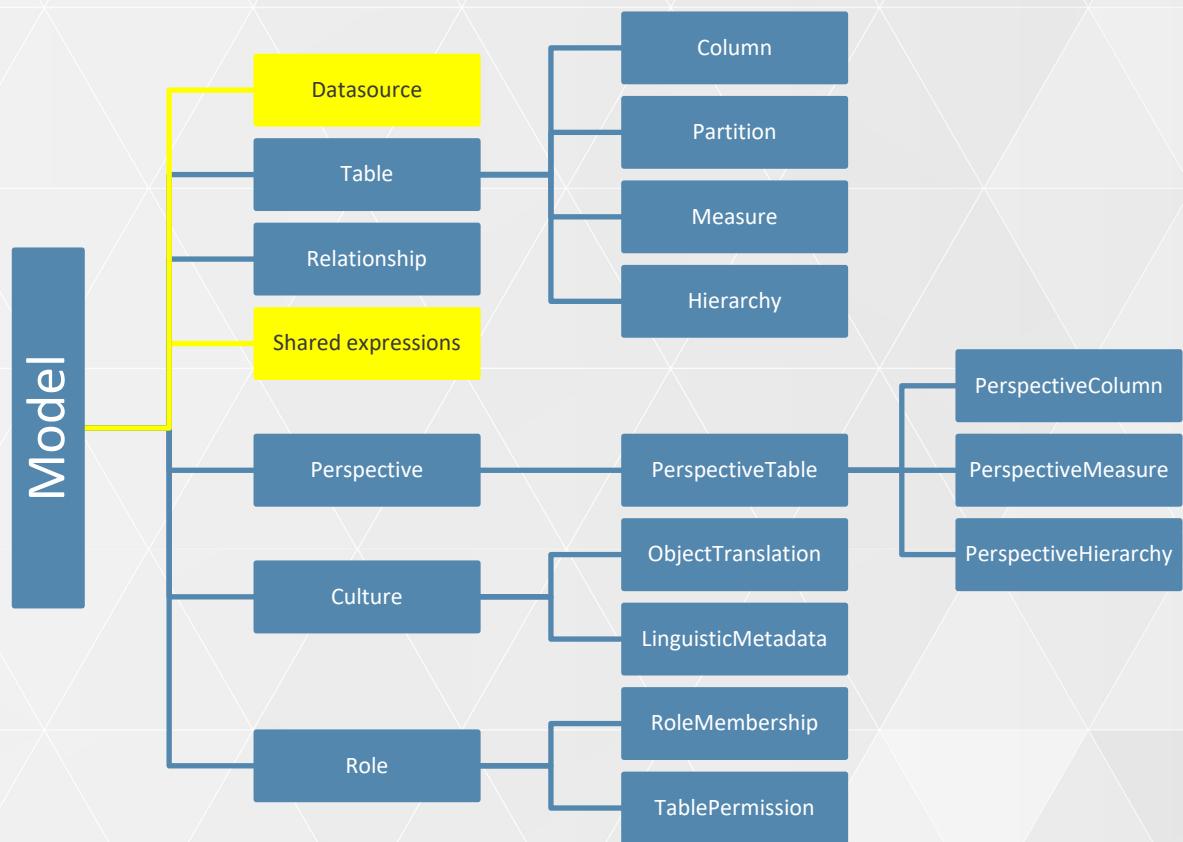
- Tables frequently joined should be in the same island
- Reduce cross-island queries
  - Cross-island relationships are expensive
  - Mixing different Storage Engines might be expensive
- DirectQuery and VertiPaq require different optimization techniques

An overview of the data sources used in Tabular

# Data sources for Tabular



# Data sources for Tabular



## Data sources

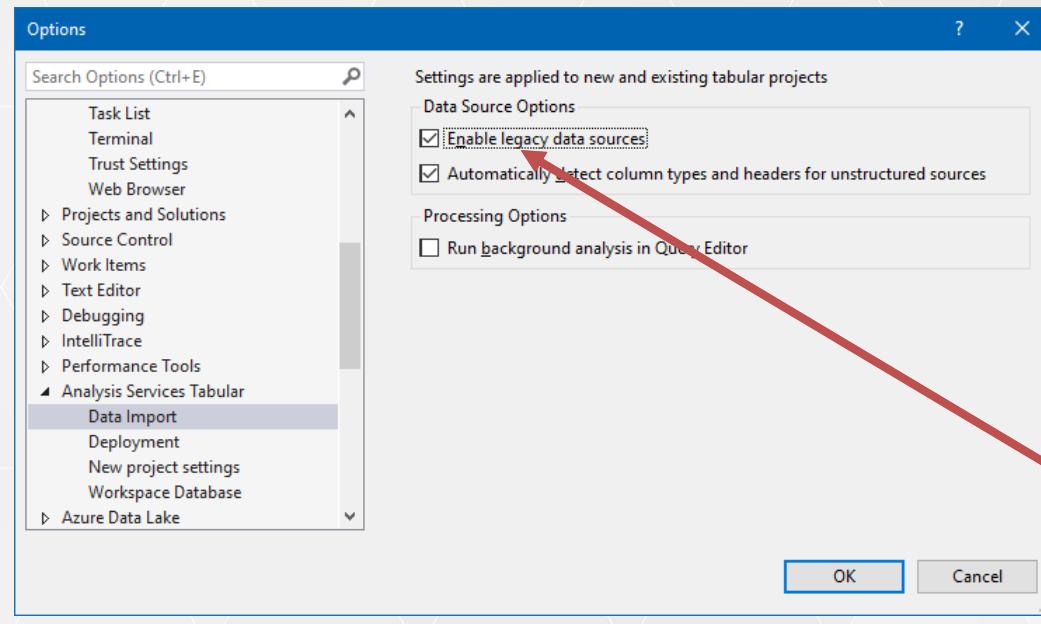
- A model contains a collection of data sources
- Data sources are interfaces to external data
  - Loaded in the VertiPaq database
  - Connected in DirectQuery

## Legacy and Power Query sources

- Legacy
  - AS loads data directly from the connection
  - The query logic is stored inside SQL
  - Legacy data sources are supported in Tabular Editor
- Power Query (Structured)
  - AS executes M code with Power Query
  - The query logic is in M code
  - Power Query is the default

# Legacy data sources

- SSDT defaults to Power Query import
- Legacy sources need to be enabled in SSDT



# Power Query data sources

- Default data source in Visual Studio
- Only choice in Power BI Desktop
- M language
  - Independent from data source
  - Graphical editor
  - Parameters
    - Partitions
    - PBIT template files
    - Incremental refresh

# Query Folding in Power Query

- Query folding
  - Simple transformations are pushed to SQL
  - Complex transformations are executed locally
- Query folding required for DirectQuery sources

## Shared expressions

- Collection of M expressions that can be used in the model
- Useful for
  - Partitions
  - Common subqueries
  - Composite model connections
- Using model expressions for partitions makes it easier to update the query of all the partitions at once
- M queries ***not loaded*** in the model become shared expressions automatically

# Authentication

- Data access requires authorization
- Each data source has different authentication requirements
- For example, SQL server can use:
  - Integrated security (Windows or AD user)
  - Standard authentication (user/password)
- Authentication credentials are specified in the data source

# Impersonation

- Analysis Services
  - Runs under a specific user account
  - Can impersonate a different user when using a data source
- For example
  - SQL Server DEMO recognizes the user DemoReader
  - AS runs under the account AnalysisServicesUser
  - When AS connects to DEMO it can impersonate DemoReader to obtain access under integrated security
- Impersonation is not the same as Authentication

# Client and server operations

- Client-side operations
  - Operations executed in the development tool
  - For example, table preview, or adding a table
  - Use the credentials of the current user
- Server-side operations
  - When a table is processed, or a DirectQuery query is executed
  - AS follows the impersonation configured in the data source
- *Integrated workspace operations are client-side*
- Remember what happened during the guided tour!

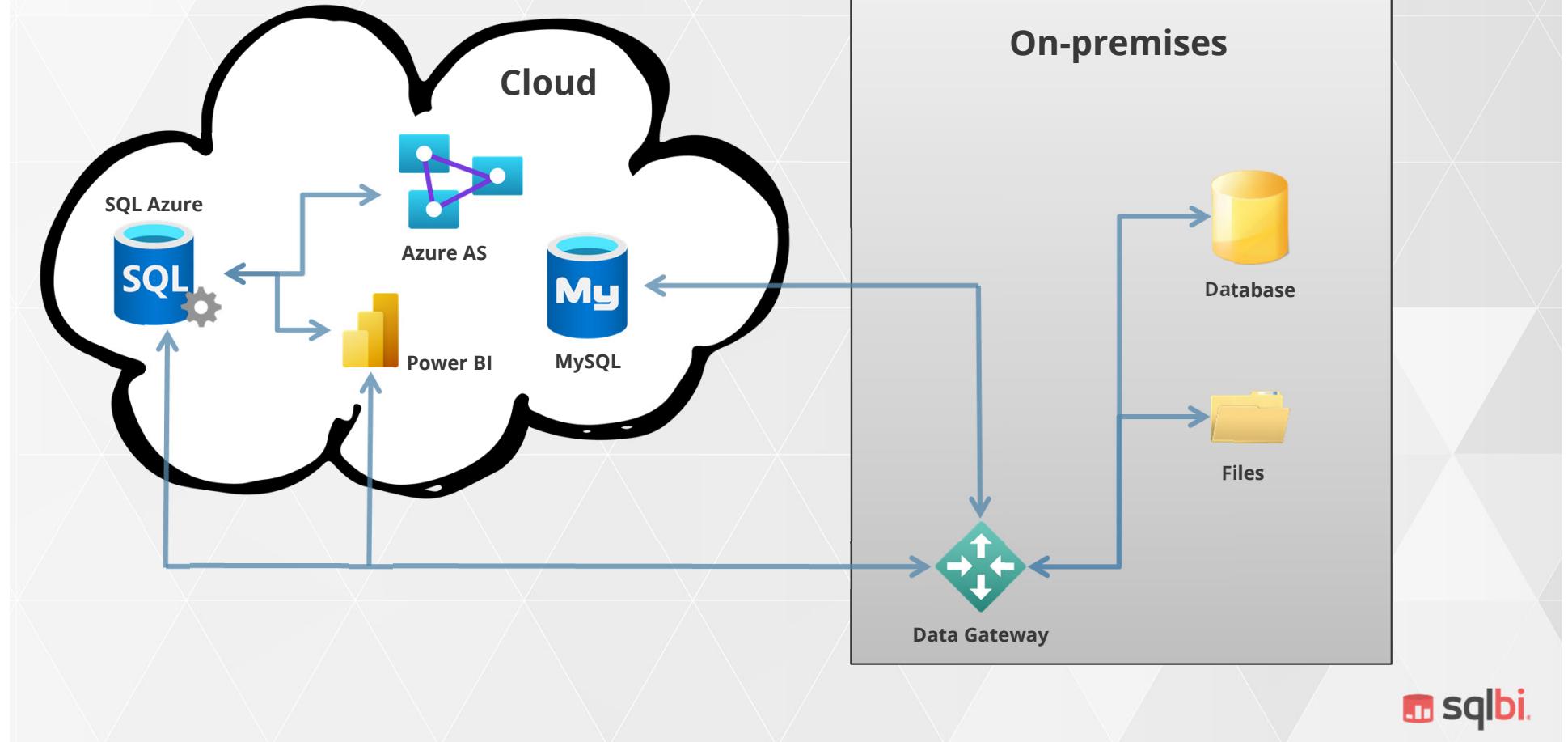
# Impersonation options

Impersonate	Description
Account	The model uses a Windows user account to import or process data from the datasource. The domain and name uses the following format: Domain name\User account name
Current User	Data should be accessed from the datasource using the identity of the user who sent the request. This setting applies <b>only to DirectQuery mode</b> .
Identity (Power Query only)	Specifies a username to access the datasource, but doesn't need to specify the account's password. This setting applies only when the Kerberos delegation is enabled and specifies the <b>service-for-user (S4U) authentication</b> should be used.
Service Account	The model uses the security credentials associated with the Analysis Services service instance that manages the model. Only on-premises.
Unattended Account	Specifies the Analysis Services engine should use a pre-configured unattended account to access data. Used by <b>Excel Services in SharePoint</b> .

## Data gateway

- Required for cloud-published models
  - Unsupported cloud data sources
  - On-premises data sources
- Not required for cloud-published models
  - Supported cloud data sources
- Not needed when publishing on-premises

## Data gateway scenarios



## Working with large tables

- Visual Studio loads data in the workspace database
  - Large tables slow down operations
- Several options
  - Use views (for example TOP 1000)
  - Use a development database with a smaller size
  - Use development partitions – separate from deployed partitions

# Tables or views?

- Tables
  - Quick and dirty
  - High dependency on the database structure
- Views
  - Easy to author and change outside the development tool
  - Can be optimized by the DBA
  - Provide documentation of the project
  - Define a clear contract with the database
  - Easier to debug

## Data sources: conclusions

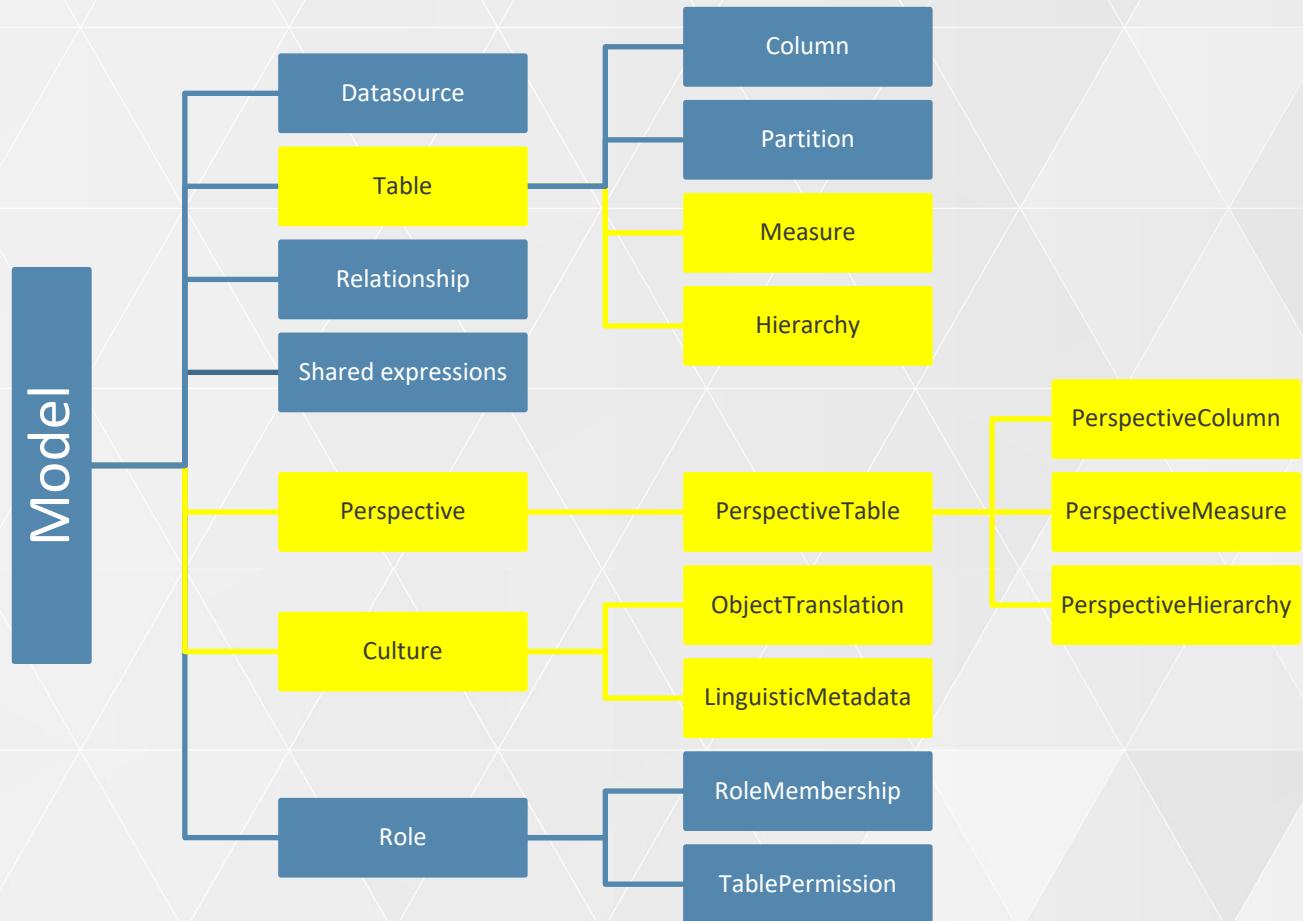
- Data sources are connections to databases
- Legacy, Power Query, Query folding
- Authentication / Impersonation
- Data gateway

Properties of tables and column to present data

# Tabular presentation layer



# Presentation layer



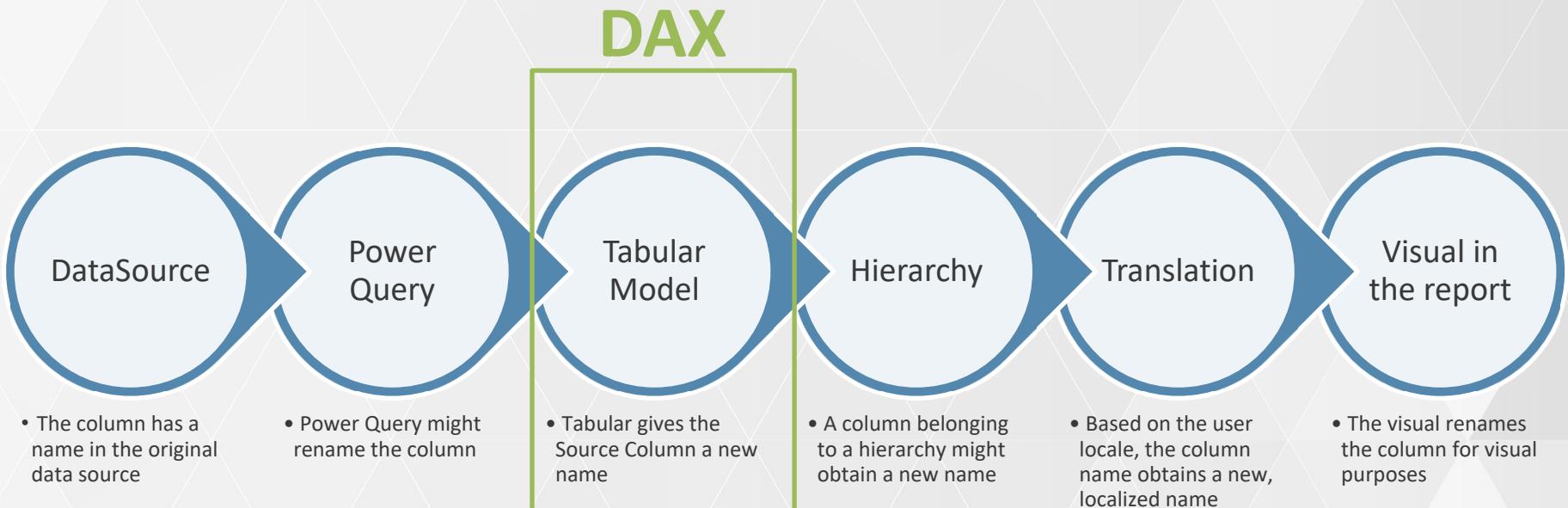
# Presentation layer: the topics

- Naming convention
- Display folders
- Hierarchies
- Formatting
- Sorting
- Summarize by
- Data category
- Table properties
- Perspectives
- Translations
- KPIs
- Q&A

## Naming objects

- Tabular uses the physical model for presentation
- Use names that make sense
  - DimCustomer → Customer
  - FiscalQuarter → Fiscal Quarter
  - FactSales → Sales
  - Sum Of SalesAmount → Sales Amount
- Choose names very early on in the project
  - Changing later might invalidate reports
  - Avoid conflicts between columns and measures

# The path of a column name



# Column names in Power BI Desktop

- Power BI Desktop uses Power Query only
- When you rename a column, Power BI Desktop:
  - Modifies the M code to reflect the new name
  - Does not use the model renaming feature
- Tabular Editor does not modify the M code, it uses the model renaming feature
- Model renaming is supported in Power BI, just not used by the user interface editing the model

## Display folders

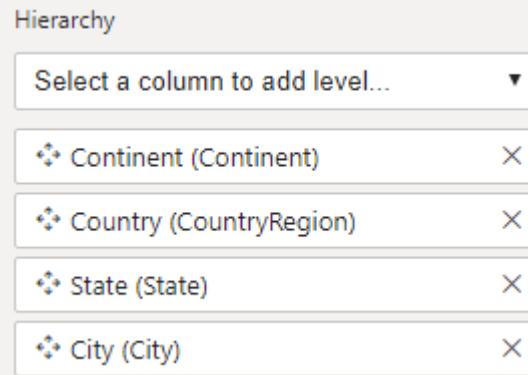
- Folders to group similar entities
- Applies to both measures and columns
- Tables still drive the folder hierarchy
- One entity can belong to multiple folders
  - Use semicolon (;) to separate multiple folders for the same entity
  - Use backslash (\) to create subfolders

# Introducing hierarchies

- Predefined exploration paths
  - Year – Month – Day
  - Category – Subcategory – Product
  - Make it easier to browse a model
- A hierarchy can contain columns from a single table only
- Hierarchies are not available in MDX with DirectQuery over SQL models
  - Mostly relevant for Excel client

## Renaming columns in hierarchies

- Columns in a hierarchy can have a different name
- CountryRegion (column) is named just Country
- The new name is displayed only by navigating the hierarchy



Model\Table\Hierarchy\Level\{Name, Column}

## Ragged hierarchies

- **Hide Members** Property in Hierarchy
  - Set to **Hide blank members** instead of Default
- Mostly useful in parent-child hierarchies
- Hide members that should be hidden to the user
- The feature depends on the client tool
  - **Power BI ignores the setting**
  - Excel hides levels correctly (using MDX)

# Format strings

- Date
  - Locale date format is the best option
- Numbers
  - General
  - Decimal
  - Whole Number
  - Currency
  - Percentage (x 100)
  - Scientific
- Custom format string
- Dynamic format string only available in calculation items

Model\Table\{Column, Measure}\FormatString

## Sorting options

- Attributes that need sorting:
  - Month name
  - Week of the day
  - Product Size
- Sort By Column
  - Sort column needs to be in the same table
  - 1:1 relationship between values and sorters

## Summarize by

- Column property
- Used by Power BI to summarize a column
- Do not overuse:
  - It is always better to showcase any calculation with a measure
  - It conflicts with **DiscourageImplicitMeasures**, activated by calculation groups
  - Useful only for small self-service BI models that are not published as shared datasets

## Data category

- Property of tables and columns
- AS stores and publishes it, but it does not use it
- Client tool uses this metadata for different purposes
- Power BI:
  - Geographic attributes in Map visuals and Q&A
  - **Image URL** to show images in table / matrix visuals
  - **Web URL** shows a link in table / matrix visuals
  - **Barcode** scanned by mobile apps to filter data

Model\Table\Data Category, Model\Table\Column\Data Category

# Visibility

- Hidden
  - Object hidden from UI
  - Power BI can show hidden objects
  - Available for tables, columns, and measures
- Private
  - Available only for tables
  - Power BI does not show private tables
  - A private table should also be hidden, but that is not enforced

Model\Table\Hidden, Model\Table\Private – Model\Table\Column, Model\Table\Measure

# Perspectives

- A perspective filters the content of the model
  - Tables
  - Columns
  - Measures
  - Hierarchies
- Clients connect to either the model or a perspective
- Useful feature on large models with many tables

## Perspectives are not security

- A perspective hides entities, it does not protect them
  - A perspective only hides metadata, not data
  - DAX and MDX code both unaffected by perspectives
- You **cannot** :
  - Define a default perspective
  - Assign a perspective to a user
  - Restrict perspective visibility to users
  - Restrict DAX/MDX queries over hidden data

## Table key identifier

- Sets the Row Identifier for the table
  - Identifies the primary key of the table
  - AS checks that the key has unique values
- Only one column can be set as the key
- Set by the Mark as Date table feature
- Used for the Keep Unique Rows feature
  - Power BI groups by the key if required, instead of grouping by specific columns

## Keep Unique Rows

- If set, the client tool should group by the key instead of using the column value
- Metadata only, no effects on behavior of AS / DAX / MDX
- Power BI uses Keep Unique Rows
- Limited to a full table – you need to build a snowflake to take advantage of the feature
- Not recommended, it is a fragile behavior

## Group By Columns

- Enhanced version of Keep Unique Rows
- Defines a set of columns as the key for a column
- By providing only one column you can obtain
  - Name
  - Key
- By providing multiple columns, you can create composite keys
- DAX enforces the correct use of Group By Columns

Model\Table\Column\Group By Columns ([RelatedColumnDetails](#))

# Date table in Tabular and Power BI Desktop

- Auto Date/Time in Power BI Desktop
  - Automatic creation of date tables
  - One date table for each native date column in the model
    - Calculated columns are not considered
    - Calculated tables are not considered
- Mark as date table
  - Identifies a unique Date column in the Date table
  - Simplifies the use of time intelligence functions
  - In TOM sets Key=true on the Date column

Model\Table\Column\Key

# Featured tables in Power BI

- Power BI service can have featured tables in a dataset
  - Excel can consume featured tables as Data Types
- Table property:
  - **Description:** description of the featured table
- Column properties:
  - **Key:** Key column for the featured table
  - **Default Label:** Row label for the featured table and Q&A
- Only one column in a table can have Default Label=TRUE or Key=TRUE (but this is not enforced by TOM)
- Additional undocumented data set by Power BI Desktop in table Annotations property

Model\Table\Description - Model\Table\Column\Key | Default Label

## Detail Rows Expression

- Controls drillthrough operations
- Defines DAX expression returning rows
- Supports calculation changing filters (e.g. YTD)
- Supported by Excel (not by Power BI yet)
- Calculation groups do not affect the execution of Detail Rows Expression

Model\Table\Measure\Detail Rows Expression

# KPI

- Visual representation of a value and its trend
- Based on a value and a target
- Budgeting example:
  - Value: sales amount
  - Target: budget
  - Status: comparison of sales and budget
  - Trend: comparison of status with previous period

## KPI: the result

Brand	Margin	Measure	Target	Status	Trend
		Margin %	Margin % Goal	Margin % Status	Margin % Trend
A. Datum	11,679,369.31	58.52%	50.00 %	Green	↓
Azure	595,299.84	53.10%	50.00 %	Yellow	→
Black	1,065,468.41	56.14%	50.00 %	Green	↓
Blue	350,892.43	57.69%	50.00 %	Green	↓
Gold	271,601.04	55.87%	50.00 %	Green	↓
Green	558,332.97	53.50%	50.00 %	Yellow	→
Grey	5,429,781.76	62.39%	50.00 %	Green	↓
Orange	1,057,428.38	55.74%	50.00 %	Green	↓
Pink	935,338.43	55.42%	50.00 %	Green	↓
Silver	758,797.37	54.60%	50.00 %	Yellow	→
Silver Grey	656,428.69	58.43%	50.00 %	Green	↓

## KPIs are based on measures

- A measure can be transformed into a KPI
- For the **SalesAmt** measure, Tabular creates three additional measures:
  - [**\_SalesAmt Goal**] is the target
  - [**\_SalesAmt Status**] is the status
  - [**\_SalesAmt Trend**] is the trend
- The code and properties of these measures are defined in the KPI object inside the measure

## KPI properties

- Properties in the KPI object are exposed in measures
- The value of a KPI is the value of the base measure
- Other properties:

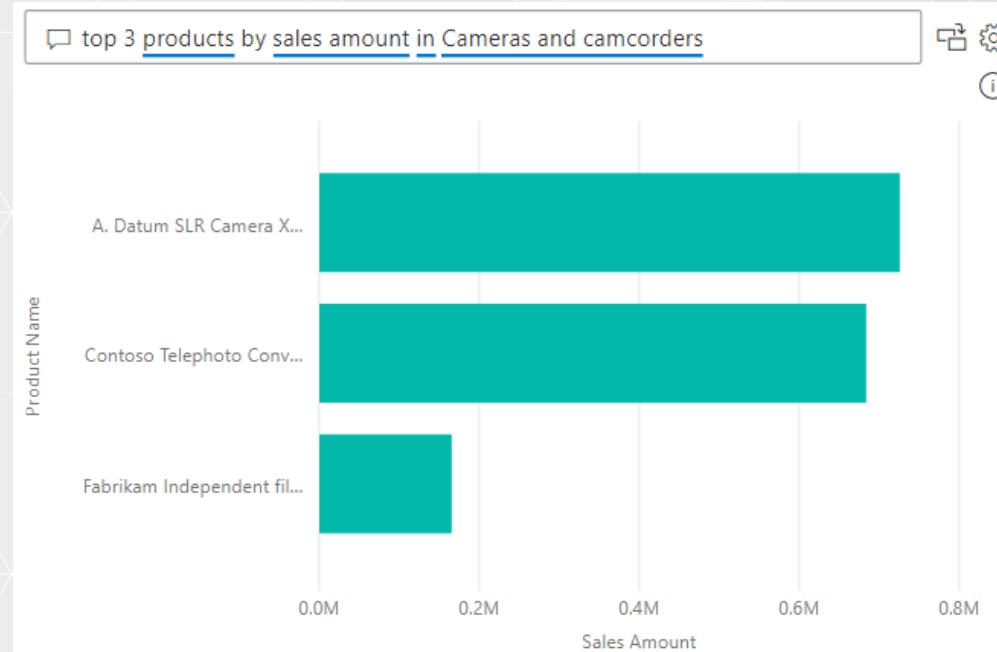
Property	Target	Status	Trend
Description			
Expression	X	X	X
Format string	X		
Graphic		X	X

# Translations

- Metadata can be translated
  - Tables, display folders, columns, measures, descriptions
- Data cannot be translated
  - Unlike Multidimensional, where it is an option
- Tabular editor offers an integrated translation editor
- Using Visual Studio, translations are much harder to handle & require an external JSON file
  - Open-source editor: [SSAS Tabular Translator](#)
  - Open-source editor for PBI Desktop: [Metadata translator](#)
  - Docs: [Translations in Analysis Services tabular models](#)

## Questions and answers

- Q&A lets users ask questions in plain English
- Also works offline
- Needs to be configured properly to obtain good results

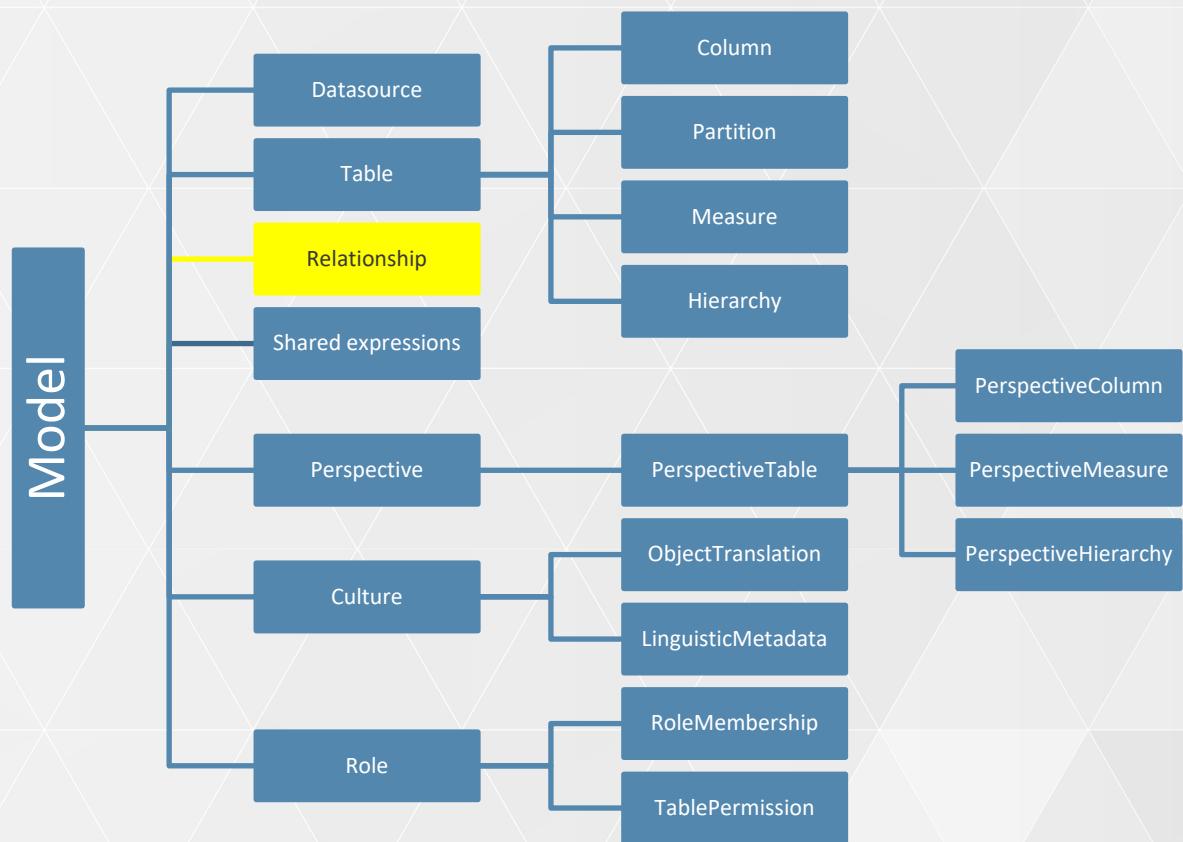


Model\Culture\LinguisticMetadata

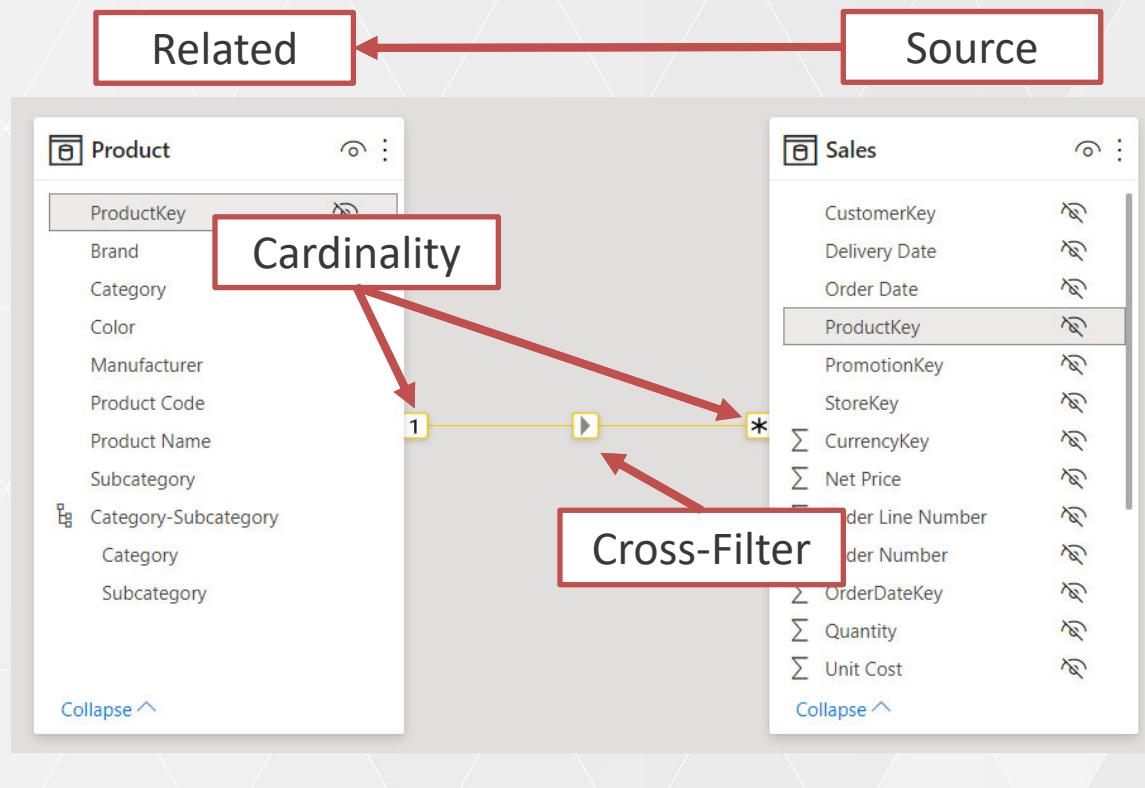
# Understanding relationships



# Relationships in TOM



# Introducing relationships



# Introducing table expansion

Sales (many)

ProductKey	Amount
1	100
2	120
1	150

Product (one)

ProductKey	Name
1	Apple
2	Orange

\*

1

Sales (expanded)

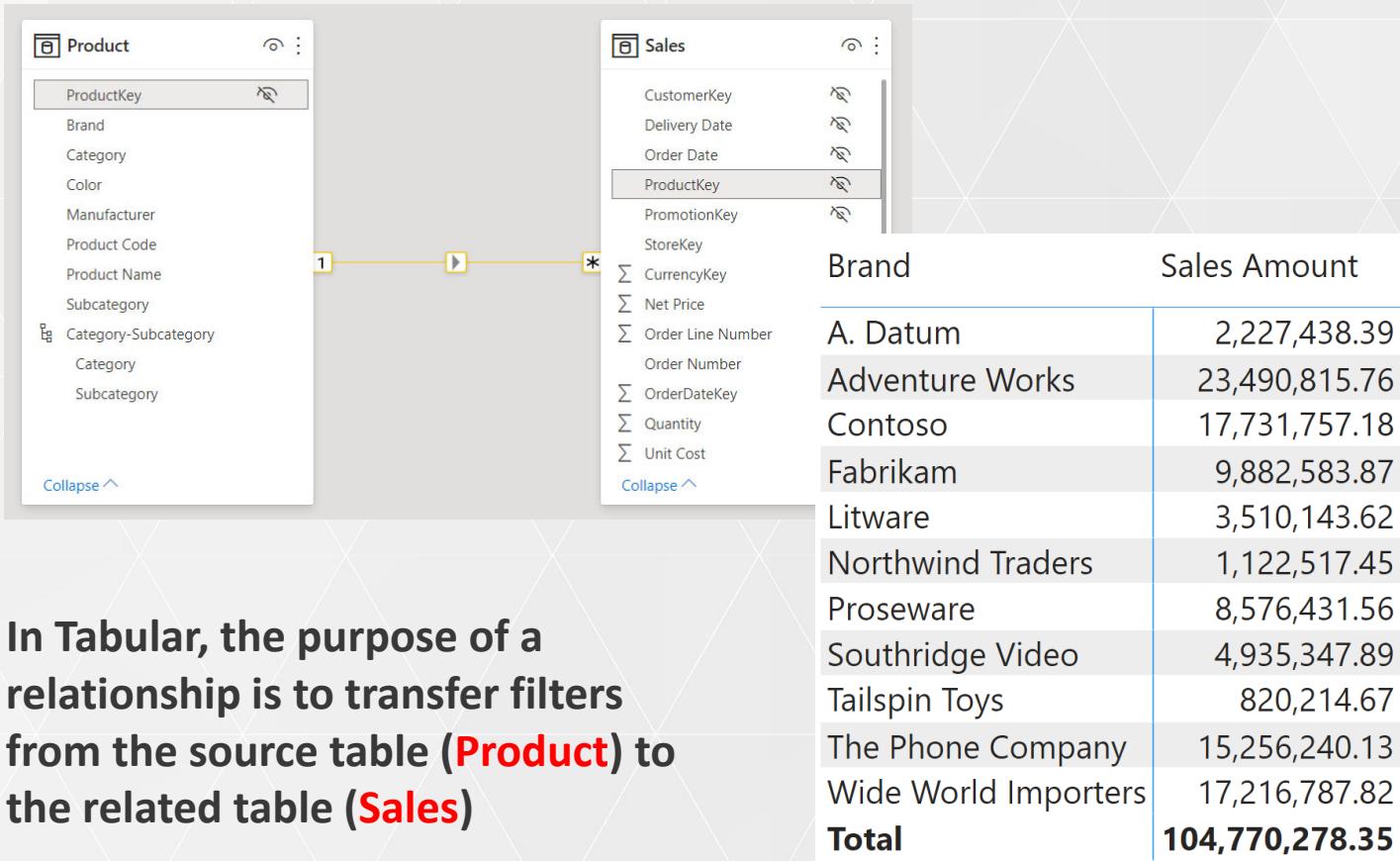
ProductKey	Amount	ProductKey	Name
1	100	1	Apple
2	120	2	Orange
1	150	1	Apple

# Cardinality

- One-side
  - Column needs unique values
  - Column cannot be blank
  - Target of table expansion
  - Source for filter context propagation
- Many-side
  - Column may contain duplicates
  - Source of table expansion
  - Target for filter context propagation

Model\Relationship\FromCardinality (ToCardinality)

# The purpose of relationships



In Tabular, the purpose of a relationship is to transfer filters from the source table (**Product**) to the related table (**Sales**)

## Cross-filter direction

- Single
  - Filter context propagates only in one direction
  - Default behavior
  - Safe, fast, convenient
- Both (bidirectional)
  - Filter context propagates in both directions
  - Needs to be activated
  - Unsafe, slow, extremely dangerous

# Types of relationships

- One-to-many
  - The most common type of relationship
- One-to-one
  - Quite uncommon
  - Expansion goes both ways
  - Cross-filter needs to be both
- Many-to-many cardinality relationship
  - Generates complex calculations
  - No expansion
  - Need to decide the cross-filter direction

## Many-to-many cardinality relationships

- Relationships transfer filters at their cardinality
- Many-to-many cardinality relationships are no exception
- Slicing by other columns leads to complex numbers

Category	Budget
Audio	11,619,250.00
Cameras and camcorders	15,795,722.00
Cell phones	9,361,624.00
Computers	29,196,537.00
Games and Toys	2,244,079.00
Home Appliances	32,748,928.00
Music, Movies and Audio Books	8,771,458.00
TV and Video	18,040,658.00
<b>Total</b>	<b>39,004,512.00</b>

The value shown is the budget of any brand that contains at least one product of the selected category

In short: it does not make sense

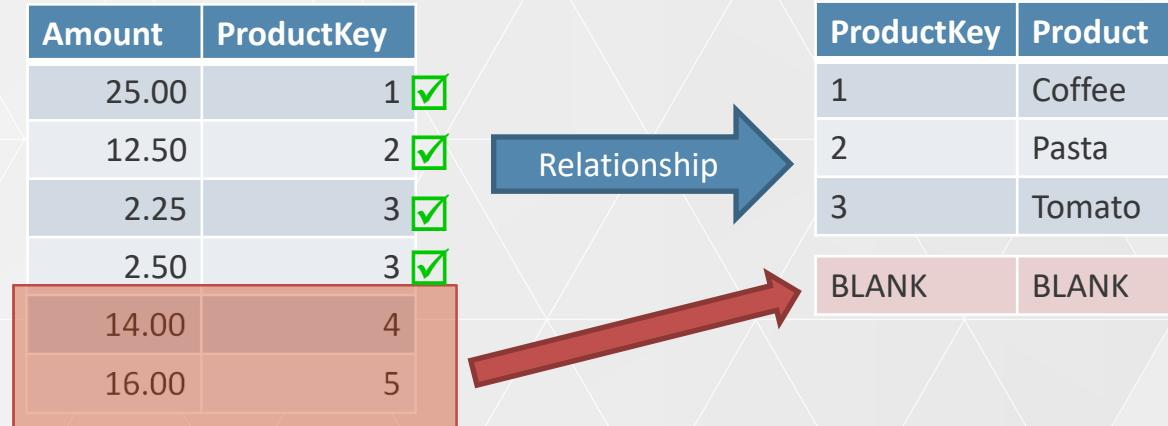
## Specificity of DateTime columns

- DateTime contains both date and time
- Configure the relationship to use only the date part
- Possible values
  - DateAndTime
  - DatePartOnly
- Source column still contains the time, but the relationship uses only date to perform matching
- Filtering the date key on the one-side automatically triggers a REMOVEFILTERS on the table

## Security filtering behavior

- Cross-filter direction for security purposes
- A relationship can be bidirectional
- But still be unidirectional for security purposes
- Values:
  - OneDirection
  - BothDirections

## The additional blank row



Tables with an incoming relationship might contain an additional blank row, created by DAX to guarantee referential integrity

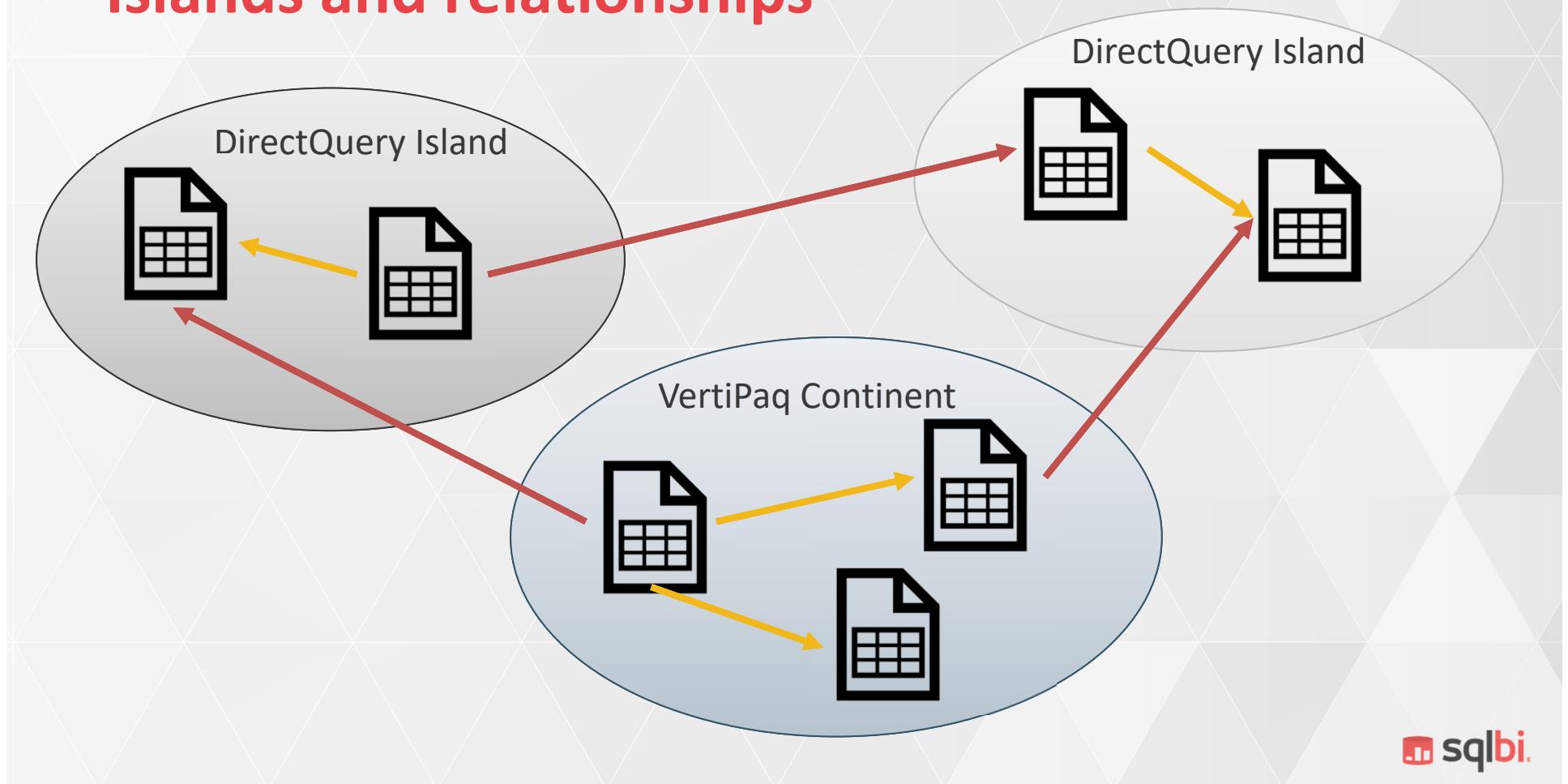
## Rely on referential integrity

- DirectQuery setting only
- Tabular cannot validate referential integrity
- Boolean value, when set
  - The query uses INNER JOIN instead of OUTER
  - No blank row is added to the target table

## Limited relationships (weak relationships)

- When uniqueness constraint cannot be enforced  
*(column is not a key in both tables)*
  - Cross-island relationships are always limited
  - Many-to-many cardinality relationships are always limited
- Limitations
  - Blank row is not enforced
  - Table expansion does not occur

# Islands and relationships



# Limited: no expansion, no blank row

Slice by Product[Brand]

Brand	Budget
Adventure Works	4,985,172.00
Contoso	7,127,903.00
Fabrikam	8,667,819.00
Litware	4,284,028.00
Northwind Traders	911,918.00
Proseware	3,192,659.00
Southridge Video	1,643,555.00
Tailspin Toys	600,524.00
The Phone Company	2,233,721.00
Wide World Importers	3,579,429.00
<b>Total</b>	<b>39,004,512.00</b>

Slice by Budget[Brand]

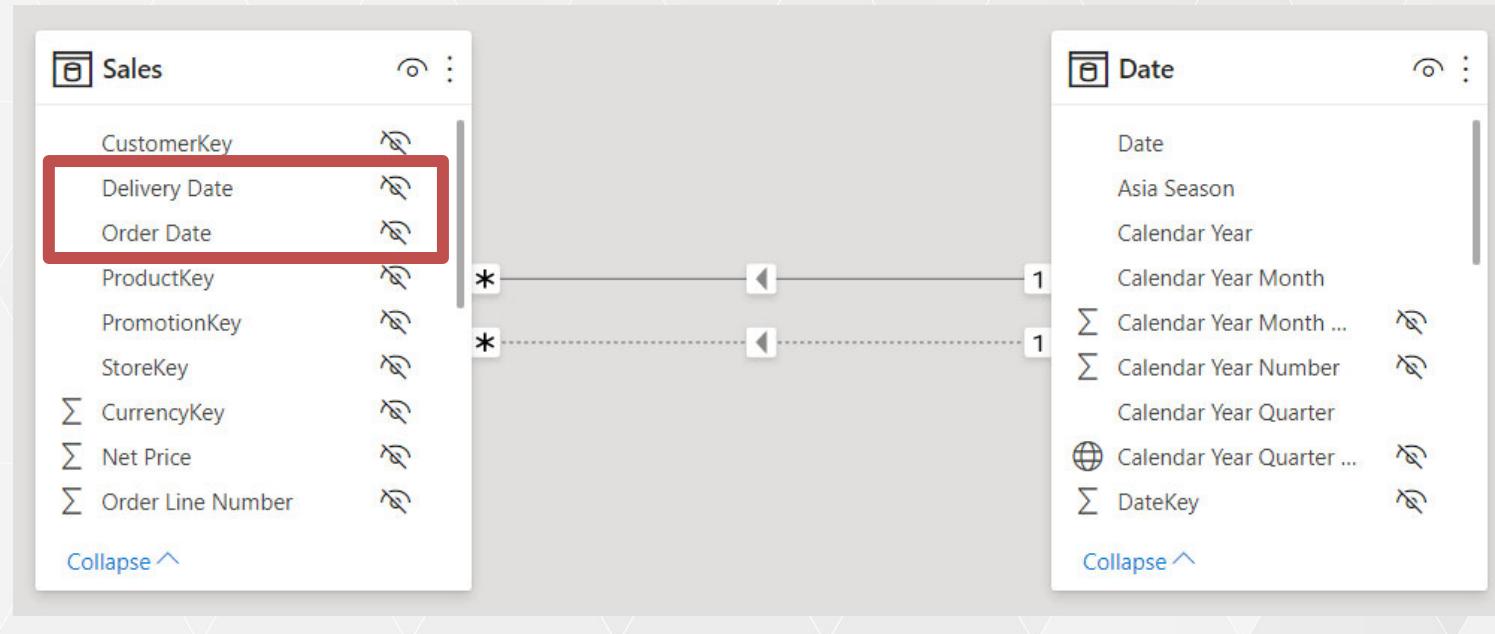
Brand	Budget
A. Datum	1,777,784.00
Adventure Works	4,985,172.00
Contoso	7,127,903.00
Fabrikam	8,667,819.00
Litware	4,284,028.00
Northwind Traders	911,918.00
Proseware	3,192,659.00
Southridge Video	1,643,555.00
Tailspin Toys	600,524.00
The Phone Company	2,233,721.00
Wide World Importers	3,579,429.00
<b>Total</b>	<b>39,004,512.00</b>

# Ambiguity



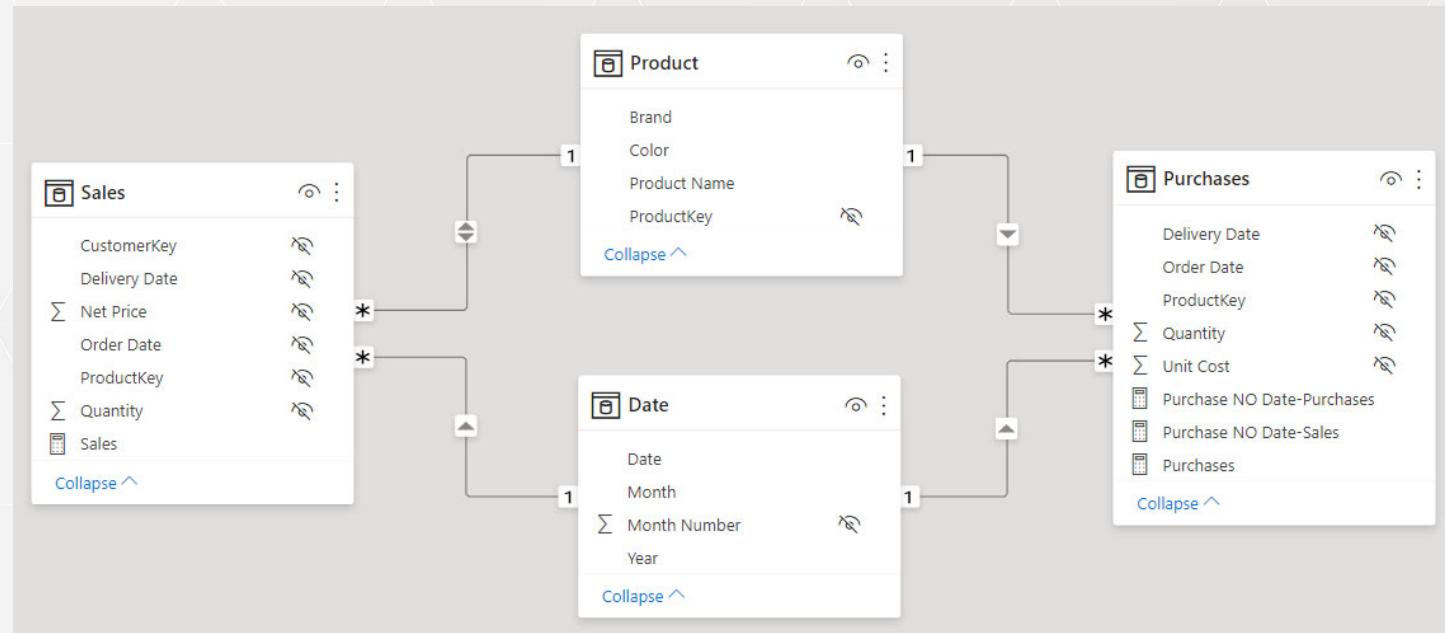
# Ambiguity

- Multiple paths between any two tables
- Tabular does not work on ambiguous models
- Only one relationship can be active



# Ambiguity and bidirectional cross-filter

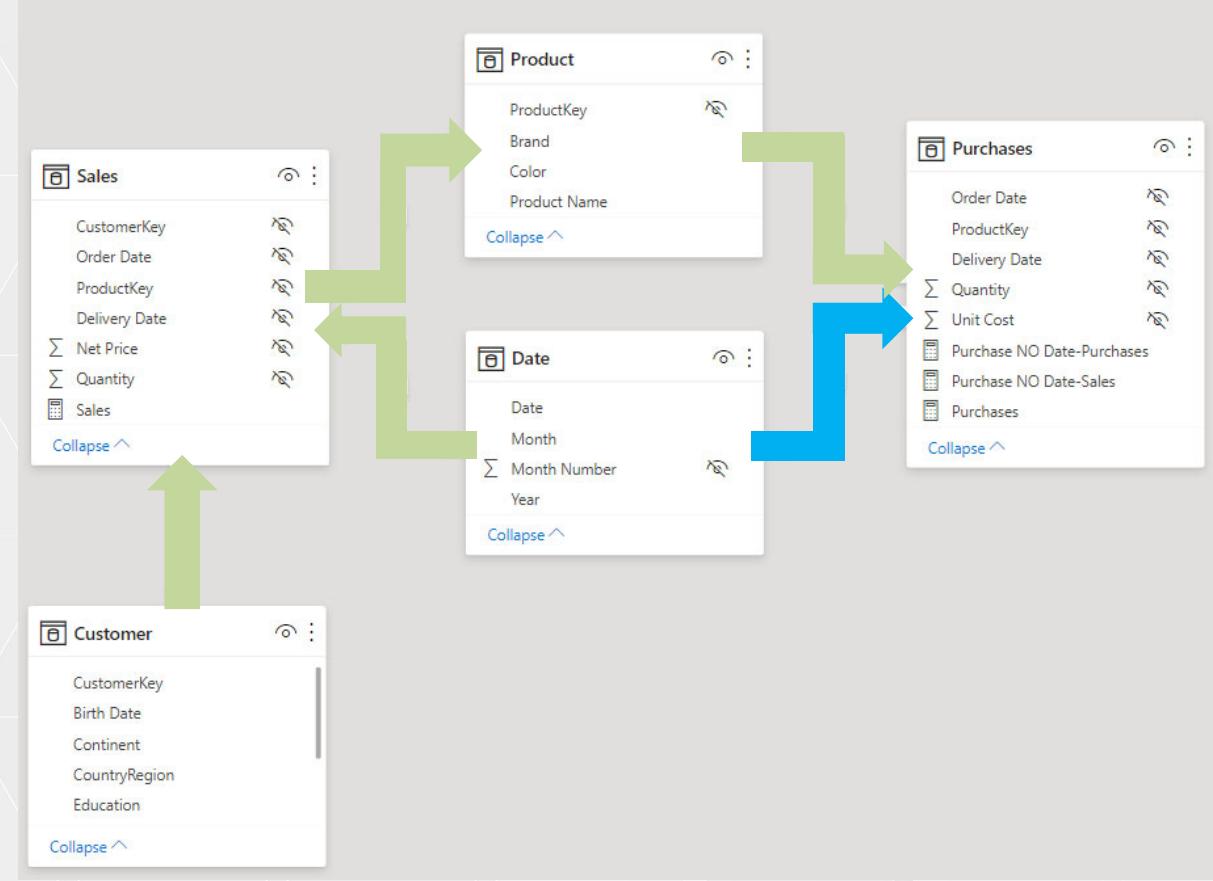
- Bidirectional cross-filter increases the chances of ambiguity
- Filter propagation always goes both ways



# Things are more complex in a larger model

A few questions:

- Does Date filter Sales?
- Does Customer filter Purchases?
- Does Date filter Purchases?
- Which subset of sales is actually filtering Purchases?



Ambiguity - Power BI Desktop

Alberto Ferrari

File Home Insert Modeling View Help External Tools

Get data Excel Power BI datasets SQL Server Data Refresh Recent sciences Transform data New visual Text box More visuals New measure Quick measure Publish

Continent

- Asia
- Europe
- North America

Brand	CY 2007	CY 2008	CY 2009	Total
A. Datum	133,602.05	37,469.91	32,064.81	<b>203,136.77</b>
Adventure Works	244,665.47	82,056.59	30,410.61	<b>357,132.67</b>
Contoso	294,648.80	153,409.58	129,808.34	<b>577,866.71</b>
Fabrikam	127,649.11	200,002.96	100,148.99	<b>427,801.06</b>
Litware	70,921.44	178,555.93	112,368.11	<b>361,845.48</b>
Northwind Traders	104,537.80	17,747.95	2,129.45	<b>124,415.20</b>
Proseware	94,045.80	88,333.39	62,860.34	<b>245,239.53</b>
Southridge Video	86,181.47	37,907.98	34,537.13	<b>158,626.58</b>
Tailspin Toys	7,432.48	7,901.42	9,939.46	<b>25,273.36</b>
The Phone Company	32,190.44	51,583.06	27,178.89	<b>110,952.39</b>
Wide World Importers	44,458.69	128,287.73	58,580.36	<b>231,326.79</b>
<b>Total</b>	<b>1,240,333.55</b>	<b>983,256.49</b>	<b>600,026.49</b>	<b>2,823,616.54</b>

Report

Page 1 of 1

Visualizations Fields

Purchases

- Purchase NO Date
- Purchase NO Date
- Purchases

Sales

Customer

Date

Product

Values

Add data fields here

Drill through

Glossary

Off

Keep all filters

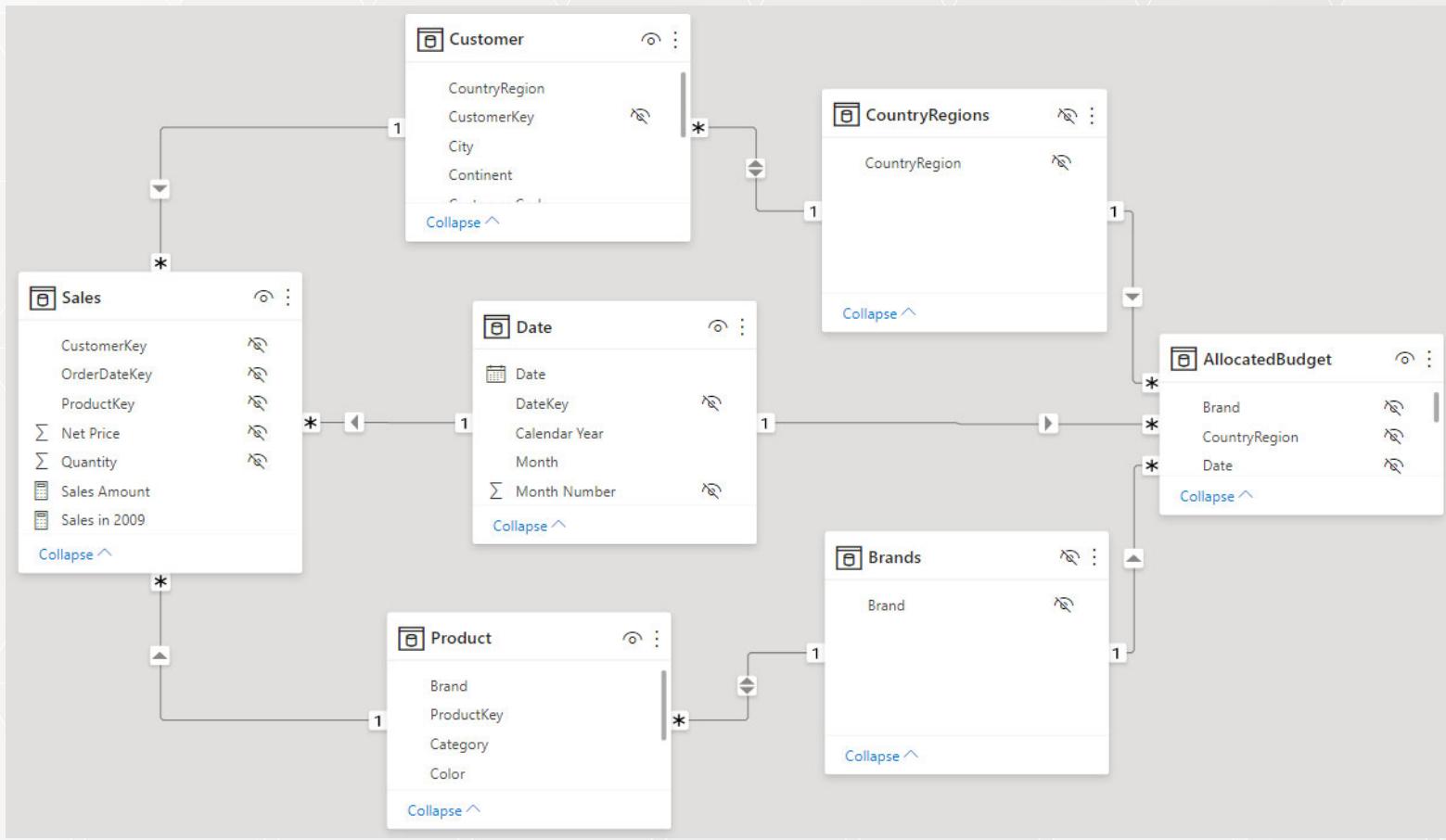
Off

Add drill-through fields here

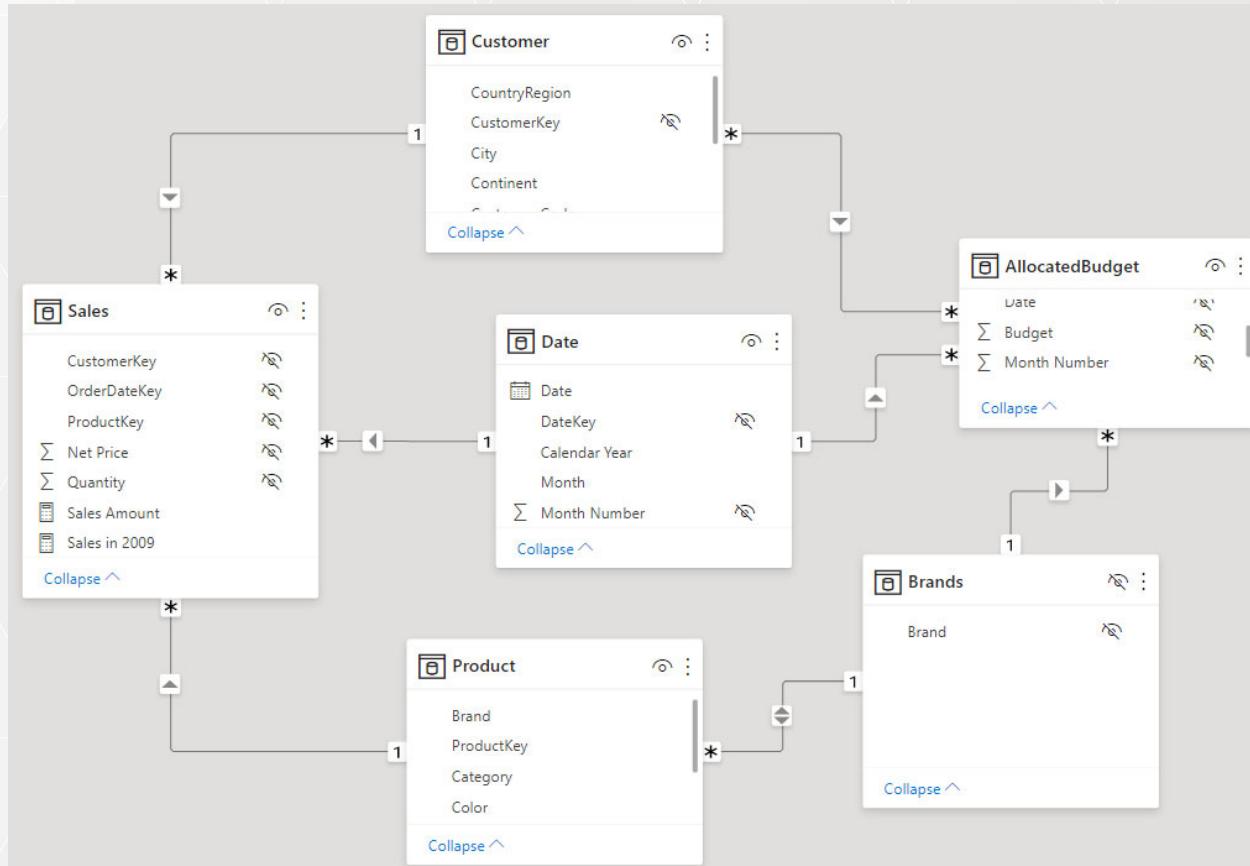
# Never work with an ambiguous model

- Ambiguity is not always evident
  - The engine might consider as non-ambiguous a model that is indeed ambiguous
  - Adding a table might completely change the relationships setup
- Bidirectional cross-filter is the major culprit
  - It is an advanced feature
  - Make sure no ambiguity is present

# Non-ambiguous model



# Many-to-many cross-filter relationships

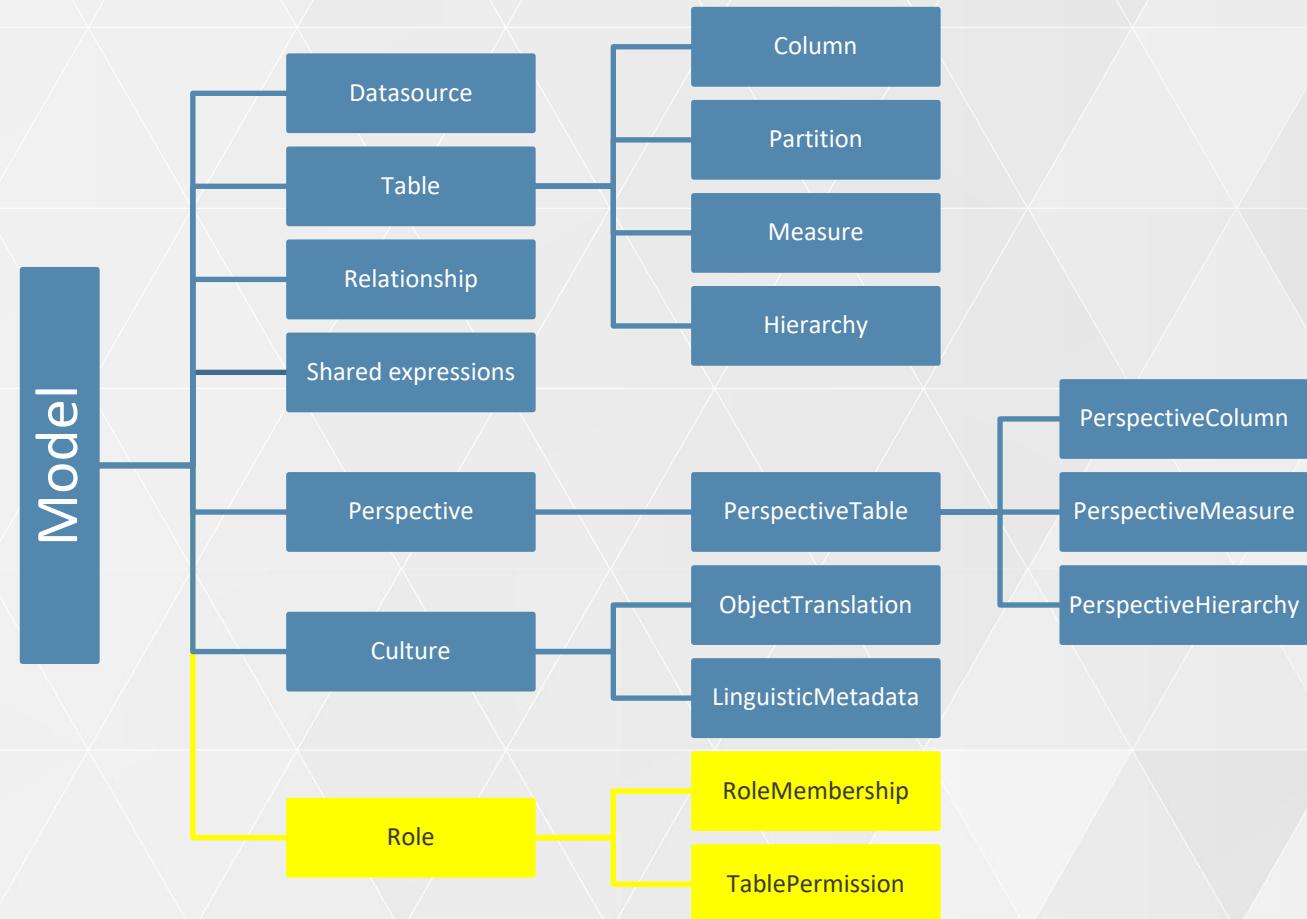


Overview of the security features of Tabular

# Handling Security



# Tabular security model



# Security in AS Tabular

- Model security (administrative security)
- Row-level security (static/dynamic)
- Object-level security

## Security roles

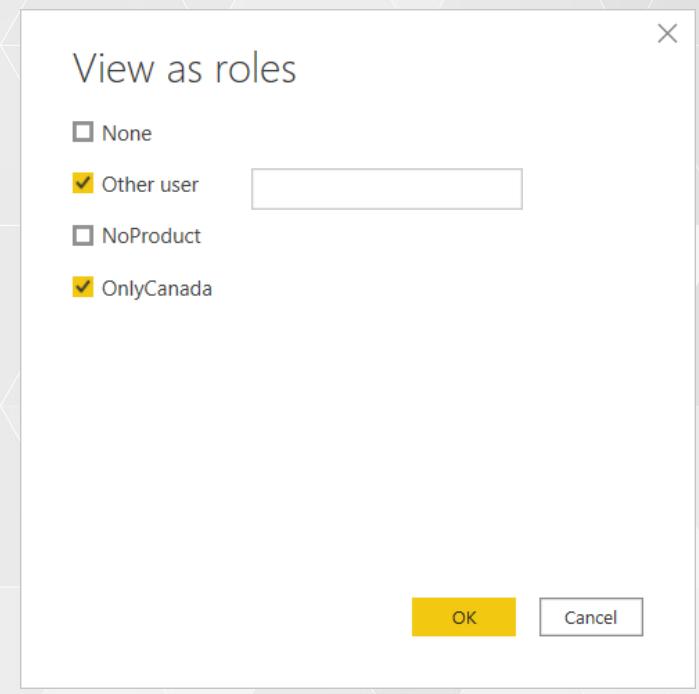
- Security is based on roles
- A role can contain
  - Windows Active Directory users (or groups)
  - Azure Active Directory users (or groups)
- Security rules are activated on roles
- A user can belong to multiple roles

# What can be secured?

- Model security
  - Defines the operations allowed on the entire model
- **RLS:** Row-level security
  - Hides rows in a table, based on a condition
  - Also hides rows in cross-filtered tables
- **OLS:** Object-level security
  - Prevents access to tables or columns
  - Breaks measures depending on hidden objects

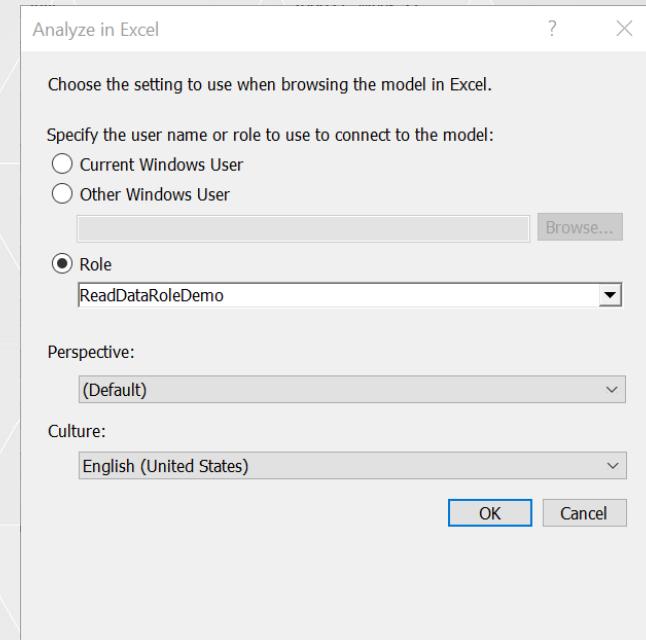
# Testing security in Power BI Desktop

- Use View as to impersonate a role or a user
  - Analyze with a specific **User**
  - Analyze with specific **Roles**
- Choose None to go back to your logged-in user



# Testing security in Visual Studio

- Analyze in Excel
  - Analyze with a specific **User**
  - Analyze with a specific **Role**
- Excel opens with the specified roles/credentials activated
  - Credentials passed in connection string
  - Effective user has to be an SSAS administrator



# Security in connection string

- The role can be set in the connection string
  - Testing purposes, only if the user is an administrator
- Adding
  - Roles=RoleName1, RoleName2...
- Useful in
  - Excel, when browsing a model
  - SSMS, to test query security
  - DAX Studio

## Belonging to multiple roles

- Each role adds permissions
- Being a member of many roles
  - Grants the permissions of all the roles
  - UNION of permissions:  
the user can see a row if it is visible in at least one role
- There is no «deny» rule
- A user cannot be a member of two roles  
if the result mixes OLS and RLS

# Model permissions

- Database-level role permissions
  - None
  - Read
  - Process
  - Administrator
- Administrator level
  - Full access to any table
  - Ignores RLS / OLS
  - Users can deploy / add / change objects

# Server administrator

- Has all permissions on all databases
- Can impersonate roles or username in the connection string
  - Required to test security roles during development
  - Impersonate users that are not administrators
  - If a user is a server or database administrator, security roles are ignored
- At least one user must be server administrator
  - Required during initial setup

# Inherited instance administrators (on-premises)

The screenshot shows the 'Properties' dialog for a connection named 'AzureAD\MarcoRusso'. The left sidebar has sections for 'Select a tab' (Informatic, General, Language, Security), 'Connection' (Server: lab19, Connection: AzureAD\MarcoRusso, View connection properties), and 'Progress' (Ready). The main area displays a table of properties under the 'Security' tab. A red box highlights the first four rows:

Property	Type	Value	Description
Security \ BuiltinAdministratorsAreServerAdmins	bool	true	
Security \ CellPermissionMode	int	0	
Security \ RequireClientAuthentication	bool	true	
Security \ ServiceAccountIsServerAdmin	bool	true	
OLAP \ Query \ DisableCacExpressNonEmpty	int	0	
OLAP \ Query \ NonEmptyBehaviorMode	int	1	
OLAP \ Query \ QueryOptimizerRatio	double	-1	
OLAP \ Query \ RowsetSerializationLimit	int	-1	
OLAP \ Query \ SessionCubesMode	int	1	
Port	int	0	
ResourceGovernance \ DecayIntervalCPUTime	int	60000	
ResourceGovernance \ ReservedComputerForFastQueries	int	75	
ResourceGovernance \ ReservedComputerForProcessing	int	75	
Security \ BuiltinAdministratorsAreServerAdmins	bool	true	
Security \ CellPermissionMode	int	0	
Security \ RequireClientAuthentication	bool	true	
Security \ ServiceAccountIsServerAdmin	bool	true	
ServerTimeout	int	3600	
TempDir	string	K:\Data\MSAS15..	
ThreadPool \ Command \ GroupAffinity	string	yes	
ThreadPool \ Command \ MaxThreads	int	0	
ThreadPool \ Command \ MinThreads	int	0	
ThreadPool \ Command \ GroupAffinity	string	yes	
ThreadPool \ IOProcess \ MaxThreads	int	0	
ThreadPool \ IOProcess \ MinThreads	int	0	
ThreadPool \ IOProcess \ PerNumaNode	int	-1	
ThreadPool \ Parsing \ Long \ GroupAffinity	string	yes	

At the bottom of the dialog, there are checkboxes for 'Show Advanced (All) Properties' and 'Save Only Modified Properties', and buttons for 'OK' and 'Cancel'.

- Configurable properties
  - Local administrators are instance administrators
  - Service is administrator

# Static Row-level Security (RLS)

Sales

Date	Product	Color
01/01/2020	T-shirt	Red
01/02/2020	Bike	Blue
01/05/2020	Helmet	Red
02/02/2020	Shoes	White
03/02/2020	Shoes	Blue
03/05/2020	T-shirt	Blue
04/02/2020	Bike	Red

RedSales role

`Sales[Color] = "Red"`

WhiteSales role

`Sales[Color] = "White"`

BikeSales role

`Sales[Product] = "Bike"`

# Multiple roles with static security

Sales

Date	Product	Store
01/01/2020	T-shirt	Seattle
01/02/2020	Bike	Redmond
01/05/2020	Helmet	Seattle
02/02/2020	Shoes	Seattle
03/02/2020	Shoes	Redmond
03/05/2020	T-shirt	Redmond
04/02/2020	Bike	Seattle

Product

Product	Color
T-shirt	Red
Bike	Blue
Helmet	Red
Shoes	White

Store

Store	Manager
Seattle	Paule
Redmond	Claire

RedSales role

Product  
Product[Color] = "Red"  
Store  
FALSE ()

SeattleSales role

Store  
Store[Store] = "Seattle"  
Product  
FALSE ()

## Security = filter context

- Security is applied like a filter context to the table:
  - Only regular relationships propagate security
  - By default, filtering the one-side filters the many-side automatically
  - Security Filtering Behavior controls the security propagation:
    - None – disable security propagation
    - OneDirection – default also for bidirectional filters
    - BothDirections – can be enabled on bidirectional filters

## When is security applied?

- Security is applied when the user connects to a database
- Calculated columns/tables are not affected by security
  - Beware of that, it might create security breaches
- Measures are evaluated at query time
  - Measures obey security
  - Calculated columns/tables do not
- Non Visual Totals with security roles: <https://sql.bi/32441>

# Table and column security (OLS)

Sales		
Date	Product	Store
01/01/2020	T-shirt	Seattle
01/02/2020	Bike	Redmond
01/05/2020	Helmet	Seattle
02/02/2020	Shoes	Seattle
03/02/2020	Shoes	Redmond
03/05/2020	T-shirt	Redmond
04/02/2020	Bike	Seattle

Product	
Product	Color
T-shirt	Red
Bike	Blue
Helmet	Red
Shoes	White

Store	
Store	Manager
Seattle	Paule
Redmond	Claire

# Multiple roles with Object-Level Security (OLS)

Sales		
Date	Product	Store
01/01/2020	T-shirt	Seattle
01/02/2020	Bike	Redmond
01/05/2020	Helmet	Seattle
02/02/2020	Shoes	Seattle
03/02/2020	Shoes	Redmond
03/05/2020	T-shirt	Redmond
04/02/2020	Bike	Seattle

Product	
Product	Color
T-shirt	Red
Bike	Blue
Helmet	Red
Shoes	White

Store	
Store	Manager
Seattle	Paule
Redmond	Claire

## OLS and relationships

- A table cannot be secured if it breaks a relationship



- You can secure the columns involved in the relationship, but not the table

## Measures cannot be secured directly

- You can secure:
  - Rows
  - Columns
  - Tables
- A measure is secured if it depends on secured objects:
  - Measure expression includes secured columns/tables
  - Measure defined in a secured table

# Introducing dynamic security

User	Store	Year
User	Store	2020
S	Seattle	2020
S	New York	2021
SQLBI\Alberto	New York	2021

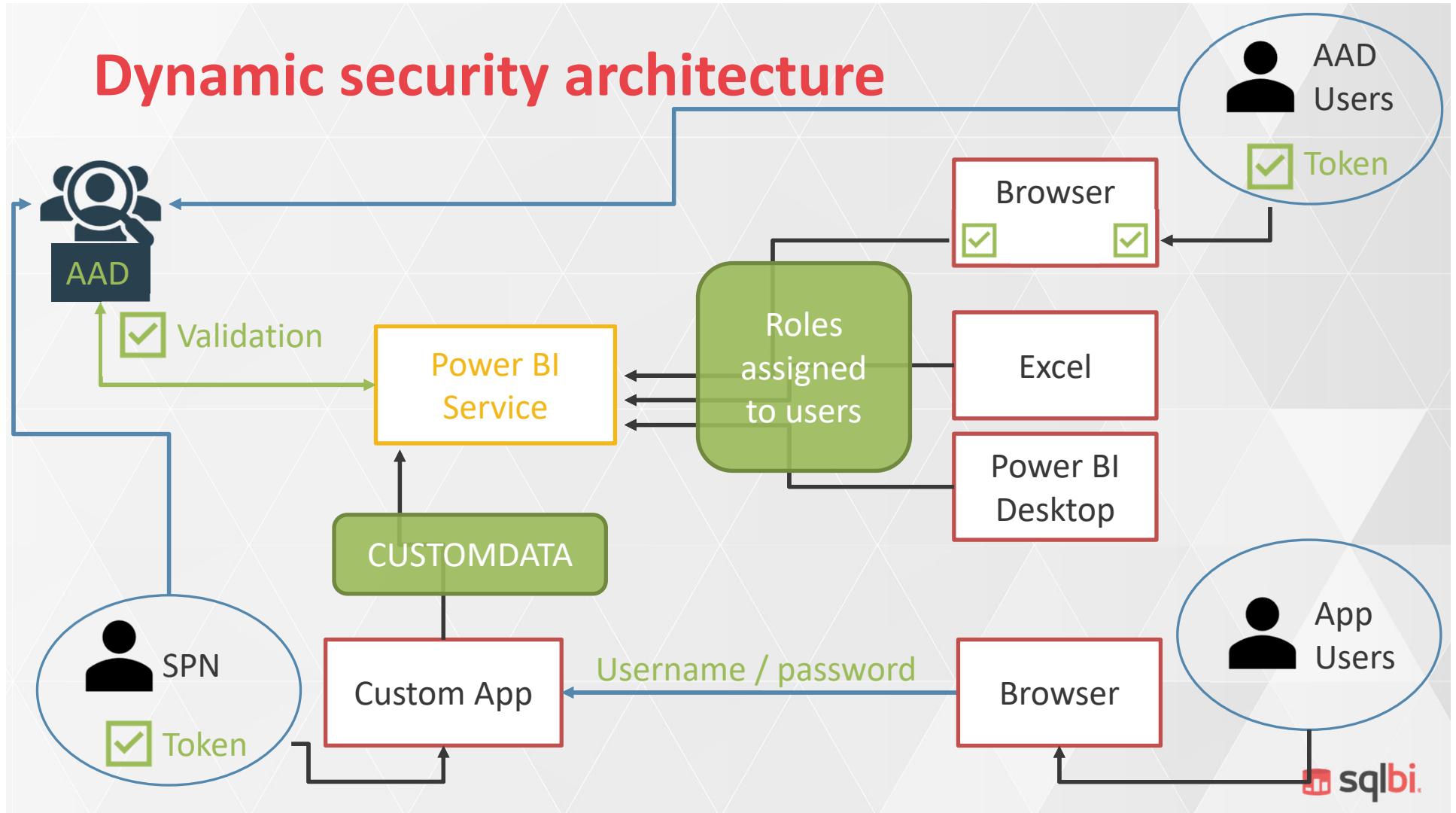
Date	Store	Amount
3/31/2020	Seattle	100,000
9/30/2020	Seattle	150,000
3/31/2021	Seattle	123,000
3/31/2020	New York	960,000
6/30/2021	New York	580,000

# Dynamic security functions

Several DAX functions are useful for dynamic security

DAX Function	Returned value
<b>USERNAME</b>	Returns the Windows account currently logged in (not the role)
<b>USERPRINCIPALNAME</b>	Returns the name of the user as their email address
<b>USEROBJECTID</b>	Returns the security identifier (SID) in Windows, another identifier in Power BI or Azure AS
<b>CUSTOMDATA</b>	Returns a string passed to the connection string in CustomData parameter. CUSTOMDATA can be used only in secured environments: users can change its content through the connection string

# Dynamic security architecture



# User-driven dynamic security

User	Store
SQLBI\Alberto	Seattle
SQLBI\Marco	New York

Date	Store	Amount
3/31/2020	Seattle	100,000
9/30/2020	Seattle	150,000
3/31/2021	Seattle	123,000
3/31/2020	New York	960,000
6/30/2021	New York	580,000

## DirectQuery over SQL security

- Row-level security
  - Row-level conditions follow the same rules as calculated columns
  - WHERE and JOIN clauses added to every SQL query
- Using Impersonation
  - Security can be enforced by SQL Server RLS using integrated security (AS user used to connect to SQL)
  - On-premises: configure delegation on Kerberos to support “Double Hop” of credentials
  - Cloud services: verify gateway configuration

## DirectQuery over AS security

- Security conditions cannot go through island boundaries
- Local model security does not affect remote models
- Remote and local models are secured independently

## Security: Conclusions

- Row-Level Security (RLS)
- Table/column-level security (OLS)
- Security is based on roles
  - RLS: union of visible rows; more rows visible
  - OLS: union of restriction rules; few objects visible
- Static security through DAX expressions
- Dynamic security using USERNAME ()
- Calculated columns/tables are computed prior to security being applied

# Understanding data types



# DAX data types

DAX Data Type	Power BI Data Type	Power Pivot and AS	SQL Server	TOM
Integer	Whole Number	Whole Number	Integer / INT	int64
Decimal	Decimal Number	Decimal Number	Floating point / DOUBLE	double
Currency	Fixed Decimal Number	Currency	Currency / MONEY	decimal
DateTime	DateTime, Date, Time	Date	Date / DATETIME	dateTime
Boolean	True/False	True/False	Boolean / BIT	boolean
String	Text	Text	String / NVARCHAR(MAX)	string
Binary	-	Binary	Blob / VARBINARY(MAX)	binary
Variant	-	-	-	variant

# Integer

- Uses math over 64-bit integers (8 bytes)
- The range is
  - -9,223,372,036,854,775,808 ( $-2^{63}$ ) to
  - 9,223,372,036,854,775,807 ( $2^{63}-1$ )

## Decimal

- Stores floating points with double precision, 8 bytes
- The SQL equivalent is *float*
- The range is
  - Negative numbers:  $-1.79^{+308}$  to  $-2.23^{-308}$
  - Positive numbers:  $2.23^{-308}$  to  $1.79^{+308}$
- The number of significant digits is limited to 15

# Currency

- Stored as 64-bit integer: the last four digits are the decimals
- The range is derived from that of Integer
  - -922,337,203,685,477.5808 to
  - 922,337,203,685,477.5807
- All math happens with 4 decimal digits
  - Summing or subtracting currencies generates currencies
  - Dividing or multiplying currencies generates decimals
- If you need more than 4 decimal digits, then currency is not the right choice. Use Decimal instead

# Conversion matrix for numeric datatypes

Addition	Integer	Decimal	Currency
Integer	Integer	Decimal	Currency
Decimal	Decimal	Decimal	Decimal
Currency	Currency	Decimal	Currency

Subtraction	Integer	Decimal	Currency
Integer	Integer	Decimal	Currency
Decimal	Decimal	Decimal	Decimal
Currency	Currency	Decimal	Currency

Product	Integer	Decimal	Currency
Integer	Integer	Decimal	Currency
Decimal	Decimal	Decimal	Currency
Currency	Currency	Currency	Decimal

Division	Integer	Decimal	Currency
Integer	Decimal	Decimal	Decimal
Decimal	Decimal	Decimal	Decimal
Currency	Currency	Currency	Decimal

# The order of evaluation matters

Even though these are borderline scenarios, you can introduce rounding errors if you do not pay attention to the order of evaluation.

```
DEFINE
    VAR Quantity = 100000                      -- Integer
    VAR Discount = 0.10                         -- Decimal
    VAR UnitPrice = CURRENCY ( 12.3456 ) -- Currency
    VAR Result1 = ( UnitPrice * ( 1 - Discount ) ) * Quantity
    VAR Result2 = UnitPrice * ( ( 1 - Discount ) * Quantity )

EVALUATE
ROW (
    "Result1", Result1,
    "Result2", Result2
)
```

Result1	1,111,100.00
Result2	1,111,104.00

# Rounding when using currency

The multiplication of the UnitPrice (Currency) by the discount (Decimal) produces a currency. This introduces a rounding error in the result, amplified by Quantity later.

DEFINE

```
VAR Quantity = 100000          -- Integer
VAR Discount = 0.10            -- Decimal
VAR UnitPrice = CURRENCY ( 12.3456 ) -- Currency
VAR Result1 = ( UnitPrice * ( 1 - Discount ) ) * Quantity
```

```
Result1 = ( CURRENCY * ( INTEGER - DECIMAL ) ) * INTEGER
```

```
Result1 = ( 12.3456 * ( 1 - 0.10 ) ) * 100000
```

```
Result1 = ( 12.3456 * ( 0.9 ) ) * 100000
```

```
Result1 = ( 11.1110 ) * 100000
```

```
Result1 = 1,111,100
```

# Currency used only at the end

If the multiplication with the currency is performed later on, the rounding error disappears.  
Need to use special care when handling currencies in the model.

DEFINE

```
VAR Quantity = 100000          -- Integer
VAR Discount = 0.10            -- Decimal
VAR UnitPrice = CURRENCY ( 12.3456 ) -- Currency
VAR Result2 = UnitPrice * ( ( 1 - Discount ) * Quantity )

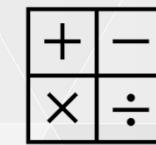
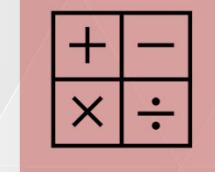
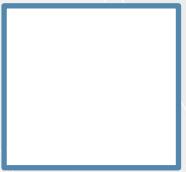
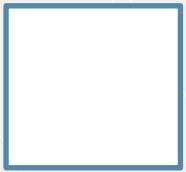
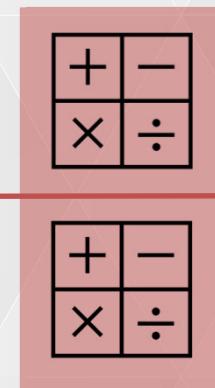
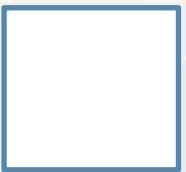
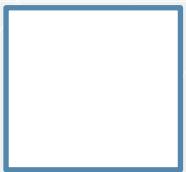
Result2 = CURRENCY * ( ( INTEGER - DECIMAL ) * INTEGER )
Result2 = 12.3456 * ( ( 1 - 0.10 ) * 100000 )
Result2 = 12.3456 * ( ( 0.9 ) * 100000 )
Result2 = 12.3456 * 90000
Result2 = 1,111,104
```

# Rounding issues with floating point

- Floating point math
- Large VertiPaq model
- Each core aggregates a number of segments
- Partial results are collected and aggregated again
- A different distribution of segments and cores produces different rounding errors
- Tiny errors, but you need to be aware of them

# Floating point rounding errors (1)

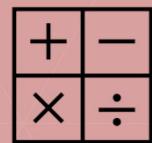
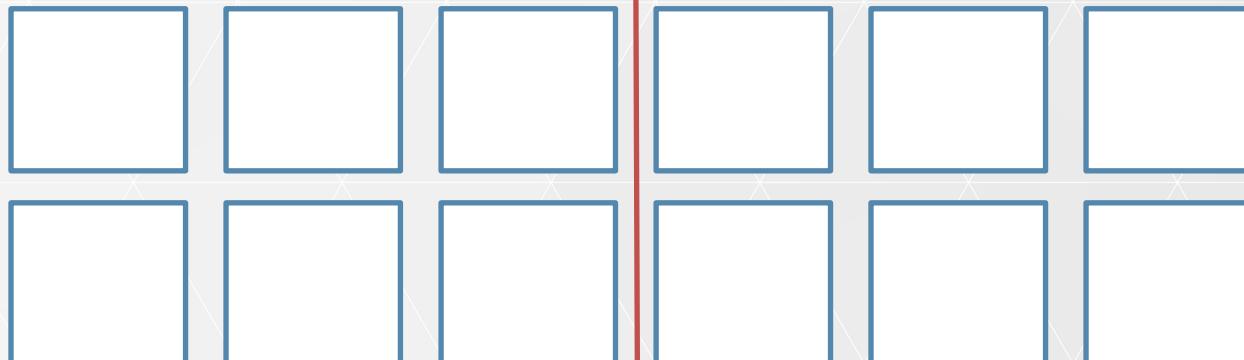
Segments



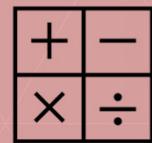
Rounded Result

## Floating point rounding errors (2)

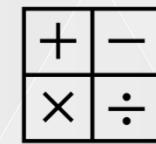
Segments



Round R1



Round R2



Rounded Result

# Warnings about numeric data types

- Currency
  - Fast calculations, internally based on integer math
  - Pay attention to divisions and multiplications
- Floating point
  - Might be slower than Currency
  - Tiny rounding errors in large models (for parallelism)
- When in doubt, check the operator and conversion matrix

## Date**Time**

- **DateTime** is a floating point
- Integer part is number of days since 12/30/1899
- Decimal part is the fraction of the day
- Dates supported from 03/01/1900 to 12/31/9999

```
1 EVALUATE
2 {
3     FORMAT (
4         4.10,
5         "mm/dd/yyyy hh:mm:ss"
6     )
7 }
```

197 %

Results

Value
01.03.1900 02:24:00

12/30/1899 + 4 days = **01/03/1900**

0.1 = 10% of 24 hours = 2.4 hours  
2 hours and 0.4x60 mins = **2:24**

# The leap year bug

Excel inherited a bug originally introduced in Lotus 1-2-3.

Excel considers 1900 a leap year, even though it was not a leap year.

This is the reason DAX starts a day earlier than Excel.

```
EVALUATE  
ADDCOLUMNS (  
    { 1, 32, 59, 60, 61 },  
    "AsDateTime", FORMAT (  
        [Value],  
        "mm/dd/yyyy"  
    )
```

## DAX

Value	AsDateTime
1	12.31.1899
32	01.31.1900
59	02.27.1900
60	02.28.1900
61	03.01.1900

## Excel

Value	AsDateTime
1	January 1, 1900
32	February 1, 1900
59	February 28, 1900
60	February 29, 1900
61	March 1, 1900

# Boolean

- Represents a condition: either TRUE or FALSE:
  - 0 = FALSE
  - 1 = TRUE
- TRUE > FALSE in sorting
- Automatic conversion to integer in expressions:
  - TRUE \* [measure] = <measure value>
  - FALSE \* [measure] = 0

# String

- Unicode representation of a string
- Each character is stored in 16 bits
- By default, comparison and sorting are case-insensitive
  - “Power BI” == “POWER BI”
  - “alberto” < “Marco”
- Pay attention to case sensitivity in relationships
  - If case sensitivity is required, use a surrogate integer key

# Binary

- Binary is a datatype used to contain unstructured data
  - Pictures
  - Files
  - Any binary data
- Seldom used, it was mainly used in Power View
- Binary is no longer available in Power BI

# Variant

Used to represent expressions that might have different data types.

Used only for measures. A column cannot be of Variant datatype.

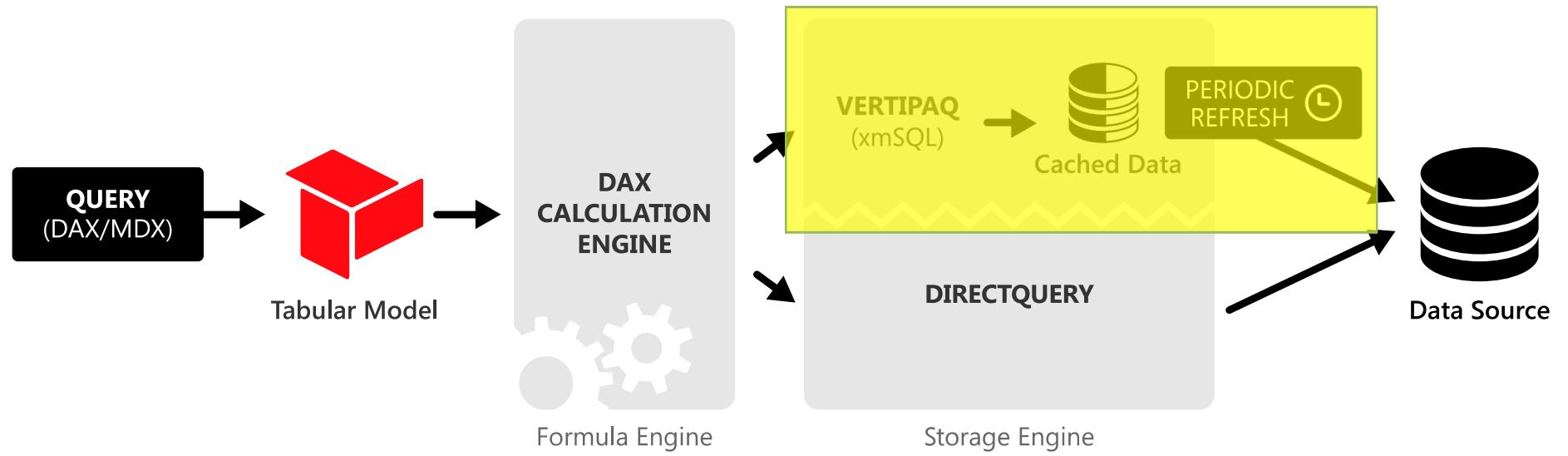
```
EVALUATE
{
    --
    -- The IF expression is of type VARIANT
    --
    -- VARIANT is valid for measures only. Columns are not supported.
    --
    IF (
        [AnyCondition],
        10,                      -- INTEGER
        "Test"                   -- STRING
    )
}
```

Deep dive into the VertiPaq storage engine

# Understanding the VertiPaq engine



# Tabular query architecture



# What is VertiPaq?

- Data is stored in the VertiPaq database
  - In-memory database
  - Based on the relational methodology
  - Column-oriented database
- Very different from a regular database, like SQL
- Understanding how it works is important to make educated choices on the model

# Row storage layout

Customers table

ID	Name	Address	City	State	Bal Due
1	Bob	...	...	NY	3,000
2	Sue	...	...	NY	500
3	Ann	...	...	WA	1,700
4	Jim	...	...	WA	1,500
5	Liz	...	...	WA	0
6	Dave	...	...	WA	9,000
7	Sue	...	...	NY	1,010
8	Bob	...	...	WA	50
9	Jim	...	...	NY	1,300

1	Bob	...	...	NY	3,000
2	Sue	...	...	NY	500
3	Ann	...	...	WA	1,700
4	Jim	...	...	WA	1,500
5	Liz	...	...	WA	0
6	Dave	...	...	WA	9,000
7	Sue	...	...	NY	1,010
8	Bob	...	...	WA	50
9	Jim	...	...	NY	1,300

Nothing special here.

This is the standard way database systems have been laying out tables on disk since the mid 1970s.

Technically, it is called a “row store”.

# Column storage layout

Customers table

ID	Name	Address	City	State	Bal Due
1	Bob	...	...	NY	3,000
2	Sue	...	...	NY	500
3	Ann	...	...	WA	1,700
4	Jim	...	...	WA	1,500
5	Liz	...	...	WA	0
6	Dave	...	...	WA	9,000
7	Sue	...	...	NY	1,010
8	Bob	...	...	WA	50
9	Jim	...	...	NY	1,300

ID	Name	Address	City	State	Bal Due
1	Bob	...	...	NY	3,000
2	Sue	...	...	NY	500
3	Ann	...	...	WA	1,700
4	Jim	...	...	WA	1,500
5	Liz	...	...	WA	0
6	Dave	...	...	WA	9,000
7	Sue	...	...	NY	1,010
8	Bob	...	...	WA	50
9	Jim	...	...	NY	1,300

Tables are stored column-by-column with all values from a single column stored in a single block

# Querying a columnar database

ID	Name	Address	State	Gender	Bal Due
1	Bob	...	NY	Male	3,000
2	Sue	...	NY	Female	500
3	Ann	...	WA	Female	1,700
4	Jim	...	WA	Male	1,500
5	Liz	...	WA	Female	0
6	Dave	...	WA	Male	9,000
7	Sue	...	NY	Female	1,010
8	Bob	...	WA	Male	50
9	Jim	...	NY	Male	1,300

# Column vs row storage

- Column Storage
  - Quick access to a single column
  - Time needed to materialize rows
  - Trade CPU vs I/O
- Row Storage
  - Quick access to a single row
  - No materialization needed
  - Trade I/O vs CPU
- From the point of view of the CPU, RAM access is I/O

# Value encoding



# Hash encoding

Quarter
Q1
Q1
Q1
Q1
Q2
Q2
...
Q2
Q3
Q3
Q3
Q3
Q4
Q4
Q4
...

HASH

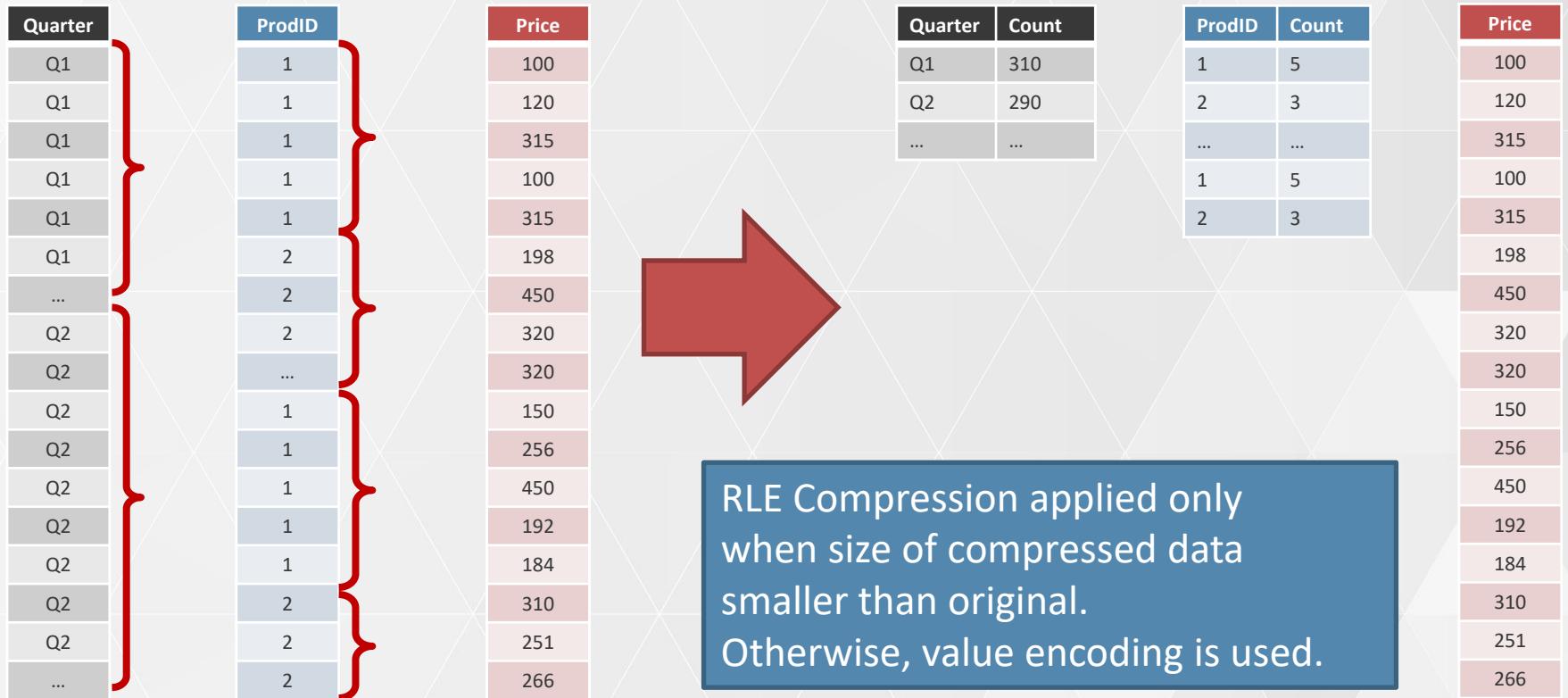
Dictionary

Q.ID	Quarter
0	Q1
1	Q2
2	Q3
3	Q4

Q.ID
0
0
0
0
1
1
...
1
2
2
2
2
3
3
3
3
...

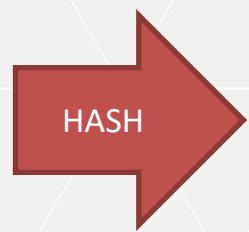
Only 4 values.  
2 bits are enough to  
represent it.

# Run Length Encoding (RLE)



# Compression of one column

Quarter
Q1
Q1
Q1
Q1
Q2
Q2
...
Q2
Q3
Q3
Q3
Q3
Q4
Q4
Q4
Q4
...



Q.ID
0
0
0
0
1
1
...
1
2
2
2
2
3
3
3
3
...

Q.ID	Quarter
0	Q1
1	Q2
2	Q3
3	Q4



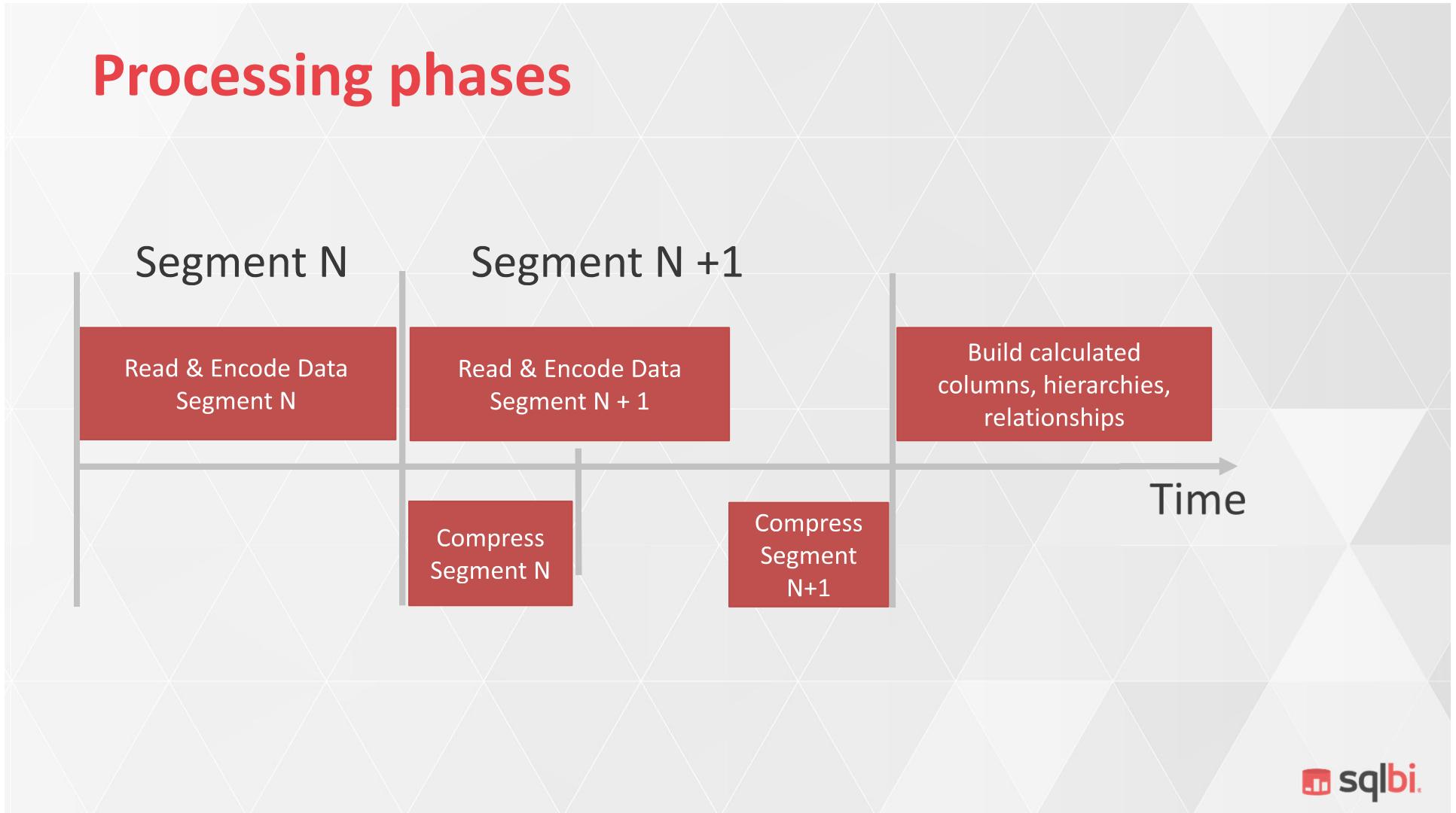
VertiPaq Store

Quarter	Count
0	4
1	10
2	4
3	15
2	30
1	15

# Segmentation

- Each table is divided into segments
  - 8 million rows for each segment in AS (configurable)
  - 1 million rows in PowerPivot and Power BI (fixed)
- Dictionary is global to the table
- Bit-sizing is local to the segment
  - Column «Date» uses 4 bits in segment 1
  - Only 2 bits in segment 2
  - ...
- DMV available to query that info

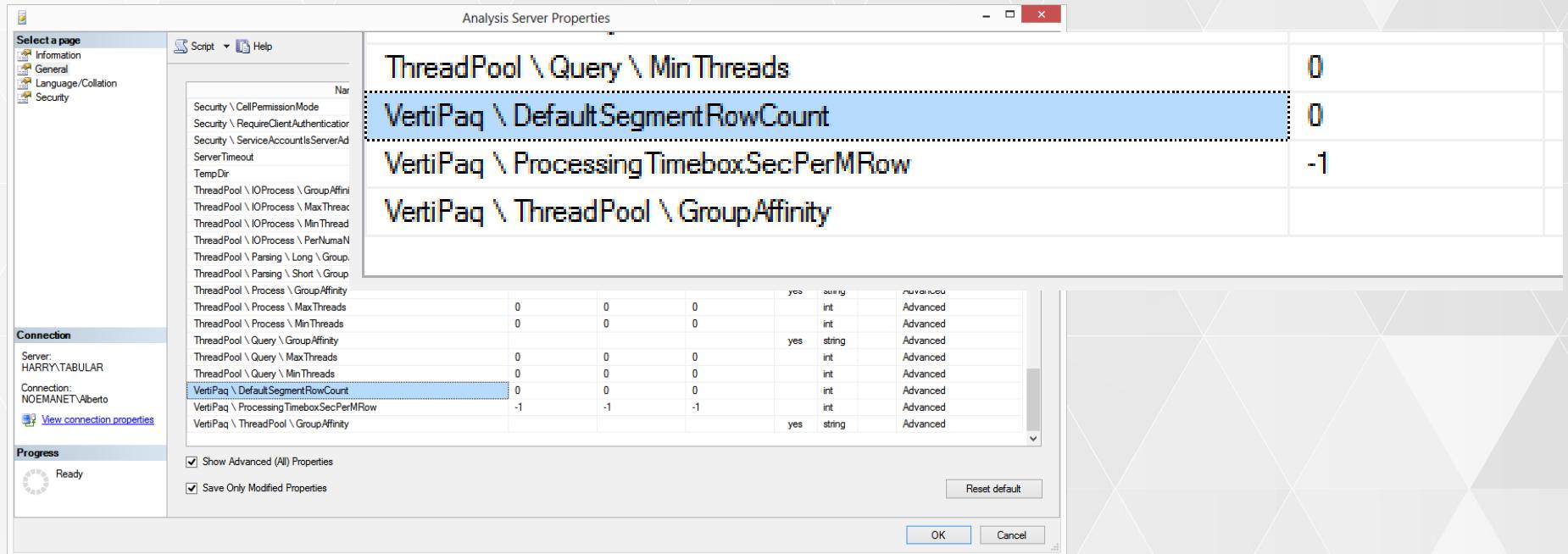
# Processing phases



# Calculated columns and tables

- Calculated columns are compressed
  - But they do not participate in the sort selection
  - Consequently, they might be larger than imported columns
- Calculated tables are compressed
  - They need to be computed, entirely in RAM
  - The process requires more RAM compared to an imported table with the same content

# Segment configuration



- Configured at the instance level on Analysis Services
- Saved at the database level after processing

## Segment size and compression

- DefaultSegmentRowCount
  - Default: 8M
  - Need to be power of 2, at least 1M
  - Larger: better compression, faster queries
  - Smaller: smaller working set during processing
- ProcessingTimeboxSecPerMRow
  - Default: -1 = 10 sec
  - Smaller: lower compression
  - Larger: better compression, better query performance
  - Increase for large number of columns (~>200)

## Segment size and partitions

- Segments cannot be larger than partitions
- Do not create too many partitions
- Partitions optimize data load, not query execution

General rule: the partition size should be a multiple of the segment size – the larger the multiple, the better.

# Hierarchies

- User hierarchies
  - Hierarchies created in the project to simplify navigation
- Attribute hierarchies
  - Internal data structures to improve performance of MDX queries and DAX filters
  - Building hierarchies is an expensive operation
  - Mandatory if the column is queried through MDX
  - Avoid attribute hierarchies for columns not used in filters and sort order

## Data memory usage

- Memory usage depends on
  - Number of columns
  - Cardinality of each column
  - Data type
  - Number of rows
- Strings
  - Average size is relevant for dictionary size
- No easy formula can be applied

# Available DMVs

You can use DMVs to query the server and discover the size of each object

```
--  
-- Discover statistics of tables  
--  
  
select * from  
    $system. discover_storage_tables  
  
--  
-- Discover details of individual columns  
--  
  
select * from  
    $system.discover_storage_table_columns
```

<https://docs.microsoft.com/en-us/analysis-services/instances/use-dynamic-management-views-dmvs-to-monitor-analysis-services>

## VertiPaq Analyzer

- Available at [www.sqlbi.com/tools/vertipaq-analyzer/](http://www.sqlbi.com/tools/vertipaq-analyzer/)
- Very detailed view of the internal structure of a single database
- DAX Studio reports a subset of the information of VertiPaq Analyzer: the most relevant



**VertiPaq  
Analyzer**

## VertiPaq best practices

- Reduce column cardinality
- Optimize size of relationships
  - Reduce cardinality of the 1-side of the relationship
- Focus on
  - Large cardinality columns
  - Large dimensions

## Reduce cardinality

- Reduce number of distinct values
  - DateTime: split in two columns
    - Date
    - Time
  - Floating Point Values: fix precision
    - 10.231 → 10.2
- Avoid partial results in calculated columns
- Avoid junk dimensions

# What to store in the Model?

OrderId	ProductId	Quantity	Price	Discount	SalesAmount
1	12	10	2.55	1.5	24.00
2	12	8	2.55	0.4	20.00
...	...	...	...	...	...
...	...	...	...	...	...
847	12	9	2.55	0.95	22.00

These columns are  
needed

Small number of Distinct  
Values

Higher Cardinality

## VertiPaq: conclusions

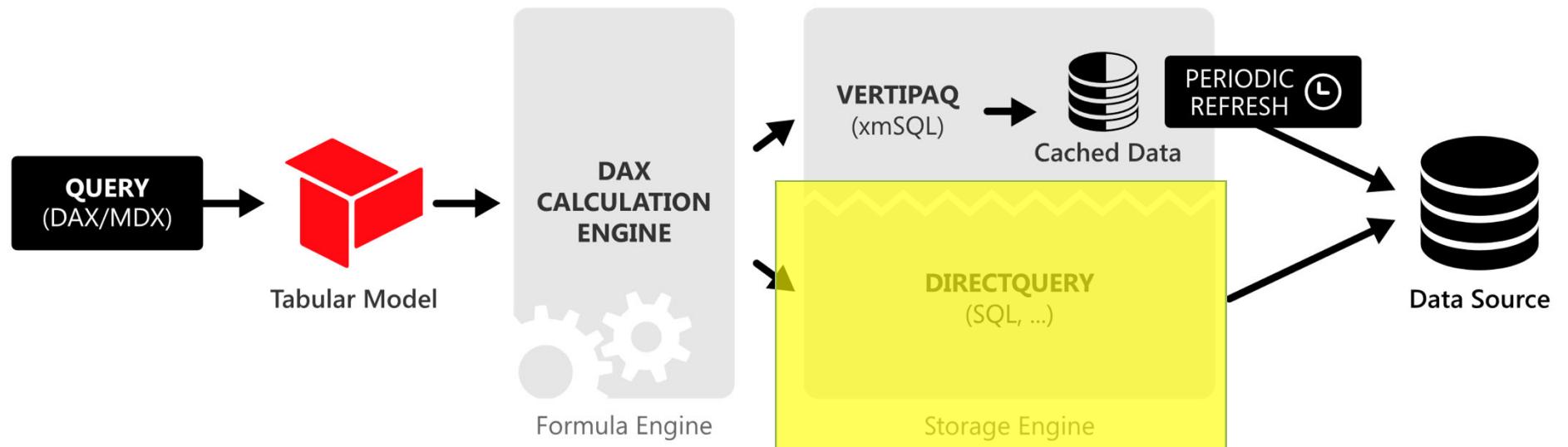
- Very efficient compression algorithm
- Very fast on single column scans
- Multi-column access requires materialization
  - Storage very different from classic databases
  - Requires shift in mindset

Deep dive into the DirectQuery Storage Engine

# Understanding DirectQuery



# DirectQuery storage engine



## Advantages of DirectQuery

- No processing time
  - Queries executed when needed
  - Semantic model contains only metadata
- Real-time queries
  - Information always up to date
- No transfer of data from data source to AS engine
  - Large datasets take time to move from the data source to the AS engine

# VertiPaq versus DirectQuery

- VertiPaq is a database
  - Data is compressed
  - Data needs to be in-memory
  - Some ETL steps done in DAX
- DirectQuery is a query technology
  - Data is stored in data source
  - Data is on-disk
  - Cannot perform ETL steps
  - No latency (but slower queries)

# DirectQuery requirements

- Legacy datasources
  - Only data sources supporting DirectQuery
- Power Query datasources
  - Query folding required
  - M transformation allowed, if the result is being folded
  - At the end, AS generates a query (usually SQL) to gather values to return to the formula engine

# Enabling DirectQuery

- Each table can be
  - Import
  - DirectQuery
  - Dual mode
- Therefore, a single model can use multiple storage engines
- If the same model contains different table types, it is a **composite model**

# Using DAX in DirectQuery

- Optimized functions, depending on data source
  - Functions that are translated straight into SQL
  - Examples: DIVIDE, ISBLANK, ROUND...
- Optimized but cannot be used in calculated columns
  - Translated to SQL, but work on tables
  - Examples: SUM, ALL, CALCULATE, FILTER
- Non-optimized functions
  - Computed by the Formula Engine only
  - Based on raw data retrieved by DirectQuery
  - Examples: TOPN, GENERATE

# Optimized function example

SUM can be executed straight in SQL; the query is optimized.

```
EVALUATE  
SUMMARIZECOLUMNS (  
    'Product'[Brand],  
    "Sales", SUMX ( Sales, Sales[Quantity] )  
)
```

```
SELECT  
    Product.Brand,  
    SUM ( Sales.Quantity )  
FROM  
    Sales  
    INNER JOIN Product  
    ON Sales.ProductKey = Product.ProductKey  
GROUP BY  
    Product.Brand
```

# Non-optimized function example

MEDIANX cannot be evaluated in SQL. DAX needs to retrieve the raw values and compute the MEDIAN in the formula engine. This might possibly generate a large materialization.

```
EVALUATE  
SUMMARIZECOLUMNS (  
    'Product'[Brand],  
    "Sales", MEDIANX ( Sales, Sales[Quantity] )  
)
```

```
SELECT  
    Product.Brand,  
    Sales.Quantity  
FROM  
    Sales  
    INNER JOIN Product  
    ON Sales.ProductKey = Product.ProductKey
```

# Query limits

Huge materializations are not supported; you can configure the *MaxIntermediateRowsetSize* setting (Analysis Services – capacity setting in PBI Premium), but you need to be careful when doing so.

```
EVALUATE  
ROW (  
    "orders",  
    COUNTROWS ( ALL ( Sales[Order Number], Sales[Order Line Number] ) )  
)
```

--Translates to this query and it is likely to fail on large datasets:

```
SELECT TOP ( 1000001 )  
    [t1].[Order Number],  
    [t1].[Order Line Number]  
FROM    ( SELECT * FROM Sales ) AS [t1]  
GROUP BY [t1].[Order Number], [t1].[Order Line Number];
```

# Calculated columns

In calculated columns you should only use optimized functions that work on a single row. Moreover, they are computed by the data source at every single table scan at query time.

```
--  
-- This calculated column works fine  
--  
Sales[Amount] = Sales[Quantity] * Sales[Unit Price]  
  
--  
-- This calculated column cannot be defined in DirectQuery  
--  
Products[TotalSales] =  
SUMX (  
    RELATEDTABLE ( Sales ),  
    Sales[Quantity] * Sales[Unit Price]  
)
```

## Row-level security

- Security expressions need to follow the same restrictions as calculated columns
- Only simple, optimized DAX functions
- WHERE and JOIN conditions injected into every SQL query to ensure security constraints

## Semantic differences in DAX

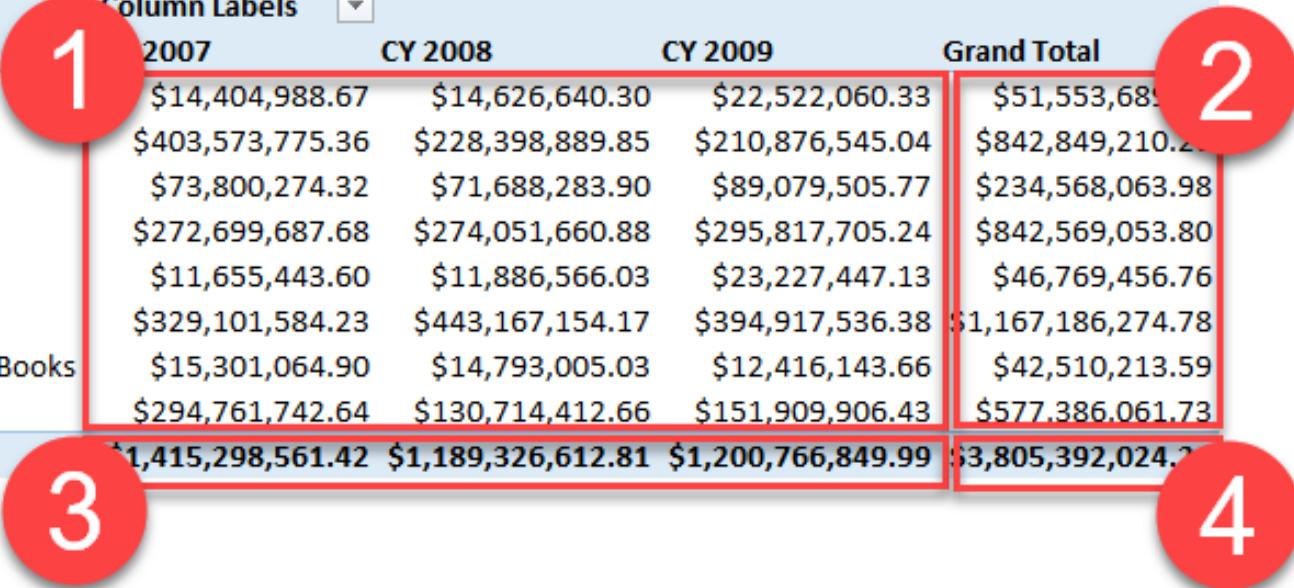
- SQL semantics differ from DAX semantics
- A few examples:
  - 1/BLANK is fine in DAX, error in SQL
  - Casting errors might appear
  - Numerical rounding and/or overflow

# Using MDX in DirectQuery

- MDX generated by Excel
  - Analyze in Excel with Power BI
- Limitations
  - **No user-defined hierarchies** (not displayed in Excel)
  - No relative object names
  - No session-scope MDX statements
  - No tuple with member from different levels in sub-selects
  - Detailed list of MDX limitations available at  
<https://docs.microsoft.com/en-us/sql/analysis-services/tabular-models/directquery-mode-ssas-tabular>

## Real-time and coherence of values

Sales Amount	Column Labels			
Row Labels	2007	CY 2008	CY 2009	Grand Total
Audio	\$14,404,988.67	\$14,626,640.30	\$22,522,060.33	\$51,553,689.30
Cameras and camcorders	\$403,573,775.36	\$228,398,889.85	\$210,876,545.04	\$842,849,210.25
Cell phones	\$73,800,274.32	\$71,688,283.90	\$89,079,505.77	\$234,568,063.98
Computers	\$272,699,687.68	\$274,051,660.88	\$295,817,705.24	\$842,569,053.80
Games and Toys	\$11,655,443.60	\$11,886,566.03	\$23,227,447.13	\$46,769,456.76
Home Appliances	\$329,101,584.23	\$443,167,154.17	\$394,917,536.38	\$1,167,186,274.78
Music, Movies and Audio Books	\$15,301,064.90	\$14,793,005.03	\$12,416,143.66	\$42,510,213.59
TV and Video	\$294,761,742.64	\$130,714,412.66	\$151,909,906.43	\$577,386,061.73
Grand Total	\$1,415,298,561.42	\$1,189,326,612.81	\$1,200,766,849.99	\$3,805,392,024.22



4 SQL queries, results might not match because of updates taking place in the meantime

# Real-time and DAX

Results might also be confusing for a single DAX expression.

Many queries are executed for each DAX statement, no matter how simple it is.

EVALUATE

```
SUMMARIZE (
    Sales,
    ROLLUP ( Product[Color] ),
    "Amt", SUM ( Sales[LineTotal] )
)
```

## DirectQuery scenarios

- Real-time reporting
- No processing window
- Not enough memory for the model
- You need to create a semantic layer on a relational database

## Real-time modeling

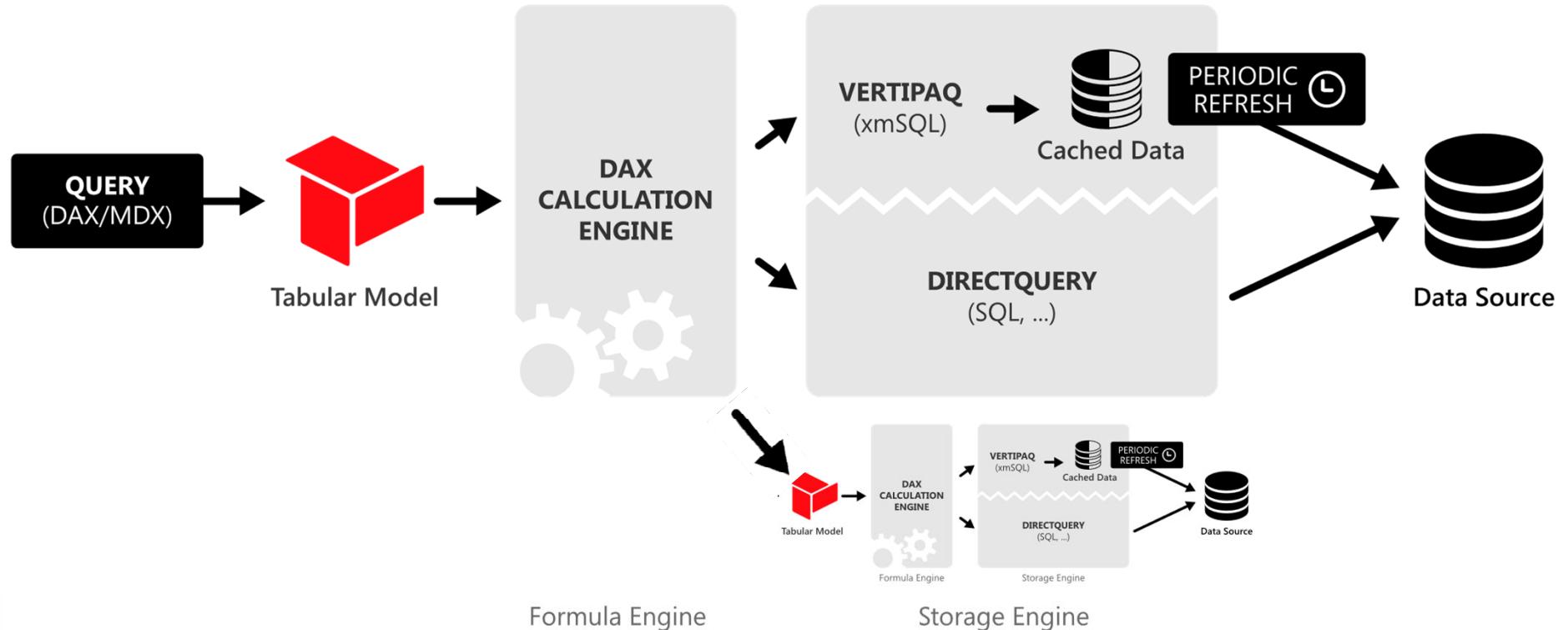
- Do you really need real-time on the entire data model?
  - Many limitations might create issues in the model
  - Large datasets are hard to optimize
  - Real-time data may be useful for the last few days, with simpler calculations
- Build two models
  - One VertiPaq model for historical data
  - One DirectQuery model for real-time data
- Alternatively: build a push model (covered later)

Deep dive in the DirectQuery storage engine for Analysis Services

# Understanding DirectQuery over Analysis Services



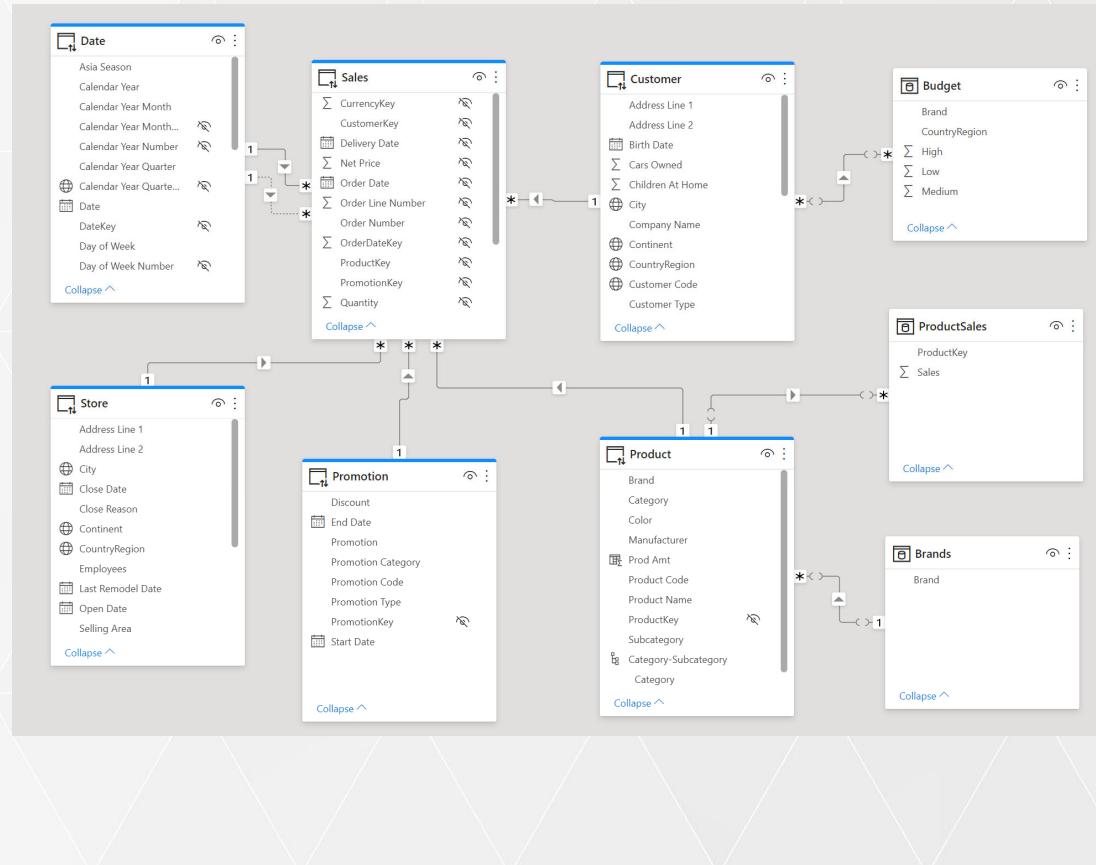
# DirectQuery over Analysis Services



## DirectQuery over AS: pros

- Extend the corporate model with new data
- Re-use the entire semantic model
- Merge different models into a single view
- The storage engine shares the full power of DAX

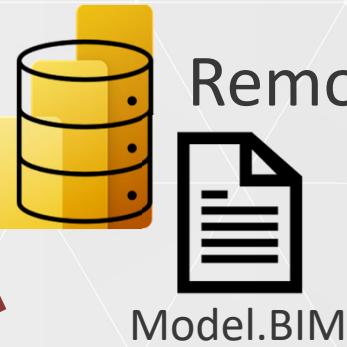
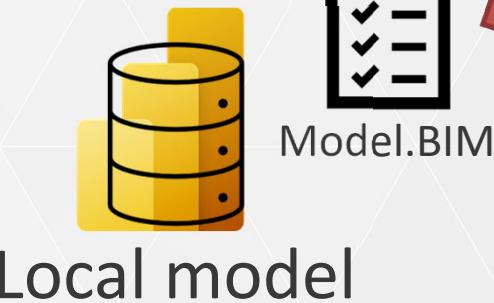
# The local model extends the remote model



- Local tables extend the remote model
- You can build relationships, calculated tables, calculated columns and measures

# Composite models architecture for Model.BIM

The local model.bim contains the full model description

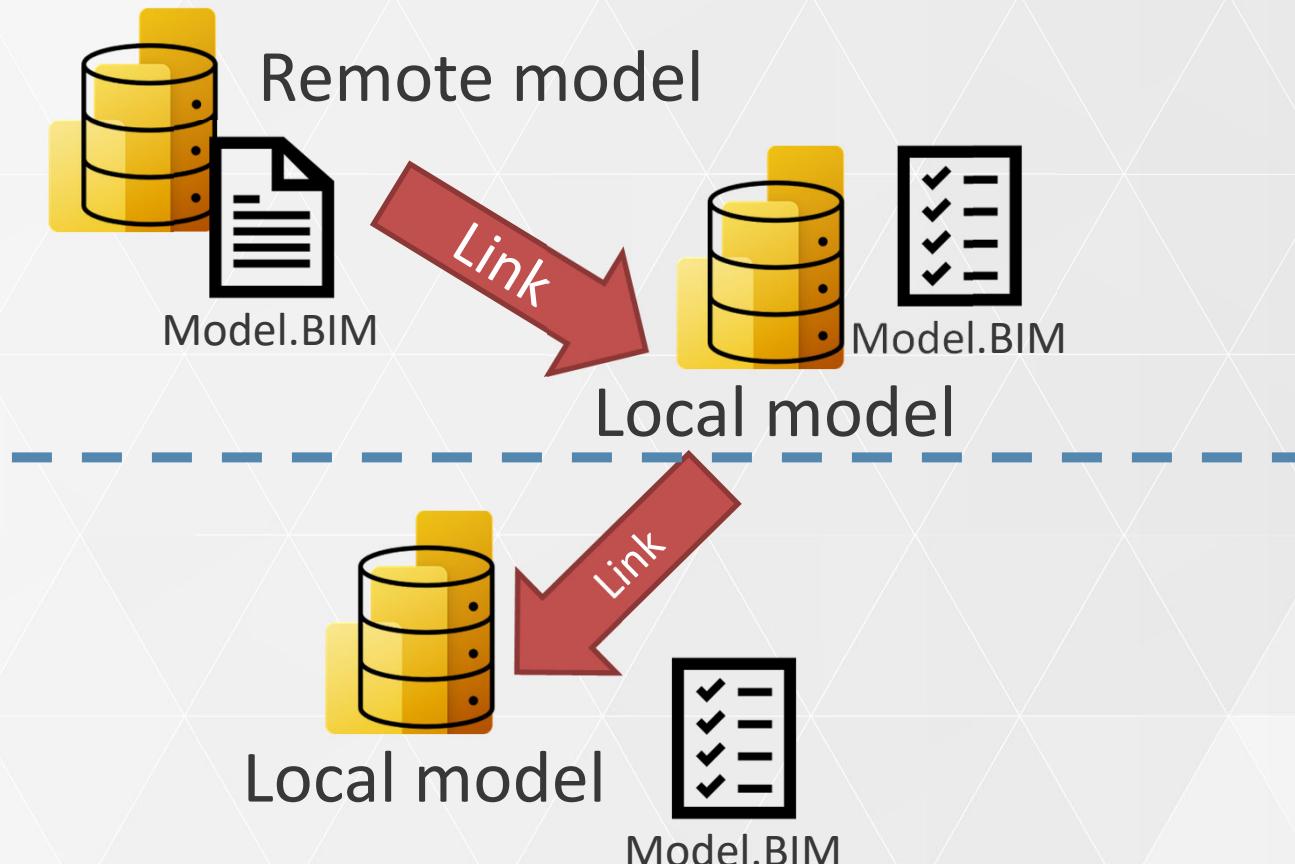


No data is copied.  
Only metadata links are used.

## Contents of local model.bim

- The local model.bim contains:
  - Tables and objects linked from the remote model(s)
  - Measures linked as EXTERNALMEASURE
  - Local object definitions
- Each remote object linked in the local mode contains:
  - Properties linked from the source model
  - Properties overridden from the source model
- For example: you **can** rename tables and columns
  - *Sales* → local *Orders*
  - *Sales[CustomerKey]* → local *Orders[CustomerCode]*

## Chaining models



# DirectQuery: SQL vs Analysis Services

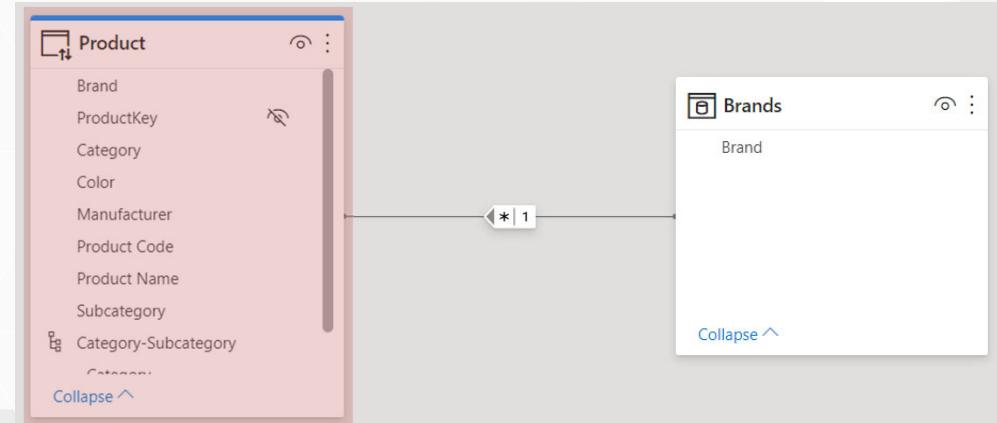
- DirectQuery over SQL
  - The model is entirely local
  - Limitations in the DAX formulas
  - The remote server runs SQL
- DirectQuery over Analysis Services
  - The model is partially remote and partially local
  - Few limitations on DAX and modeling options
  - The remote server runs AS
  - Multiple options to solve a query

# Remote model query resolution

A query that involves tables from the remote model can be pushed to the remote model, so that it is computed entirely in the remote Analysis Services.

```
EVALUATE  
SUMMARIZECOLUMNS (  
    'Product'[Brand],  
    "#Prods", COUNTROWS ( 'Product' )  
)
```

The entire query is sent to the remote server; no processing on the local AS

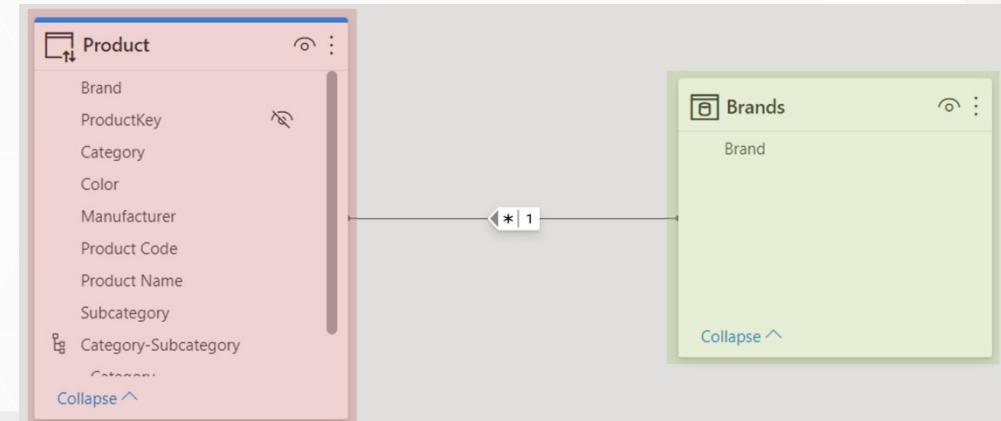


# Cross-island query resolution

A query that involves tables from different islands requires more steps. Brands is evaluated locally, then sent to the server to filter Product.

```
EVALUATE  
SUMMARIZECOLUMNS (  
    Brands[Brand],  
    "#Prods", COUNTROWS ( 'Product' )  
)
```

The query involves both the local  
and the remote engines



# Cross-island query: the steps

AS first retrieves the values from the local Brands table.

Then, it uses these values to execute a query on the remote AS.

```
EVALUATE  
SUMMARIZECOLUMNS (  
    Brands[Brand],  
    "#Prods", COUNTROWS ( 'Product' )  
)
```

XMSQL Query  
SELECT Brands[Brand] FROM Brands;

Query speed strongly depends on the size of the table passed on to the remote AS

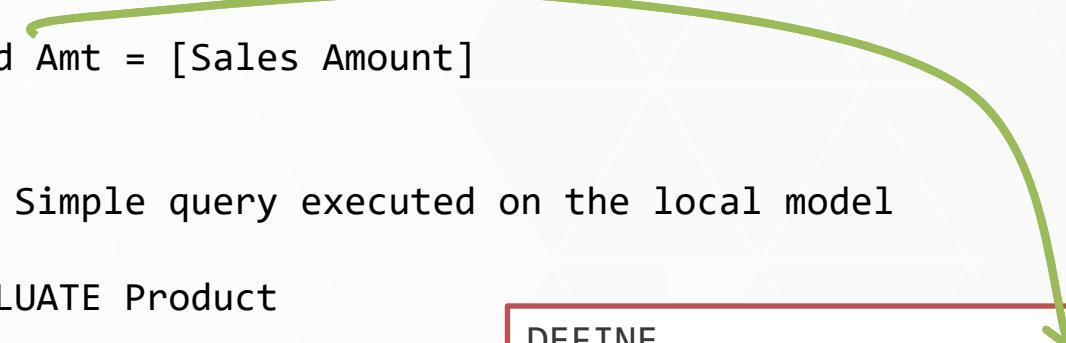
```
DEFINE  
VAR _Var1 = {  
    ( 3, "Contoso" ),  
    ( 4, "Wide World Importers" ),  
    ( 5, "Northwind Traders" ),  
    ( 6, "Adventure Works" ),  
    ( 7, "Southridge Video" ),  
    ( 8, "Litware" ),  
    ( 9, "Fabrikam" ),  
    ( 10, "Proseware" ),  
    ( 11, "A. Datum" ),  
    ( 12, "The Phone Company" ),  
    ( 13, "Tailspin Toys" )  
}  
EVALUATE  
GROUPCROSSAPPLYTABLE ( ... )
```

# Calculated columns

A locally-created calculated column stored in a remote table needs to be pushed to the remote server to be computed at query time.

```
--  
-- Calculated column in Product (Remote table)  
--  
Prod Amt = [Sales Amount]  
  
--  
-- Simple query executed on the local model  
--  
EVALUATE Product
```

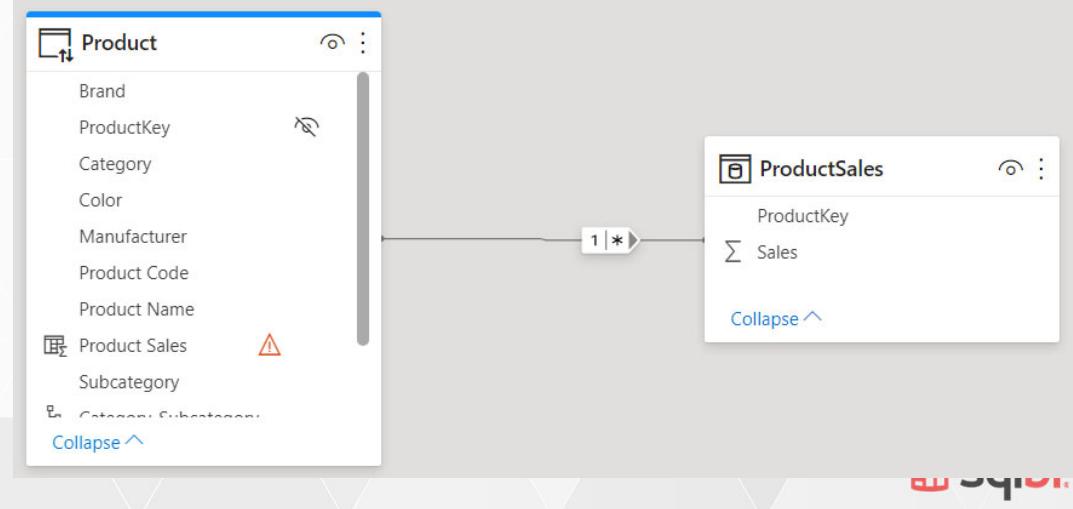
```
DEFINE  
    COLUMN 'Product'[ASDQ_Prod Amt] = [Sales Amount]  
EVALUATE  
    Product
```



# Calculated columns limitations

You cannot create a remote calculated column if its formula cannot be pushed to the remote server.  
In this example, the formula cannot be computed only on the remote server.

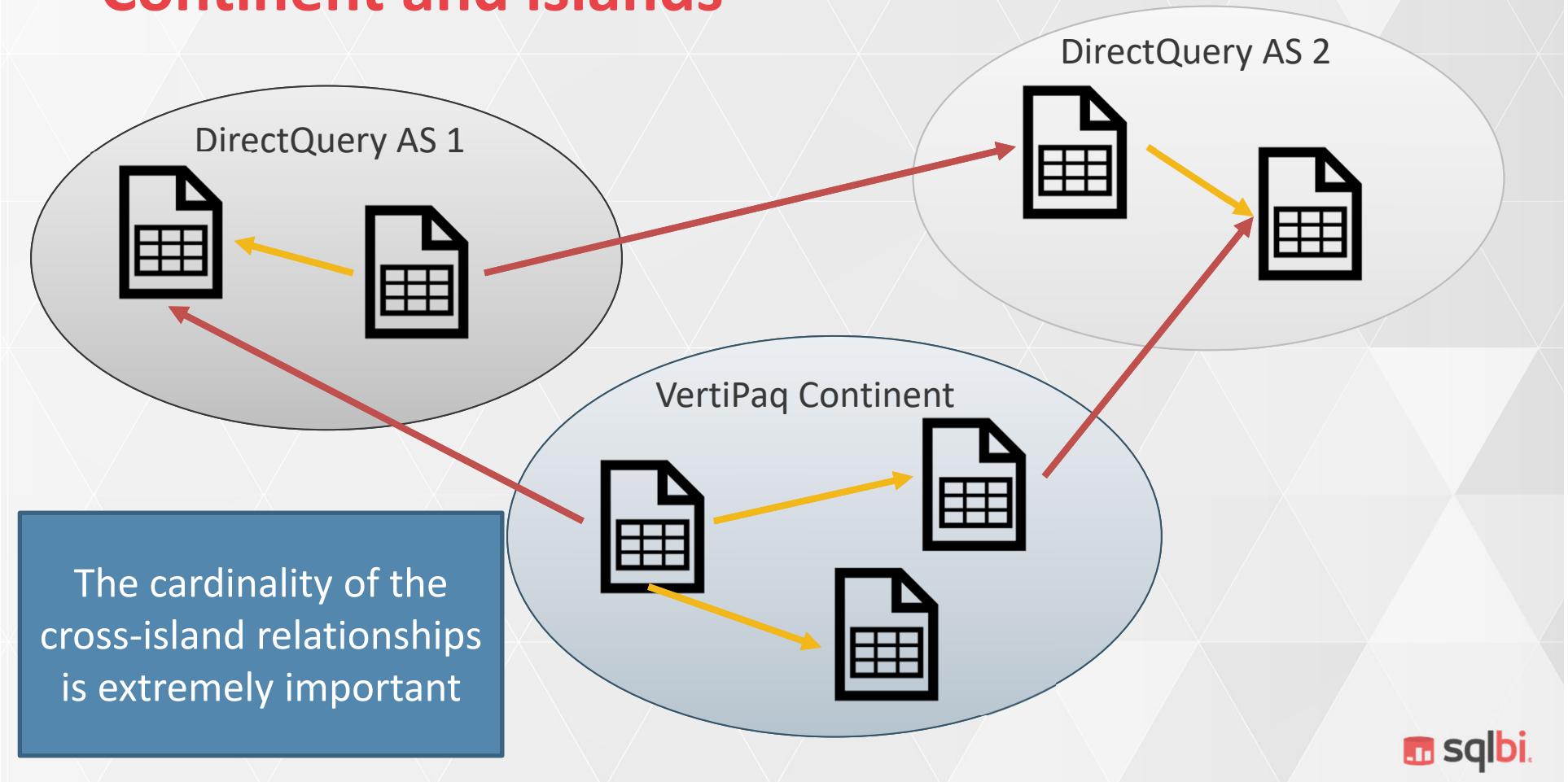
```
--  
-- Calculated column in Product (Remote table)  
  
Product Sales =  
CALCULATE (  
    SUM ( ProductSales[Sales] ),  
    ALLEXCEPT (  
        'Product',  
        'Product'[ProductKey]  
    )  
)
```



## DirectQuery over AS: limitations

- Cannot alter/create remote relationships
- One local table cannot filter two or more remote tables
- One remote table cannot filter another remote table through a local table
- Security filters do not cross the island boundaries
- Chaining can be limited in number of steps

# Continent and islands



# Designing measures in composite models

- The main requirements are:
  - Small-sized relationships between islands
  - Reduce data transfer between islands
- On the model, this becomes the following requirements:
  - The model needs to be designed carefully
  - DAX code requires dedicated optimization techniques
  - The optimizations depend on the model structure
  - The placement of tables in islands is relevant

## DirectQuery over AS security

- Security conditions cannot traverse island boundaries
- Local model security does not affect remote models
- Remote and local models are secured independently
- A published Power BI dataset can choose to disable the option of using it as a remote model for DirectQuery

## Scenarios for DirectQuery over AS

- Create a layer over an existing model
  - Renaming entities according to business requirements
  - Adding specific measures
- Import local data combined with Enterprise model
  - Extend an enterprise model with small – local datasets
- **Combining multiple models? Be careful!**
  - Combine two or more enterprise models into a single one
  - Sales, Accounting, Budgeting...

Mixing DirectQuery over SQL and VertiPaq tables in the same model

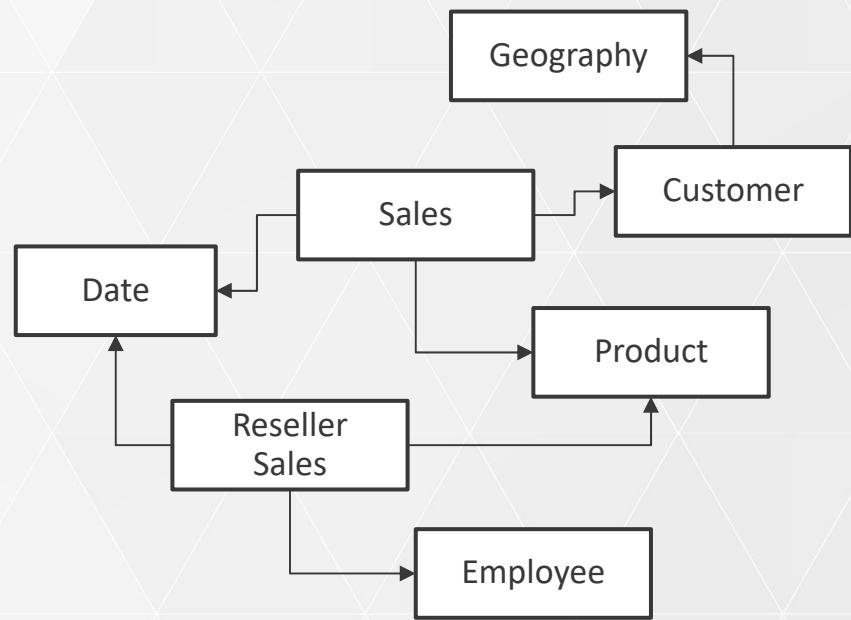
# Aggregations



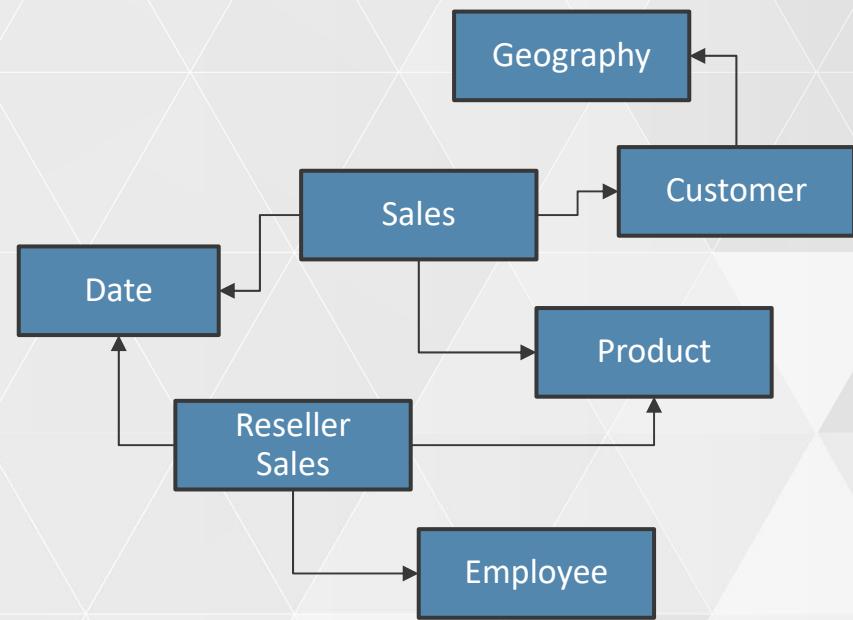
## Aggregations are alternate of model columns

- A column in an aggregation table is
  - Alternate of another column
  - Using an aggregation function
- Aggregation functions available
  - GROUPBY
  - COUNT, MAX, MIN, SUM, COUNTROWS
- Aggregation table **must** be hidden because of security

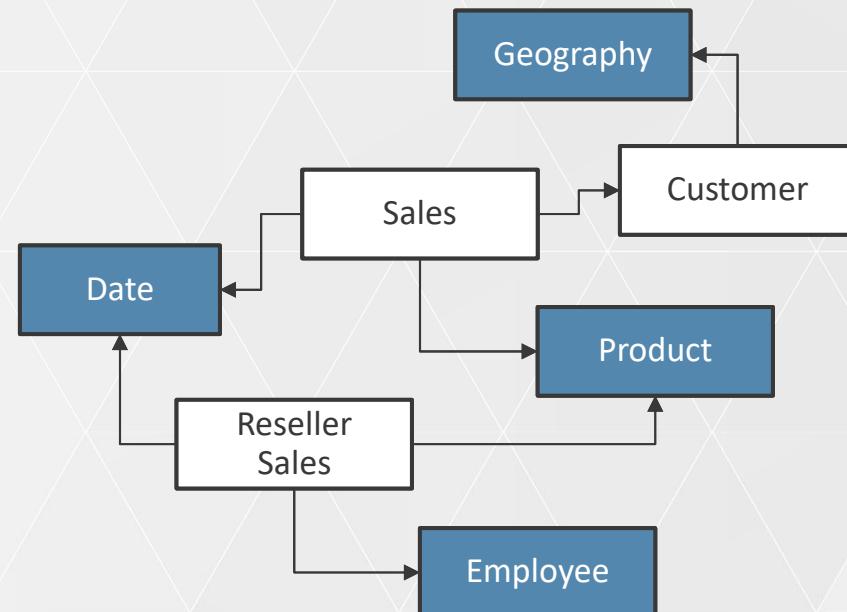
# DirectQuery



# Import



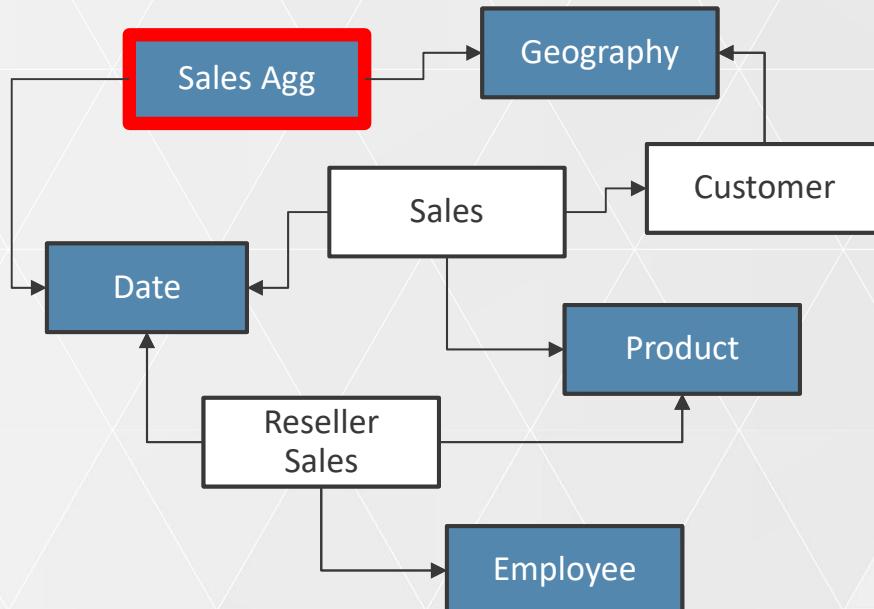
# DirectQuery & Import (Composite models)



Import

DirectQuery

# Creating aggregations

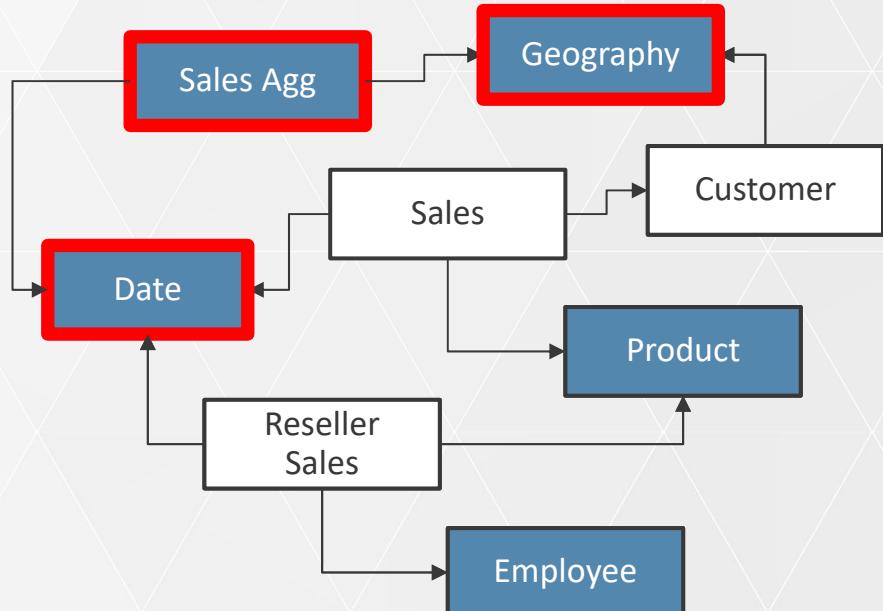


Import

DirectQuery

sq|bi.

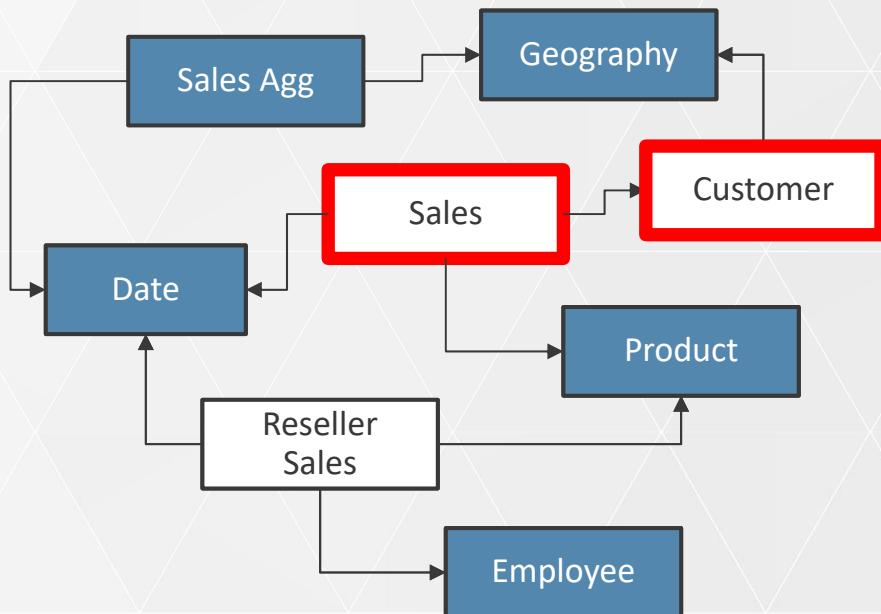
# Querying import tables



```
SUMMARIZECOLUMNS (  
    'Date'[Year],  
    Geography[City],  
    "Quantity", SUM ( Sales[Quantity] )  
)
```

Hits in-memory cache

# Querying DirectQuery tables



```
SUMMARIZECOLUMNS (  
    Customer[Name],  
    "Quantity", SUM ( Sales[Quantity] )  
)
```

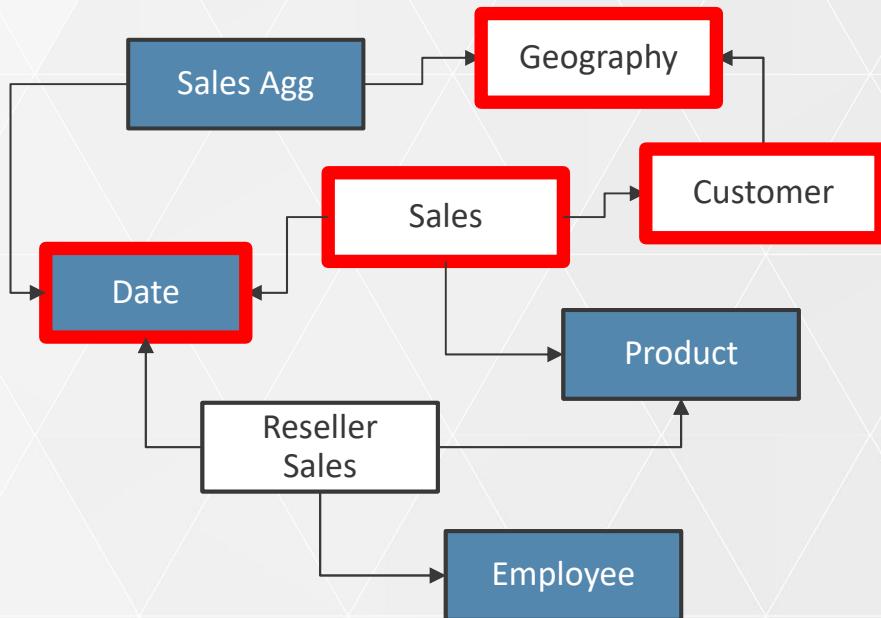
No aggregation hit, DirectQuery only

Import

DirectQuery

sq|bi.

# Querying tables in different islands



```
SUMMARIZECOLUMNS (  
    'Date'[Year],  
    Geography[Continent],  
    "Quantity", SUM ( Sales[Quantity] )  
)
```

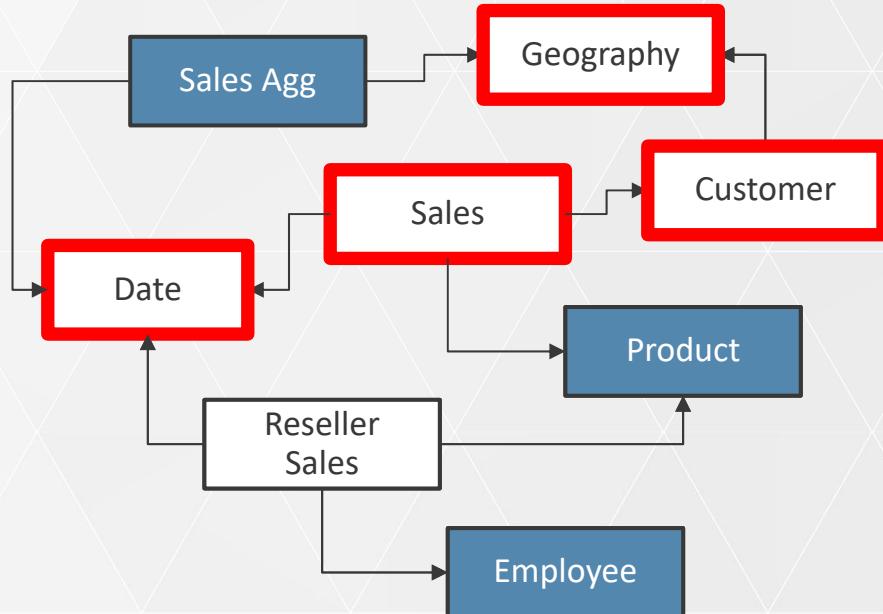
No aggregation hit, cross-islands query

Import

DirectQuery

sq|bi.

# A pure DirectQuery query would be better...



```
SUMMARIZECOLUMNS (  
    'Date'[Year],  
    Geography[Continent],  
    "Quantity", SUM ( Sales[Quantity] )  
)
```

No aggregation hit, DirectQuery only

Import

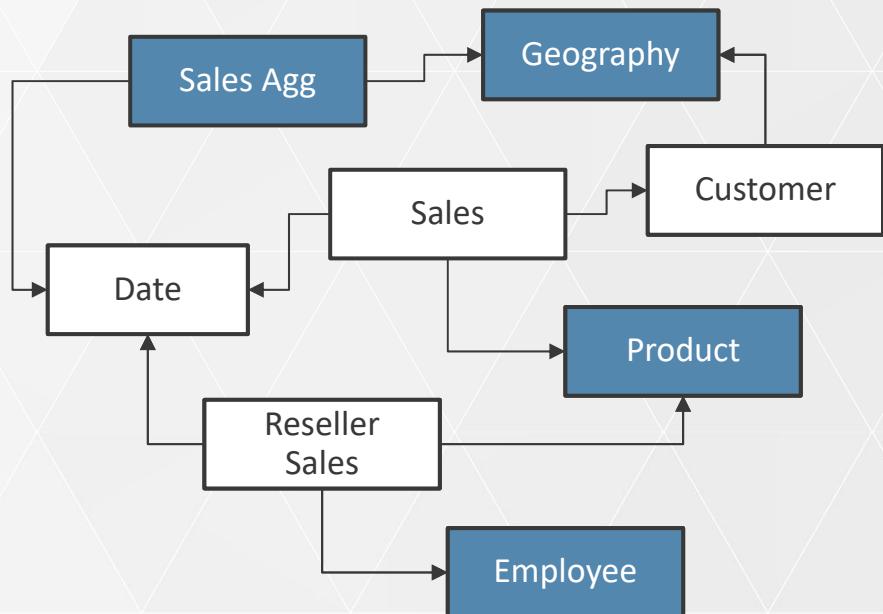
DirectQuery

sq|bi.

## Introducing dual tables

- A dual table exists in both islands
- DirectQuery
  - The table is associated with a DirectQuery query
  - The relational engine can query the table
- Import
  - The table is processed, as is any import table
  - Stored in VertiPaq
  - The VertiPaq engine can query the table
- Analysis Services can choose the storage engine to use

# Using dual tables

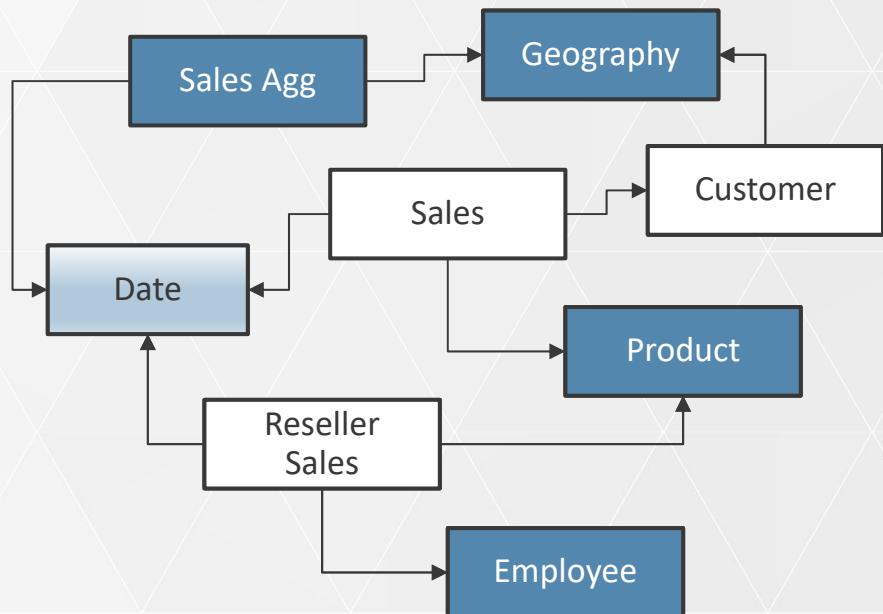


Import

DirectQuery

sq|bi.

# Using dual tables

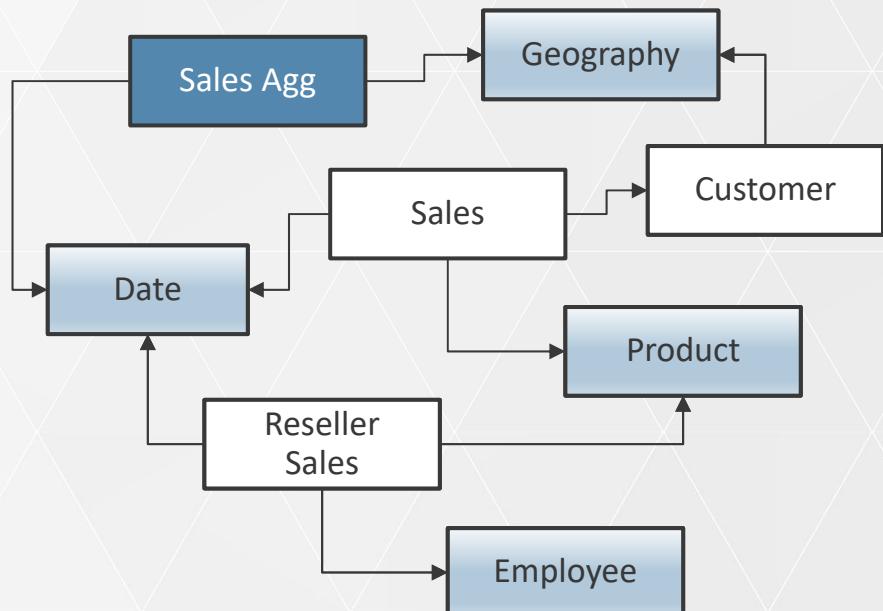


Import

DirectQuery

Dual

# Using dual tables

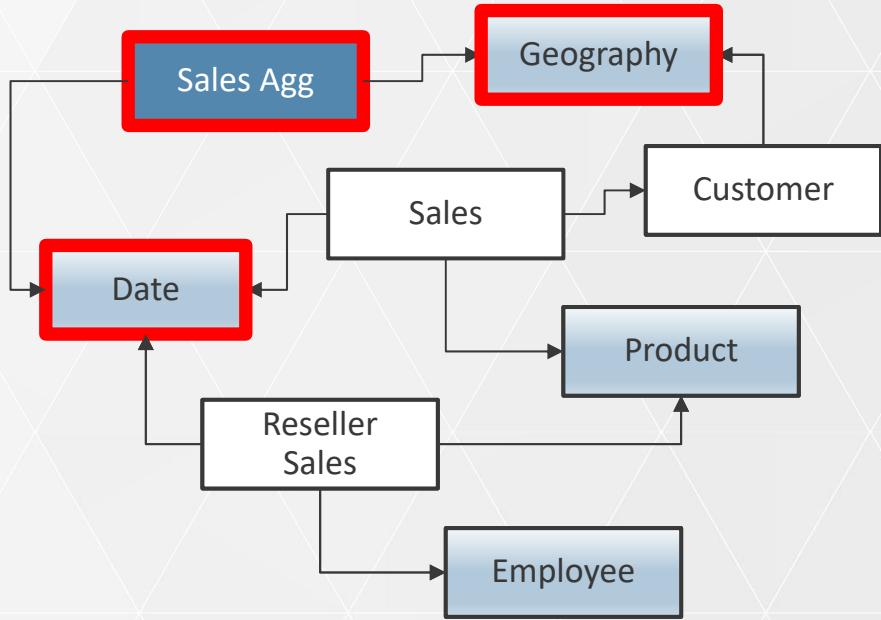


Import

DirectQuery

Dual

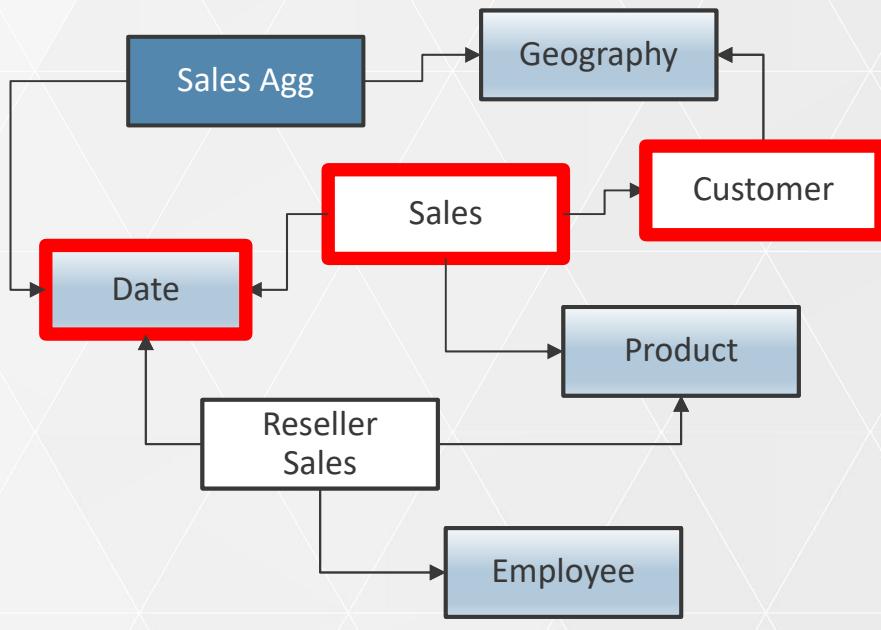
# Using dual tables



```
SUMMARIZECOLUMNS (  
    'Date'[Year],  
    Geography[City],  
    "Quantity", SUM ( Sales[Quantity] )  
)
```

Hits in-memory cache

# Using dual tables



```
SUMMARIZECOLUMNS (  
    'Date'[Year],  
    Customer[Name],  
    "Quantity", SUM ( Sales[Quantity] )  
)
```

No aggregation hit, DirectQuery only

Import

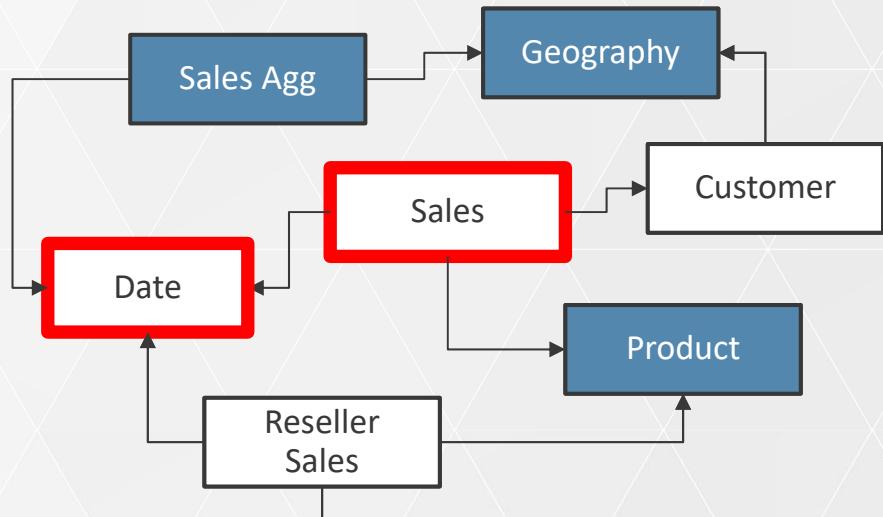
DirectQuery

Dual

## Regular vs. limited relationships

- Aggregations can be used only if the storage engine query goes through regular relationships
- Cross-island relationships are limited, by nature
- Intra-island relationships can be regular, if there is one 1-side
- A dual table exists in two islands at the same time
- Dual tables can generate regular relationships
  - Regular relationships between the import tables
  - Regular relationships between the DirectQuery tables

# Regular vs. limited relationships



```
SUMMARIZECOLUMNS (  
    'Date'[Year],  
    "Quantity", SUM ( Sales[Quantity] )  
)
```

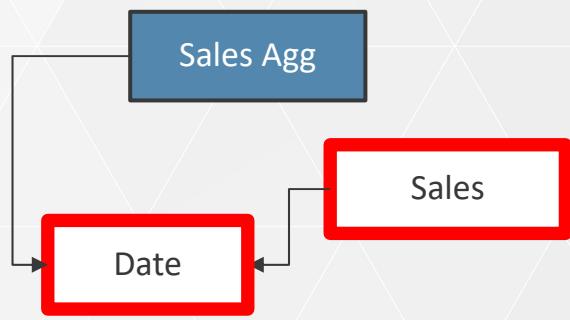
No aggregation hit

Import

DirectQuery

Dual

# Regular vs. limited relationships



```
SUMMARIZECOLUMNS (  
    'Date'[Year],  
    "Quantity", SUM ( Sales[Quantity] )  
)
```

No aggregation hit

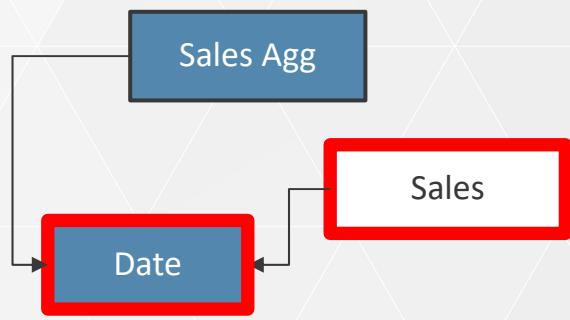
Import

DirectQuery

Dual

sqlbi.

# Regular vs. limited relationships



```
SUMMARIZECOLUMNS (  
    'Date'[Year],  
    "Quantity", SUM ( Sales[Quantity] )  
)
```

No aggregation hit

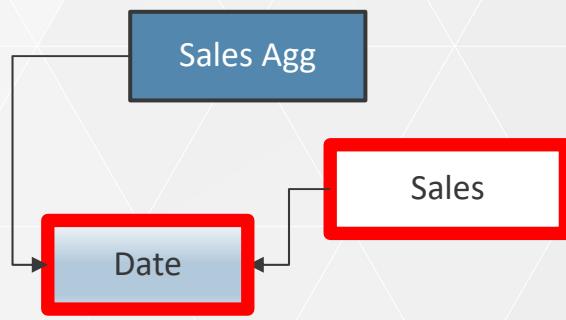
Import

DirectQuery

Dual

sqlbi.

# Regular vs. limited relationships



```
SUMMARIZECOLUMNS (  
    'Date'[Year],  
    "Quantity", SUM ( Sales[Quantity] )  
)
```

Aggregation hit

Import

DirectQuery

Dual

sq|bi.

# Regular vs. limited relationships



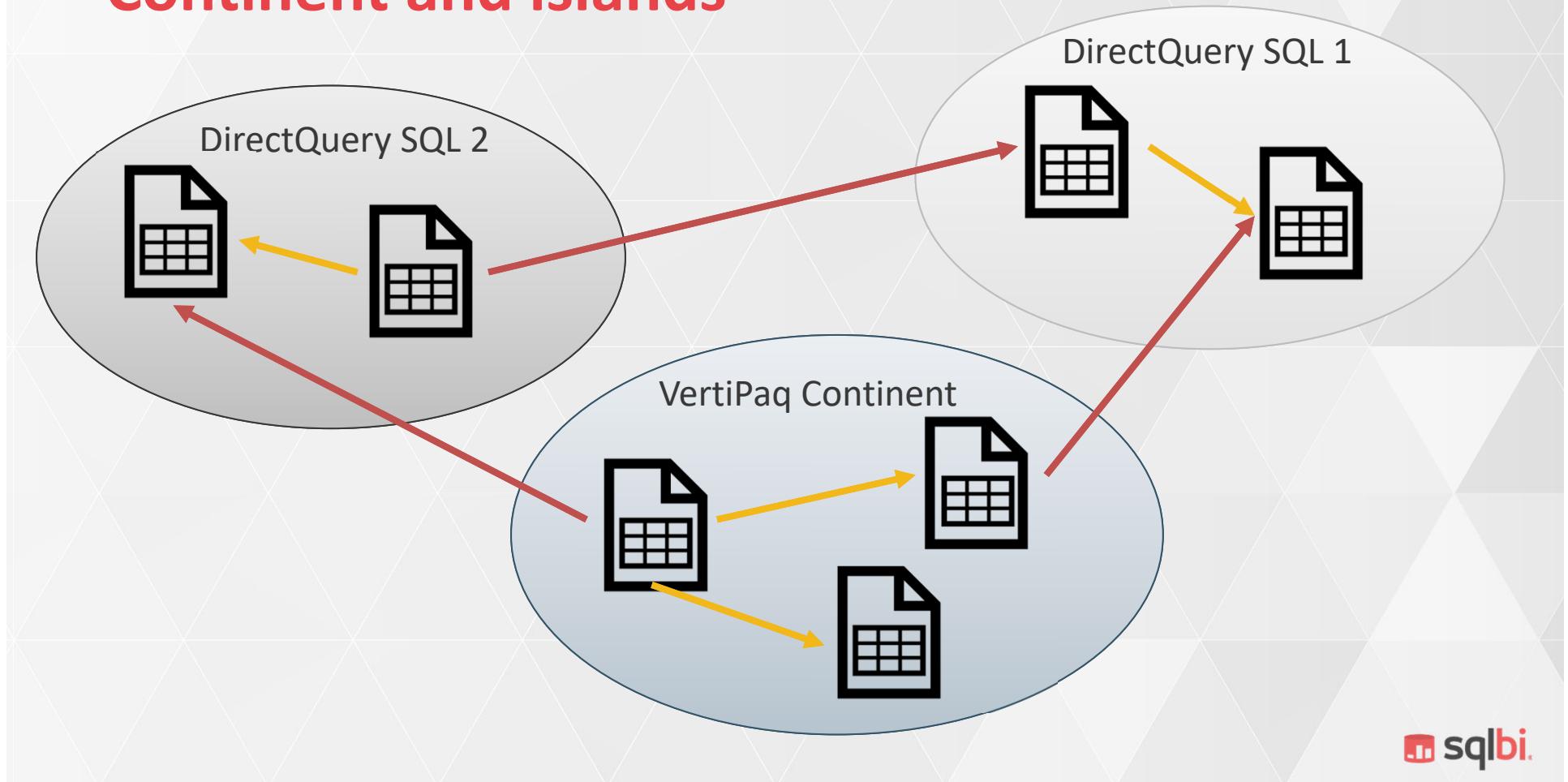
Regular / steaking

- Able to push the join to the source
- Considered for aggregation hits

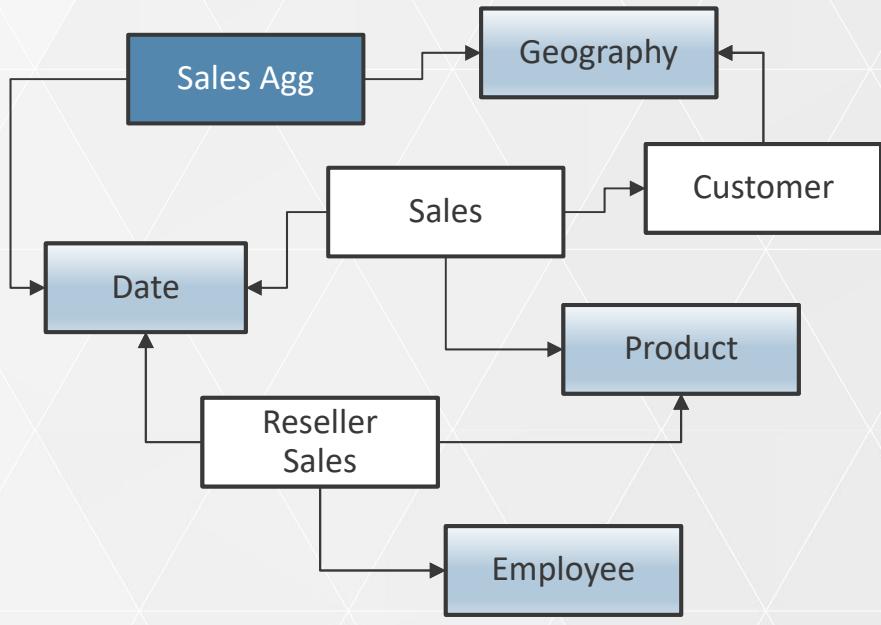
Regular relationship rules (both sides in the same island)

Many-side	One-side
Dual	Dual
Import	Dual or Import
DQ	Dual or DQ

# Continent and islands



# Using dual tables



1-to-many relationship rules

Many-side	One-side
Dual	Dual
Import	Import or Dual
DQ	DQ or Dual

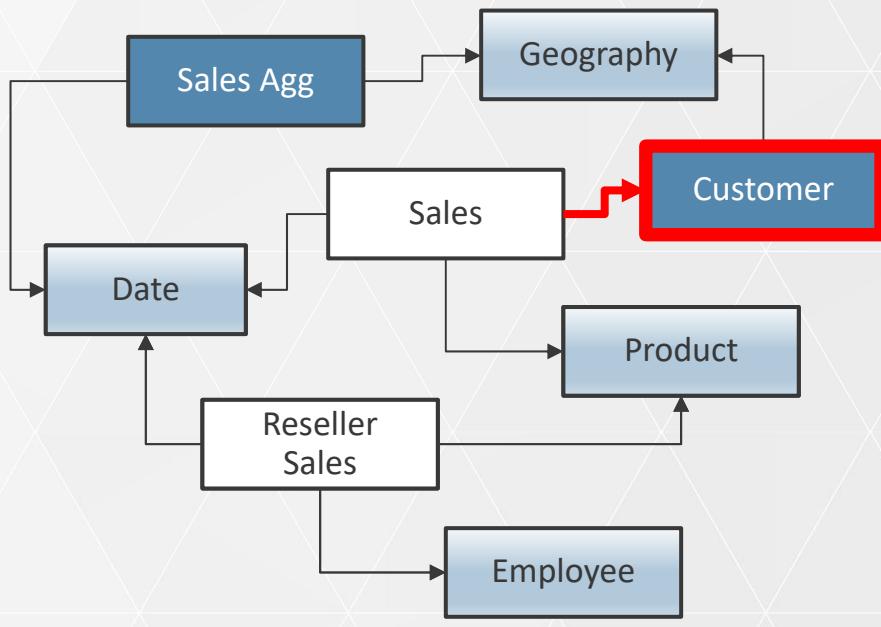
(both sides are from a single island)

Import

DirectQuery

Dual

# Using dual tables



Invalid for aggregations

1-to-many relationship rules

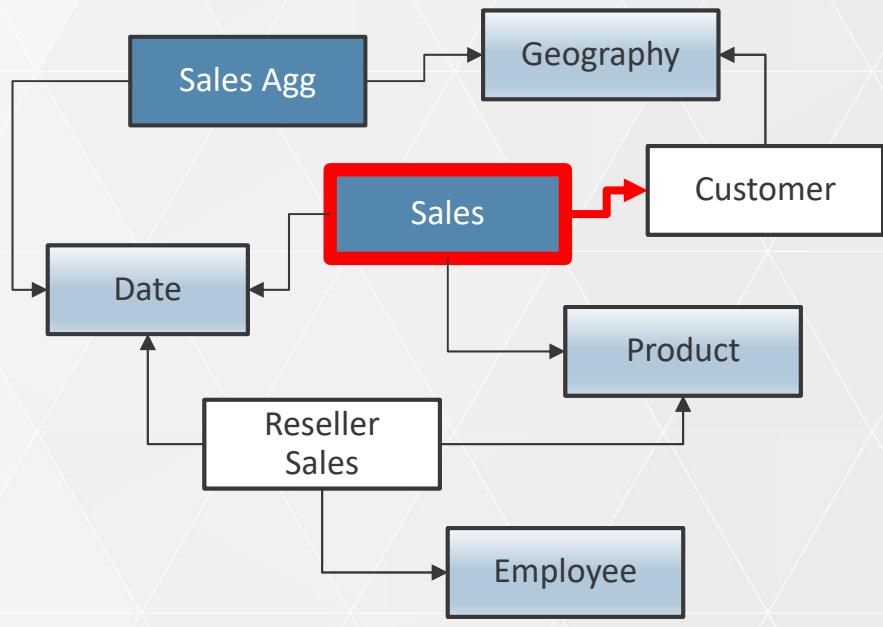
Many-side	One-side
Dual	Dual
Import	Import or Dual
DQ	DQ or Dual

Import

DirectQuery

Dual

# Using dual tables



Invalid for aggregations

1-to-many relationship rules

Many-side	One-side
Dual	Dual
Import	Import or Dual
DQ	DQ or Dual

Import

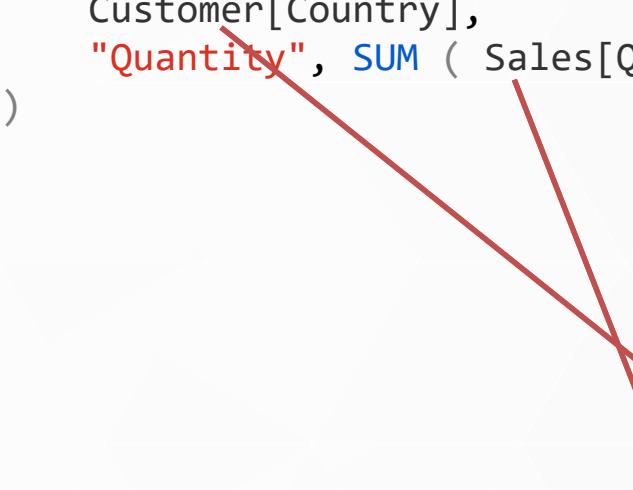
DirectQuery

Dual

# Aggregation hit

During the evaluation of a SE query, the engine checks if the query can be executed by accessing aggregations instead of base tables.

```
EVALUATE  
SUMMARIZECOLUMNS (  
    Customer[Country],  
    "Quantity", SUM ( Sales[Quantity] )  
)
```



Column	AlternateOf	Aggregation
SalesByCountry[Country]	Customer[Country]	GROUPBY
SalesByCountry[Quantity]	Sales[Quantity]	SUM

# Aggregation hit at lower grain

An aggregation can be used at a lower granularity (less detail) by aggregating the pre-aggregated values.

```
EVALUATE  
SUMMARIZECOLUMNS (  
    "Quantity", SUM ( Sales[Quantity] )  
)
```



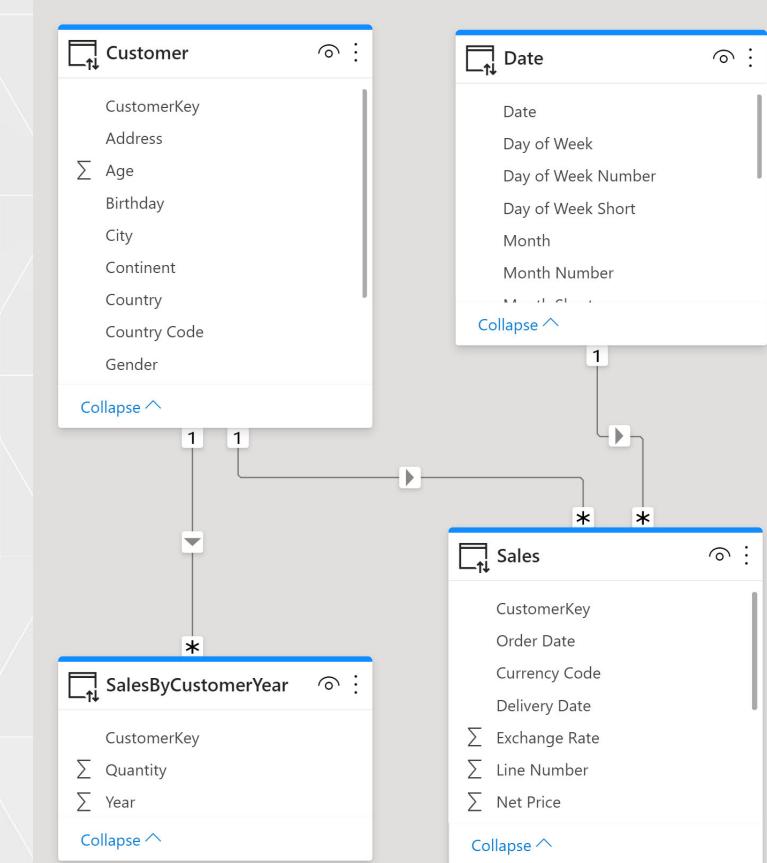
Column	AlternateOf	Aggregation
SalesByCountry[Country]	Customer[Country]	GROUPBY
SalesByCountry[Quantity]	Sales[Quantity]	SUM

# Aggregations and relationships

Need to build a relationship between Customer and the aggregation to hit the aggregation with lower granularity columns

```
EVALUATE  
SUMMARIZECOLUMNS (  
    'Date' [Year],  
    Customer[Country],  
    "Quantity", SUM ( Sales[Quantity] )  
)
```

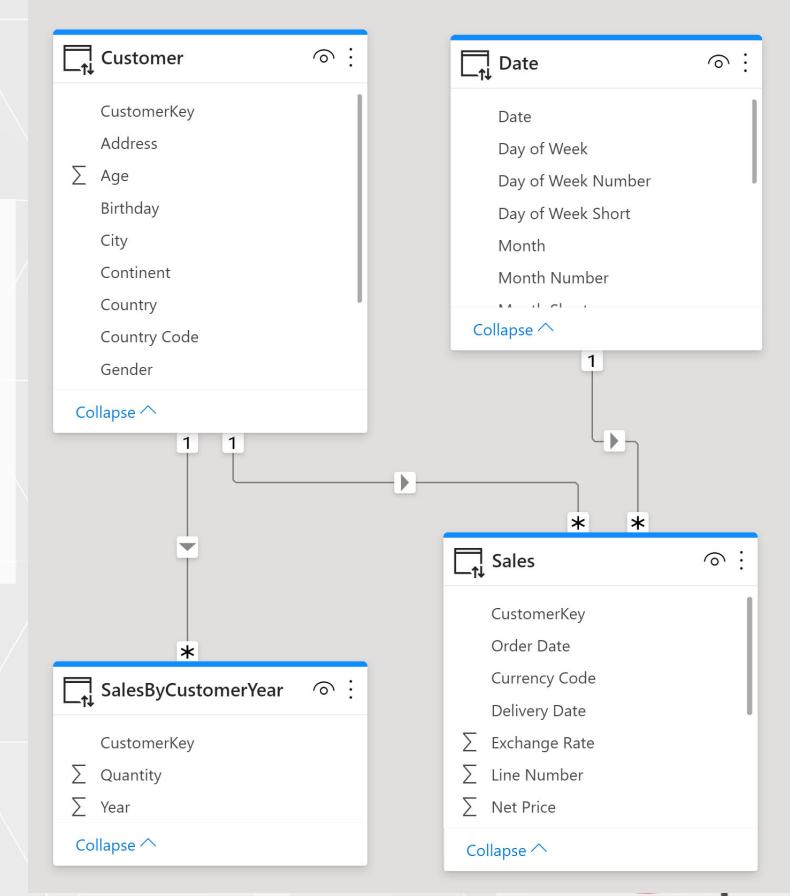
Aggregation not hit if the relationship is not in place



# Aggregations and distinct counts

Group by the Sales[CustomerKey] column to optimize distinct counts

```
EVALUATE  
SUMMARIZECOLUMNS (  
    'Date' [Year],  
    Customer[Country],  
    "Customers",  
        DISTINCTCOUNT ( Sales[CustomerKey] )  
)
```



# Aggregation miss at higher grain

At a higher granularity (more detail) the values are already pre-aggregated and cannot be split. Therefore, the aggregation cannot be used.

```
EVALUATE  
SUMMARIZECOLUMNS (  
    'Date'[Year],  
    'Date'[Month Number], ←  
    "Quantity", SUM ( Sales[Quantity] )  
)
```

No precomputed value  
at the month level

Column	AlternateOf	Aggregation
SalesByYear[Year]	Date[Year]	GROUPBY
SalesByYear[Quantity]	Sales[Quantity]	SUM

# Aggregation miss with custom formulas

Only aggregation functions supported in the aggregation type can be used. As soon as custom formulas are involved, aggregations are no longer used.

```
EVALUATE  
SUMMARIZECOLUMNS (  
    'Date'[Year],  
    "Amount",  
    SUMX (  
        Sales,  
        Sales[Quantity] * Sales[Net price]  
    )  
)
```

Cannot aggregate  
custom expressions

Column	AlternateOf	Aggregation
SalesByYear[Year]	Date[Year]	GROUPBY
SalesByYear[Quantity]	Sales[Quantity]	SUM

# Aggregation with custom formulas

You can aggregate custom formulas, as long as they can be represented as the aggregation of a calculation executed at the line level. More complex formulas cannot be aggregated.

Calculated column:

```
Sales[Line Amount] = Sales[Quantity] * Sales[Net price]
```

```
EVALUATE  
SUMMARIZECOLUMNS (  
    'Date'[Calendar year],  
    "Amount",  
    SUM ( Sales[Line Amount] )  
)
```

Column	AlternateOf	Aggregation
SalesByYear[Year]	Date[Year]	GROUPBY
SalesByYear[Line Amount]	Sales[Line Amount]	SUM

# Aggregation precedence

If multiple aggregations can be used, DAX chooses the one with the highest precedence.  
AS follows only precedence when choosing between aggregations, no other rules applied.

```
EVALUATE  
SUMMARIZECOLUMNS (  
    'Date'[Calendar year],  
    "Amount",  
    SUM ( Sales[Quantity] )  
)
```

Column	AlternateOf	Aggregation
SalesByYear[Calendar Year]	Date[Calendar Year]	GROUPBY
SalesByYear[SumOfQuantity]	Sales[Quantity]	SUM

Column	AlternateOf	Aggregation
SalesByCustomerYear[Calendar Year]	Date[Calendar Year]	GROUPBY
SalesByCustomerYear[CustomerKey]	Customer[CustomerKey]	GROUPBY
SalesByCustomerYear[SumOfQuantity]	Sales[Quantity]	SUM

# Keeping aggregations up to date

AS neither updates aggregations, nor does it check them for consistency.

The same query can use both the aggregation and the real-time DirectQuery data.

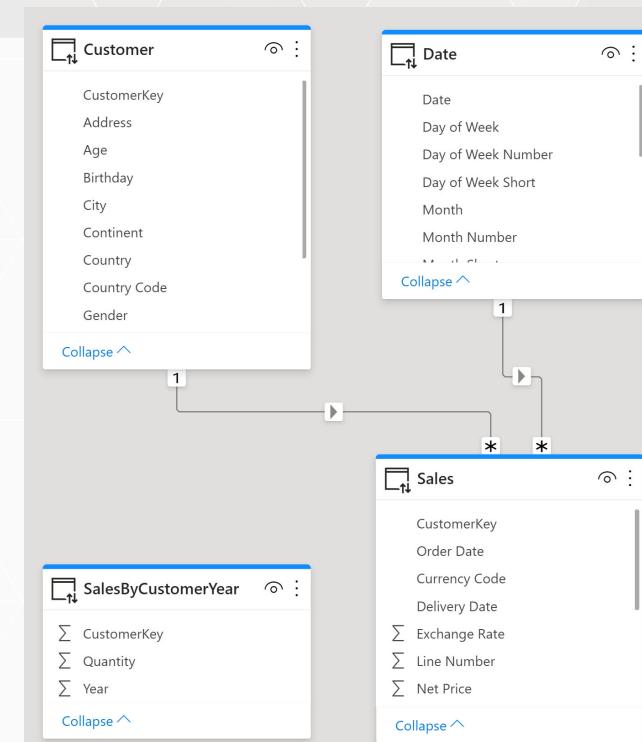
EVALUATE

```
VAR A =  
    SUMMARIZECOLUMNS (  
        'Date'[Calendar year],  
        "Qty", SUM ( Sales[Quantity] )  
    )
```

```
VAR B =  
    SUMMARIZECOLUMNS (  
        Customer[Gender],  
        "Qty2", SUM ( Sales[Quantity] )  
    )
```

RETURN

```
CROSSJOIN ( A, B )
```



## Conclusions

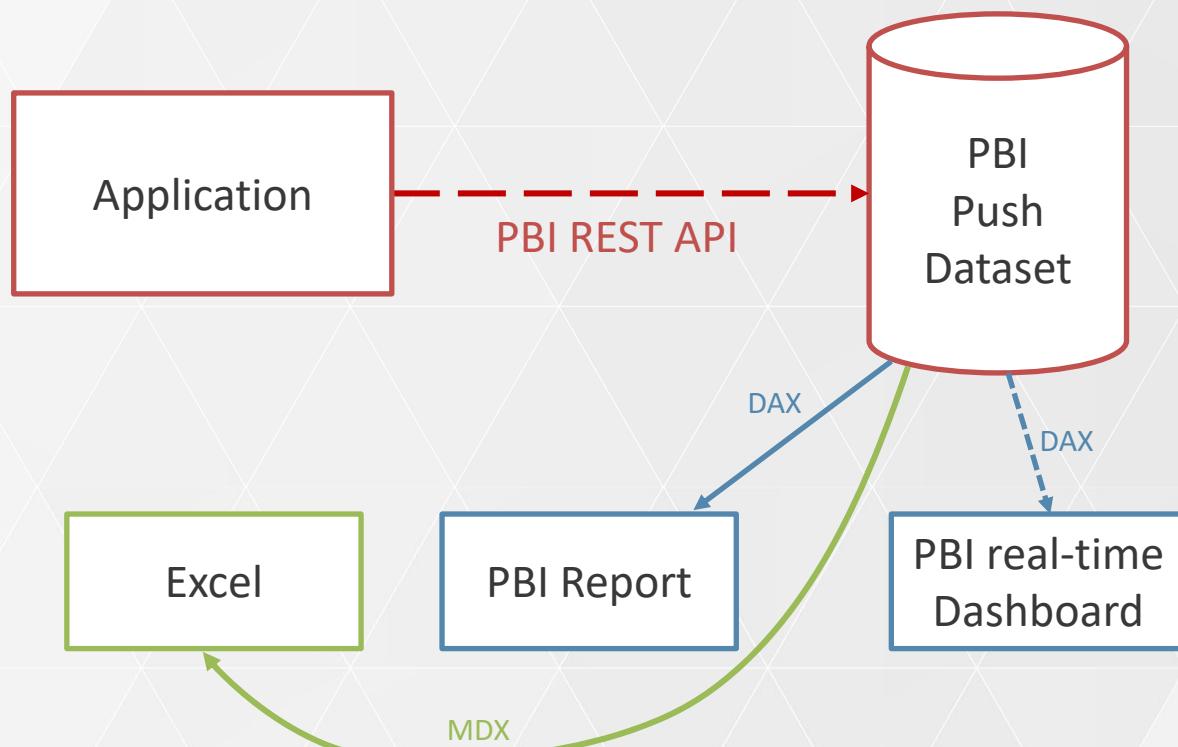
- Aggregations are powerful optimization tools
- Shine on large tables, or large-cardinality columns
- Require some attention in configuring them
- Beware of dual mode: need to sync the cache
- Aggs mapped to materialize views in DirectQuery do not have consistency issues
- Not available (yet) as import-over-import aggregations

A subset of Tabular features for real-time updates

# Push datasets



# What is a push dataset?



## Push datasets

- Tabular models updated in real time (no refresh)
- Only available in Power BI
- Push datasets use a subset of TOM features:
  - Tables, columns, measures, relationships
  - Subset of properties supported
  - Data model created through Power BI REST API
  - Data inserted through Power BI REST API
  - Not visible through XMLA endpoint

## Push datasets limitations: size/retention

- Automatic retention policy for push datasets:
  - None: max 5,000,000 rows per table
  - FIFO (first in first out): keep last 200,000 rows per table
- Max 75 columns per table
- Max 75 tables

## Push datasets limitations: throughput

- Max 10,000 rows per single POST rows call
- 1,000,000 rows added per hour per dataset
- Max 5 pending POST rows calls per dataset
- 120 POST rows calls per minute per dataset
- If 250,000 or more rows in the table:
  - **120 POST rows requests per hour** per dataset
- 4,000 characters per value for string column in POST rows operation

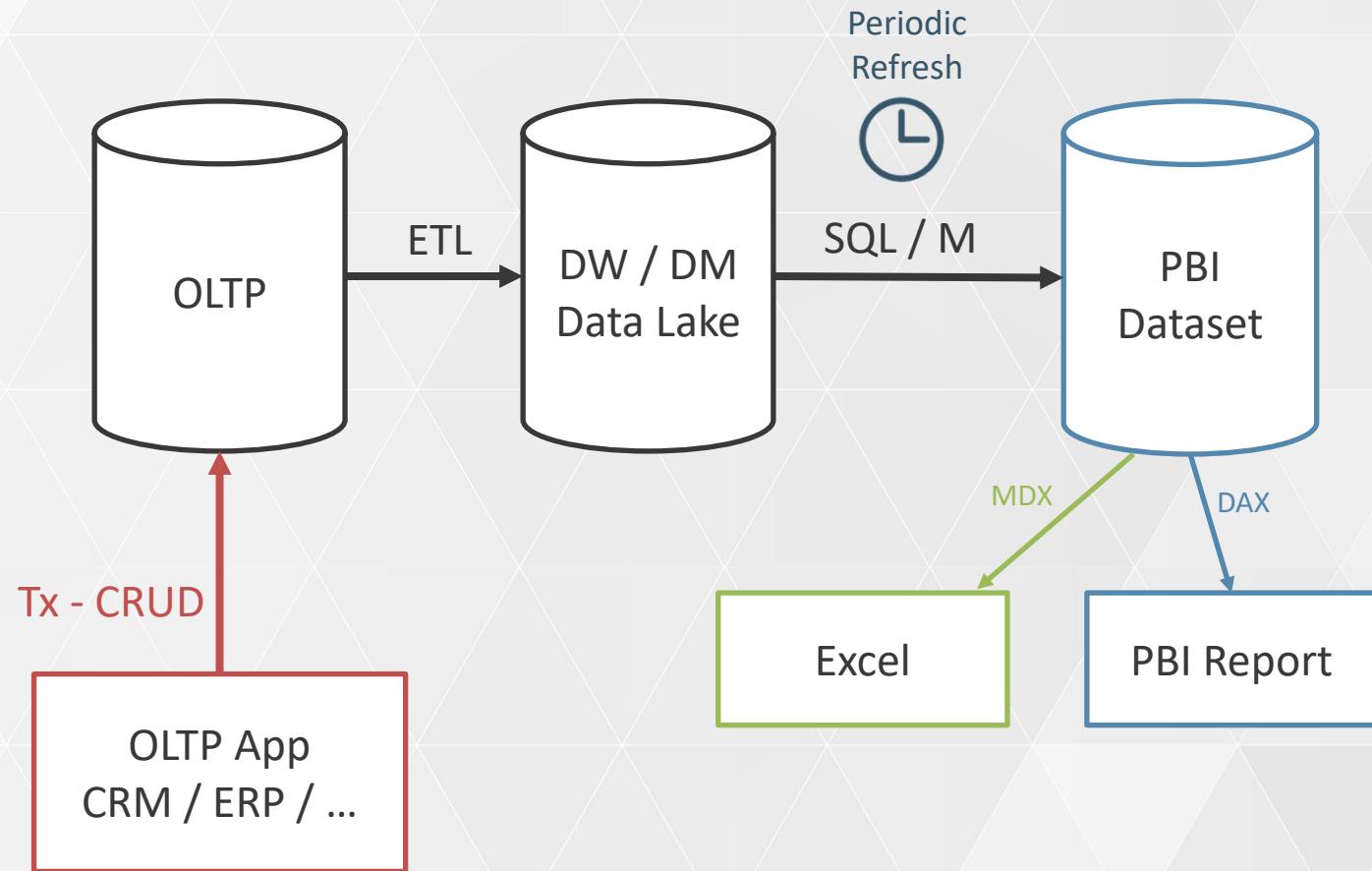
## Data model restrictions

- Inactive relationships are not supported
- Measures using USERELATIONSHIP are not supported (invalid model)
- Measures referencing invalid/non existing measures are not supported (invalid model)
- Hierarchies not supported
- Mark as Date table is not supported

## Hybrid real-time solutions

- An imported dataset can be used to populate a push dataset daily with historical data
- A service continuously pushes new transactions in real time, updating only the tables used in automatically refreshed dashboards
- Push dataset limitations:
  - Evaluate granularity of historical data
  - Evaluate daily granularity

# Classic solution



## Classic solution

- In a classic solution there is a periodic refresh of the Power BI Dataset
- The data latency depends on the refresh schedule
- The minimum latency is the duration of the refresh operation

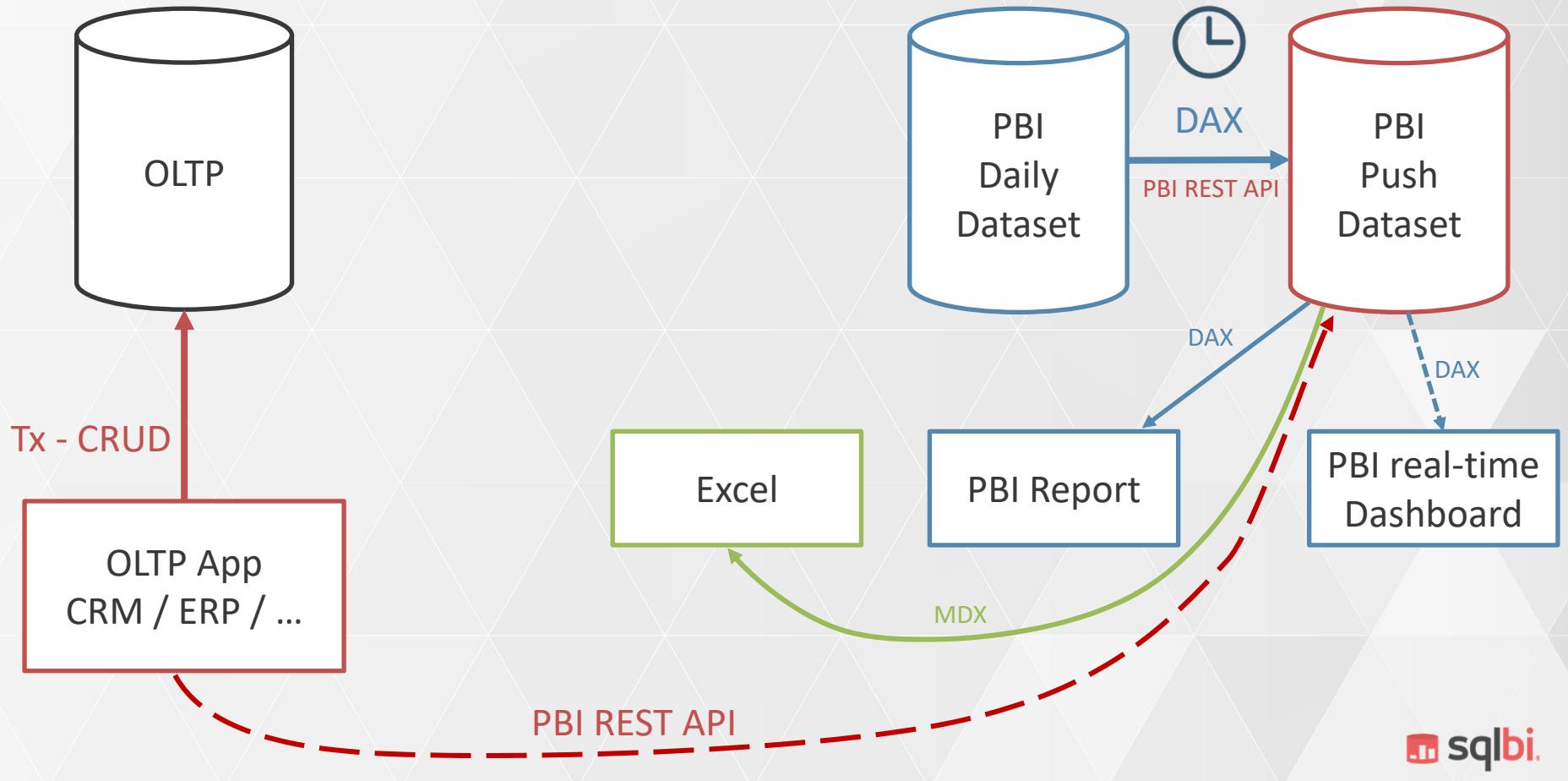
## Classic “push dataset” solution



## Classic “push dataset” solution

- In a real-time solution, the data is pushed in the Power BI Push Dataset using the Power BI REST API
- This works well to show the latest transactions
- History limited to the creation of the Push Dataset
- History limited by Push Dataset limitations:
  - 250,000 rows or 1,000,000 rows

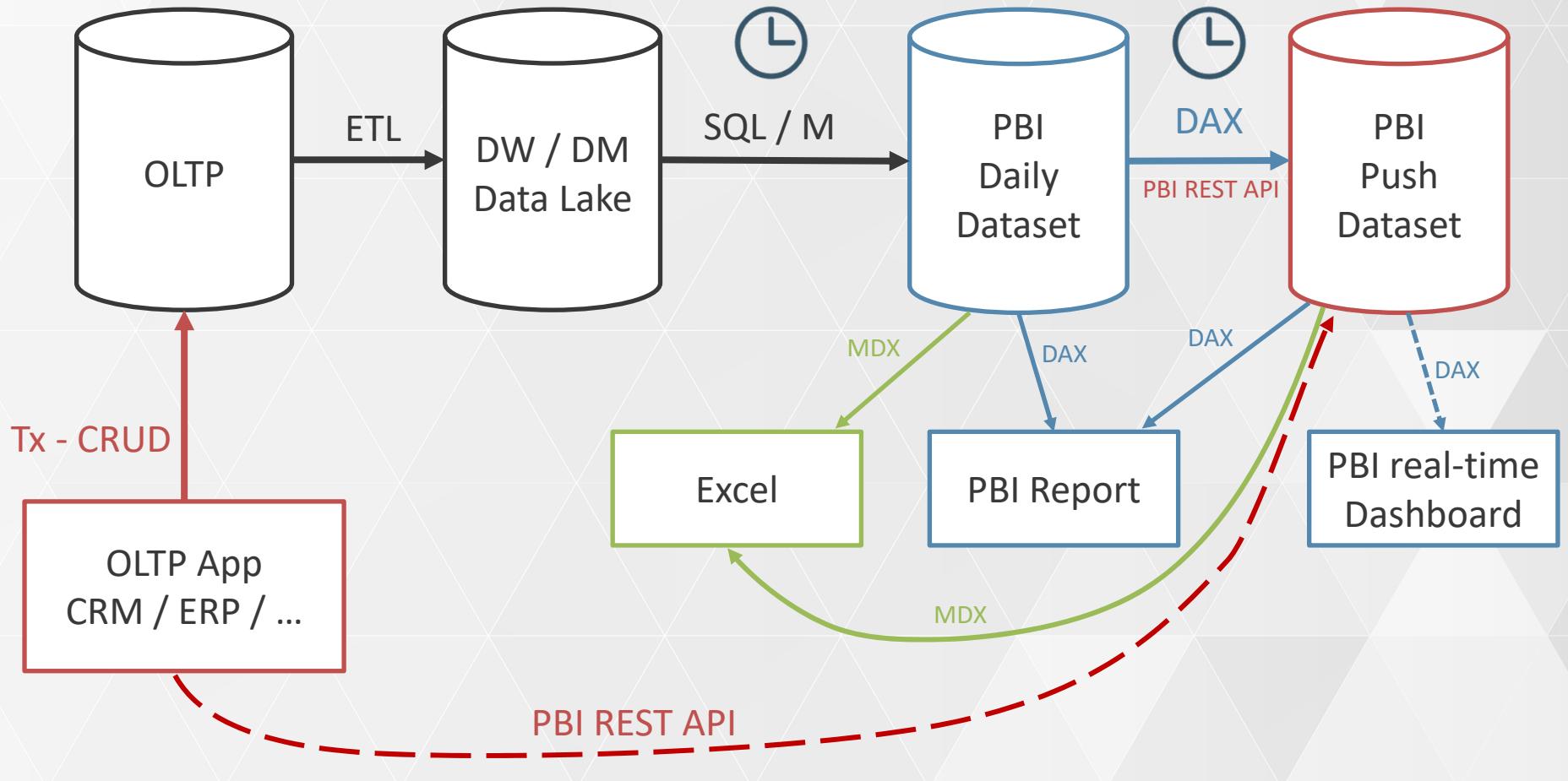
# Daily reset of push dataset



## Daily reset of push dataset

- Load an initial set of rows
  - Historical data aggregated at a certain granularity:
    - Day/product without customer, or
    - Day/customer without products
  - Initial load requires Power BI REST API
- Daily load following daily refresh of a regular Dataset
  - Use DAX queries to retrieve the data to insert
  - Daily refresh duration using DAX: a few seconds
  - Different data source: performance may vary

# Complete solution



## Complete solution

- Create a Push Dataset version of an existing Dataset
- Create DAX queries to reset the Push Dataset
- Schedule a daily reset of the Push Dataset
- Insert transactions in Push Dataset in real-time
- Create new reports over the Push Dataset
- Reuse existing reports, change connection if the granularity is compatible with Push Dataset

Choosing the right hardware is super important!

# Hardware



## Hardware considerations

- Choose the right hardware
- NUMA configurations
- Virtual machines
- Azure and Power BI services

# Buying the Hardware

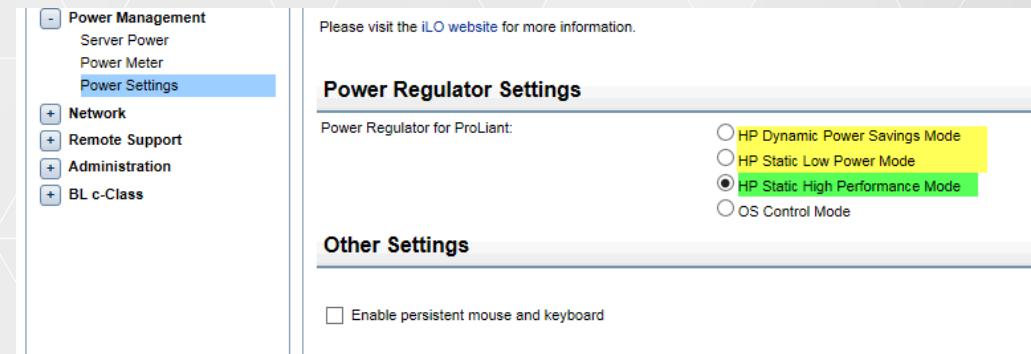
- CPU
  - The fastest you can buy (at least 3.5Ghz)
  - Largest L2/L3 caches
  - **Favor clock speed over number of cores**
- RAM
  - The fastest you can buy (at least 2400 MHz)
- Disk, other hardware
  - Not particularly relevant for AS
  - Power BI Desktop opens files faster with an SSD

## Where is the I/O bottleneck?

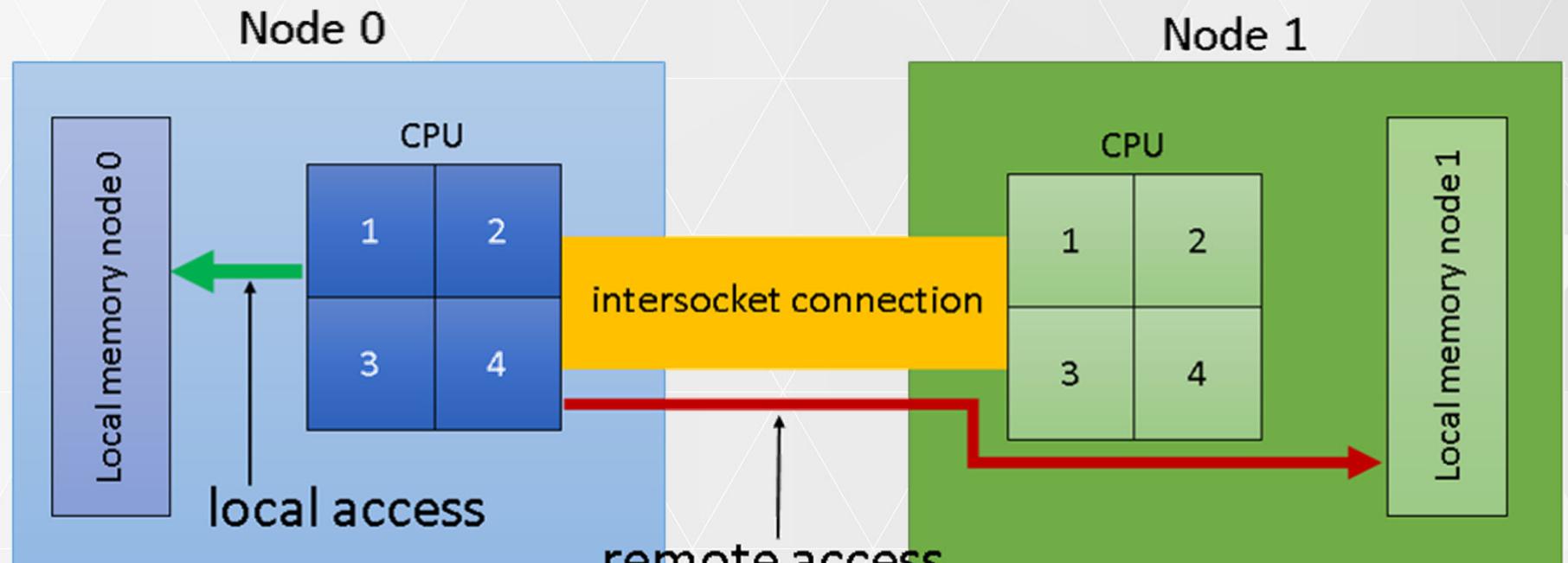
Access	Access time	Human numbers
 1 CPU cycle	0.3 ns	1 s
 L1 cache	0.9 ns	3 s
 L2 cache	2.8 ns	9 s
 L3 cache	12.9 ns	43 s
 RAM access	120 ns	6 min
 Solid-state disk I/O	50-150 µs	2-6 days
 Rotational disk I/O	1-10 ms	1-12 months

# Power settings

- Check power settings in both:
  1. BIOS
  2. Operating System
- Common issue found in many VM hosts
- More details:  
<https://www.sqlbi.com/articles/optimize-hardware-settings-for-analysis-services-tabular/>



# The NUMA Issue



## VertiPaq is NUMA-aware

- The model can be split over different nodes
  - Controlled by NUMA settings in VertiPaq configuration
  - Distributes segments over NUMA nodes
  - Run scan on same NUMA nodes
  - A VertiPaq data cache may end up being materialized on a NUMA node that is not the Formula Engine node
- NUMA distribution active by default on 4 or more nodes
  - You can enforce NUMA on 2 nodes by manipulating settings

## Running in virtual machines

- Very good option
- Chance to update the amount of RAM allocated
- Very low impact on performance
- Be careful with
  - CPU affinity, if host is on NUMA
  - Virtual memory paging (overcommit)
- More information:
  - <https://www.sqlbi.com/articles/optimize-hardware-settings-for-analysis-services-tabular/>

# Azure Analysis Services

- Service tiers and corresponding hardware
- Official CPU measures are in QPU, not tied to specific hardware: (\*) Estimated cores/nodes

Instance	QPU	Mem. (GB)	Cores*	NUMA*
S0	40	10	2	1
S1	100	25	5	1
S2	200	50	10	1
S4	400	100	20	1
S8 v2	640	200	32	2
S9 v2	1280	400	64	2

## Power BI Premium Gen2

- Per User or Capacity license
- Autoscale to add individual v-core for 24-hour periods
- Dynamic loading of databases and columns\*

Capacity	V-cores	Back-end v-cores	Mem. (GB)
P1	8	4	25
P2	16	8	50
P3	32	16	100
P4	64	32	200
P5	128	64	400

## Hardware: conclusions

- Choose the right CPU and RAM speed
- Favor clock speed over number of cores
- Expensive NUMA hardware not always cost-effective
- Virtualization might hide paging
- Cloud services: benchmark with your model

Take maximum advantage of Tabular modeling options

## Architectural choices



# Architectural choices

- Where the data is stored
- Star schema / more normalized
- On-premises / cloud services
- Power Query versus “legacy”
- VertiPaq vs DirectQuery
- Using different data islands
- Composite models and aggregations
- Calculated columns, measures, calculated tables
- Report measures vs model measures
- How security is enforced
- Data latency (days, hours, minutes, real-time)
- User interaction response time required / expected

## Data source types

- Data warehouse
  - On prem / in the cloud
- Data lake
  - ETL strictly required
- OLTP system
  - On prem / in the cloud
  - ETL required, integration with other sources
- Mixed environment
  - DWH, OLTP, Excel files... on prem and in the cloud

# Power Query or legacy connections?

- Power Query
  - Offers more power for ETL
  - Better integration with Dataflows
  - Adds an intermediate layer
  - With DirectQuery query folding is mandatory
- Legacy connections
  - Slightly faster than Power Query
  - Create a tighter connection with the source

## AS on prem or in the cloud?

- On premises
  - You have control over every aspect of the server
  - You must control every aspect of the server
  - Data never leaves your data center
- In the cloud
  - More flexibility and availability
  - Need to move data to the cloud, if not already there
  - Data gateway might be required (based on data source)

## Star schema is the way to go

- Tabular does not strictly require a star schema
- That said, a star schema is the wisest choice
- Snowflakes or more complex models
  - Produce slower query plans
  - Require more complex DAX code
  - Increase the use of complex relationships
- With DirectQuery, it is even worse
- <https://docs.microsoft.com/en-us/power-bi/guidance/star-schema>

## Plan security in advance

- Do not leave security for the end of the project
- The model should be designed with security in mind
- Security might require:
  - Changing the model
  - Creating / updating relationships
  - Performance impact evaluation
  - Different ways of computing values
- The impact on the overall model might be heavy

# Which storage engine to use

- VertiPaq
  - Super fast, compressed, in-memory
  - Full modeling power of DAX (calc columns and tables)
  - Data needs to be refreshed periodically
- DirectQuery over SQL
  - Real-time or huge models (TB scale)
  - Data does not leave the data source
  - Slower, a few DAX limitations
- DirectQuery over AS
  - Reuse of the entire semantic model
  - Increased complexity to optimize performance

## Composite models with DirectQuery over SQL

- Composite models increase the complexity
- VertiPaq is just another data island
- Different scenarios:
  - Main source DirectQuery, VertiPaq for aggregations
  - Main source VertiPaq, DirectQuery for real-time tables
  - Main source DirectQuery, VertiPaq for additional data
- Need to evaluate each scenario adequately
- For aggregations, beware of data coherency
- Hybrid partitions provide further opportunities

# Data islands

- How many data islands?
- Same data island
  - Tables frequently queried together
  - Whenever table expansion is required
- Different data islands
  - Tables coming from different databases
  - Small configuration tables, seldom used in queries
- The presence of multiple data islands is often a symptom of missing ETL steps

## How frequently do you need to refresh?

- How frequently do you need updates?
  - Real time
  - Minutes
  - Hours
  - Days
- DirectQuery (or hybrid partitions) is not the only solution for real-time, you can also create push datasets in Power BI
- Trade-off required if you need minutes
- Hours and days: VertiPaq is your best choice
- Composite models might be an all-around solution

## What are your report's time requirements?

- What is the average query time your users expect?
- How much interaction with the report are you expecting?
- Multiple options:
  - VertiPaq produces faster reports: few seconds
  - DirectQuery usually requires tens of seconds
  - Composite models might be a good balance, but they require advanced and careful planning
- Changing the storage engine is a complex operation: evaluate the requirements well in advance

# Calculated columns, tables, measures

- How much of the business logic can be represented
  - In calculated columns
  - In tables
  - In measures
- When possible, precompute values in the original data sources
- This requires a data warehouse with ETL steps
- Using DAX to perform ETL is rarely good, unless the business logic is very limited in its scope

# Report measures vs model measures

- Is it allowed / suggested to create report measures?
- Report measures
  - Easy to author whenever needed
  - Lose control of which measure is defined in which report
  - Consistency might be lost between different reports
- Model measures
  - Require updating the overall model
  - Full control on dependencies
  - Each measure requires updates made by the model author

# Conclusions

- Where the data is stored
- Star schema / more normalized
- On-premises / cloud services
- Power Query versus “legacy”
- VertiPaq vs DirectQuery
- Using different data islands
- Composite models and aggregations
- Calculated columns, measures, calculated tables
- Report measures vs model measures
- How security is enforced
- Data latency (days, hours, minutes, real-time)
- User interaction response time required / expected

What needs to be done after a model is correctly designed/developed

## Managing Tabular



## The model is ready... what's next?

- You did it! The semantic model is alive and kicking
- Still, it requires caring:
  - Monitoring, to check everything is working fine
  - Deploy new releases over time
  - Periodic processing, to refresh the data
  - Partitioning, if tables are just too large to fully process
  - Backup & restore... just in case!
  - Scaling out, when the user base grows
  - Scripting and automation, to let software do the job for you

## Skills required to manage Tabular

- No more TOM or semantic model properties
- From now on the focus is on the engine behavior and configuration
  - Interfacing with Tabular, TMSL commands
  - Different ways to process your data
  - Partitioning strategies
  - Near real-time scenarios
  - SSAS configuration, memory paging, memory limits
  - Performance counters, extended events

Protocols and libraries to deploy and maintain a Tabular model

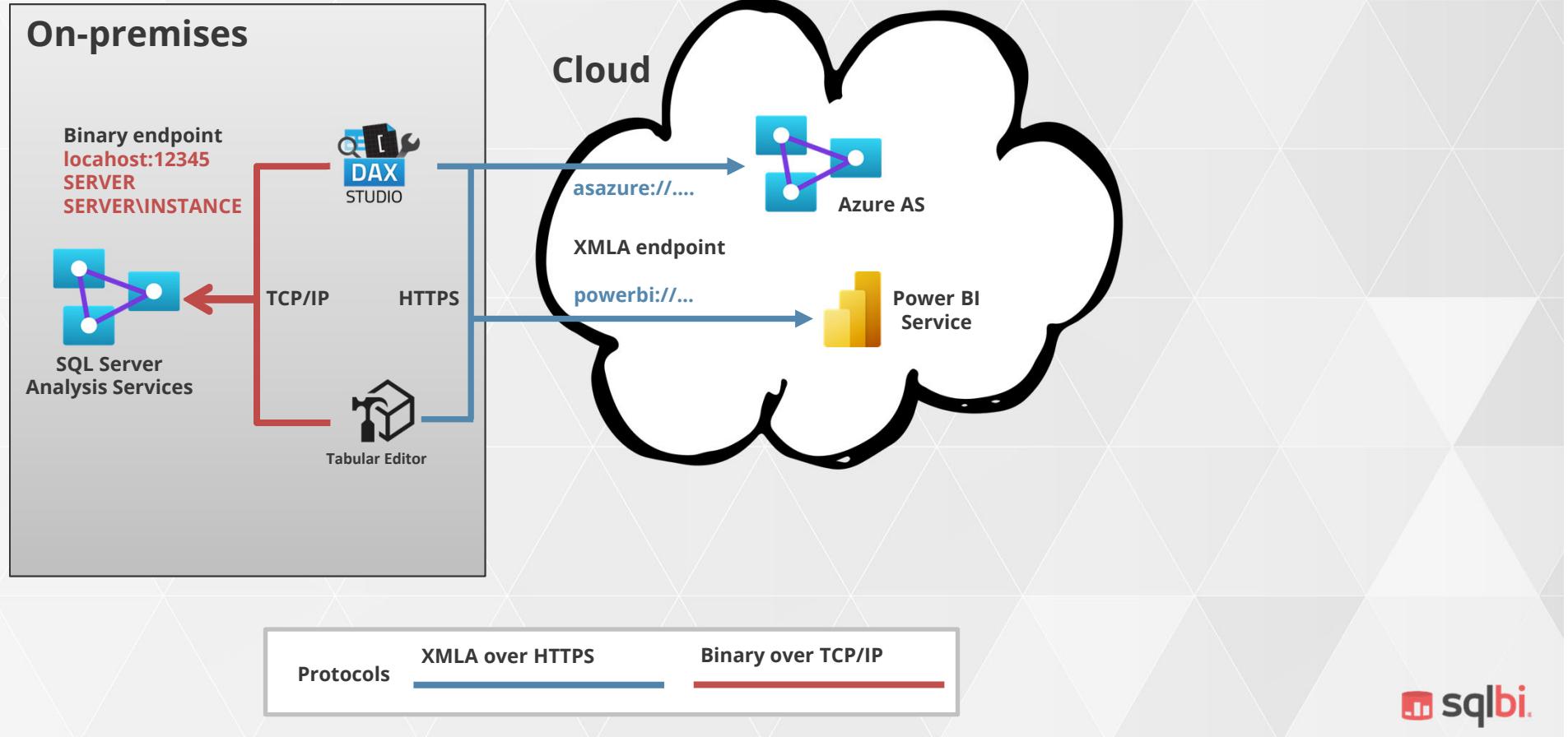
# Interfacing with Tabular



# Interfacing with Tabular

- XMLA endpoint
- AMO and TOM libraries
- TMSL commands
- REST API
  - Azure Analysis Services
  - Power BI

# XMLA endpoint



# Power BI connections

- Exposes AMO/TOM APIs for Power BI datasets
- Necessary only for administration
  - Only available in Power BI Premium
  - `powerbi://api.powerbi.com`
  - Compatible with client tools – like Excel, Tableau
  - Required for management tools – like SSMS, Tabular Editor, DAX Studio
- Client tool alternative
  - OLEDB for OLAP client for Excel
  - `pbiazure://api.powerbi.com`
  - Supported for Excel
  - Compatible but not supported with OLEDB for OLAP clients – like Tableau

## Azure Analysis Services connections

- Azure Analysis Services uses the same XMLA protocol through a different connection URL
  - The URL depends on the Azure region
  - `asazure://<region>.asazure.windows.net`
- No need for a special client-only connection

# AMO and TOM Libraries

- .NET Libraries to access and control AS instances
- **AMO – Analysis Management Objects**
  - Common between Multidimensional and Tabular
  - Multidimensional code refers to AMO only
  - Required at compatibility level lower than 1200
- **TOM – Tabular Object Model**
  - Specific for Tabular
  - Only at compatibility level 1200 or higher

# Example: refreshing a table using TOM

PowerShell Script that requests a full refresh of a table using the TOM library

## PowerShell + AMO/TOM library

```
$server = New-Object Microsoft.AnalysisServices.Tabular.Server  
$server.Connect("powerbi://api.powerbi.com/v1.0/myorg/workspace")  
  
$db = $server.Databases.FindByName("Contoso")  
  
$model = $db.Model  
  
$tableSales = $model.Tables["Sales"]  
  
$tableSales.RequestRefresh("Full")  
  
$result = $model.SaveChanges()
```

# TMSL Commands

- TMSL: Tabular Model Scripting Language
  - JSON syntax to send commands to SSAS Tabular
  - Generated by TOM library using the JsonScripter class
  - SSMS can generate TMSL scripts from UI dialog boxes
- TMSL is good to schedule tasks like data refresh
  - SSAS Server interprets TMSL commands using an additional layer
  - Using TOM directly is more efficient

# Example: refresh a table using TMSL

Requests a full refresh of a table using a TMSL script

## TMSL

```
{  
  "refresh": {  
    "type": "full",  
    "objects": [  
      {  
        "database": "Contoso",  
        "table": "Sales"  
      }  
    ]  
  }  
}
```

# Azure Analysis Services REST API

- Manage Azure Analysis Services
  - Resume/suspend service
  - Refresh
    - POST requests use a TMSL compatible syntax
  - Sync replica servers
- <https://docs.microsoft.com/en-us/rest/api/azure/>
- <https://docs.microsoft.com/en-us/azure/analysis-services/analysis-services-async-refresh>

# Example: refresh a table using AAS REST API

Requests a full refresh of a table using a POST request sent to:

<https://<rollout>.asazure.windows.net/servers/<servername>/models/<database>/refreshes>

## POST payload

```
{  
    "Type": "Full",  
    "CommitMode": "transactional",  
    "RetryCount": 3,  
    "Objects": [  
        {  
            "table": "Sales"  
        }  
    ]  
}
```

# Power BI REST API

- Manage Power BI features
  - No direct manipulation of TOM
  - One exception: Push Datasets API
  - <https://docs.microsoft.com/en-us/rest/api/power-bi/>
- Available through libraries:
  - .NET API  
<https://www.nuget.org/packages/Microsoft.PowerBI.Api/>
  - JavaScript API  
<https://www.nuget.org/packages/Microsoft.PowerBI.JavaScript/>

# Azure Resource Manager (ARM) REST API

- Dynamically manage capacity for Power BI Embedded
- Operations on capacities:
  - Create/Delete/List
  - Resume/Suspend (similar to pause/resume for Azure AS)
- <https://docs.microsoft.com/en-us/rest/api/power-bi-embedded/>

# Static scripts vs. dynamic scripts

- TMSL – Tabular Model Scripting Language
  - Script interpreted by Analysis Services
  - Schedule actions (SQL Agents or other schedulers)
  - Use TMSL for simple **static scripts**
- PowerShell / .NET scripts (C#)
  - Use the TOM model
  - Deploy changes dynamically
  - Use TOM for complex and **dynamic scripts**

## Automating deployment (TOM)

- Customize deployment using PowerShell and C#
- Examples:
  - **PowerShell**  
Copy the same database on a different server/database
  - **C#**  
Deploy a Model.bim file choosing database and server name

# Manipulating a data model (TOM)

- Use TOM to alter a deployed Tabular model
- Example: add a measure to existing model
  - **PowerShell**  
<https://www.sqlbi.com/articles/adding-a-measure-to-a-tabular-model/>
  - **C#**  
Sample code in examples of this section

# Interfacing with Tabular: conclusions

- Different protocols and APIs:
  - XMLA Endpoint
  - TOM / AMO
  - TMSL
  - REST API (Azure AS and Power BI)
- We have seen an overview
- More details in following modules

What you need to know after development

# Deploying Tabular models



## Deploying a solution in production

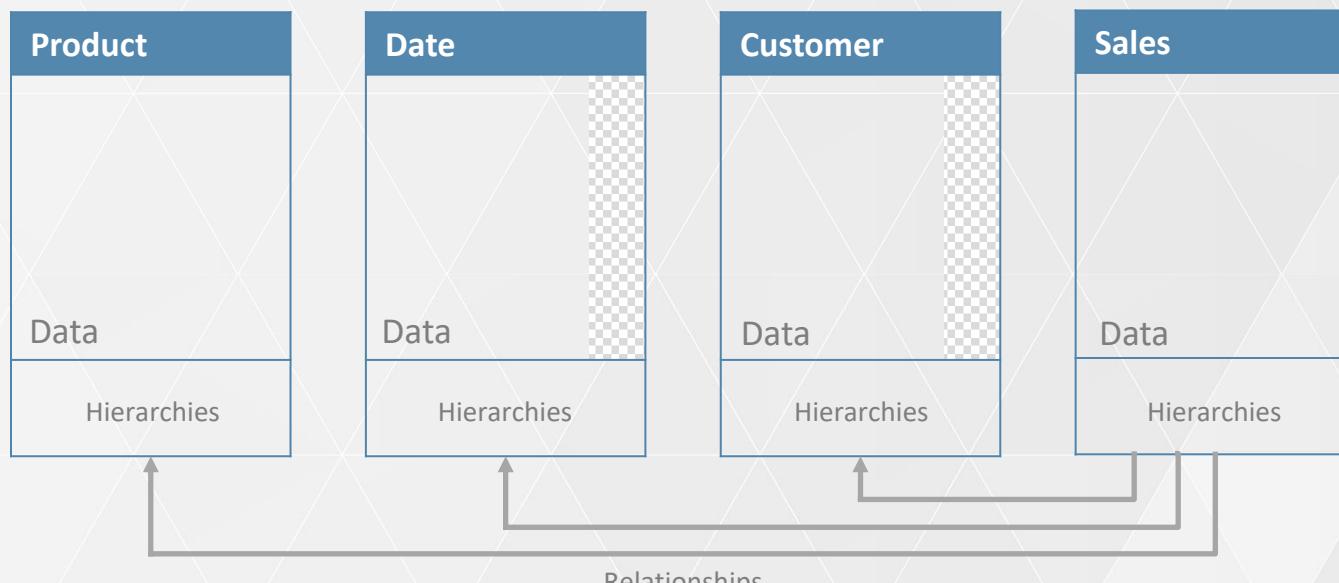
- Admin rights needed
- Development and Production configurations might be different
  - Different impersonation modes
  - Different database connections
- Security role membership is often handled after the deployment
- Imported data (VertiPaq) must be updated after deployment

# Deployment steps

1. Deployment
  - New database: no initial data
  - Update existing database: invalidate one or more tables
2. Data refresh
  - Read tables from data source and populate columns
3. Model recalc
  - Compute calculated columns and calculated tables
  - Compute relationships and user/attribute hierarchies

# Deployment of new model

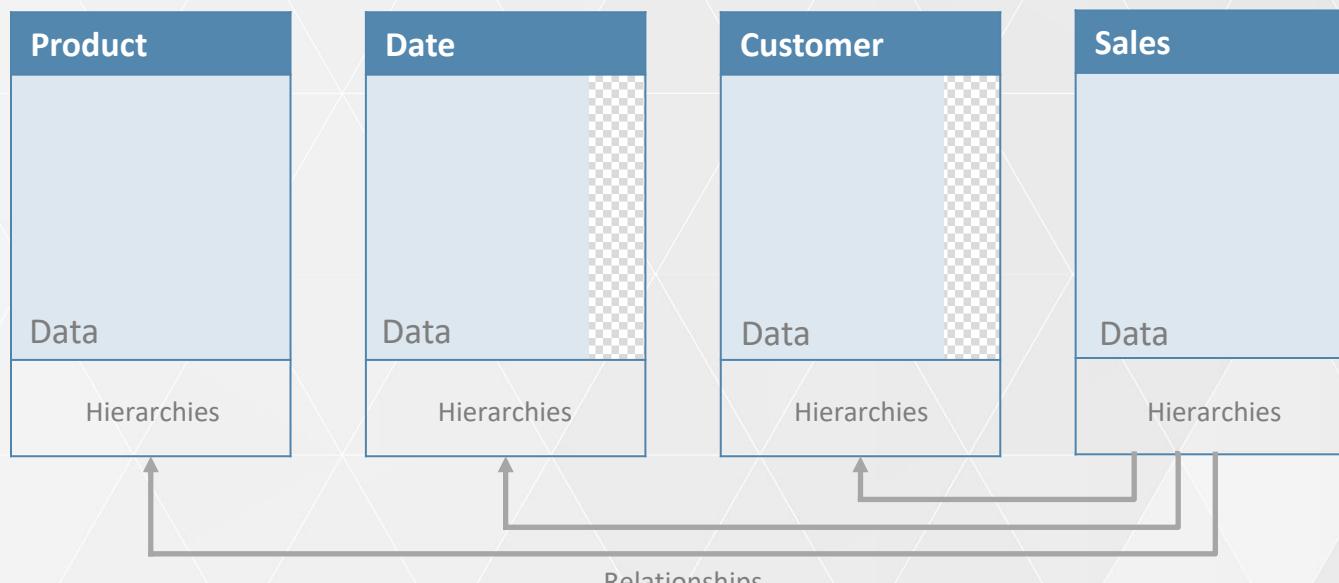
After the deployment of a new model, there is only metadata.  
Table data, hierarchies, and relationships are not processed.



# Data refresh

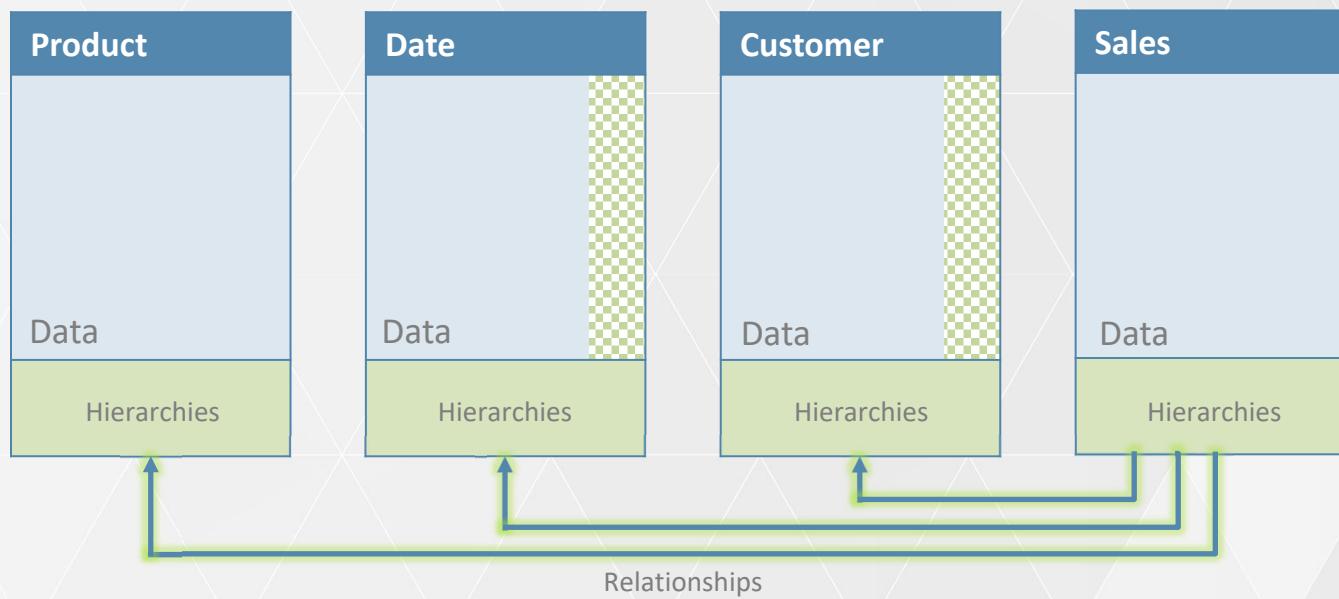
During data refresh, all the native columns are processed.

Calculated columns, calculated tables, user/attribute hierarchies, and relationships are not processed.



# Model recalc

During model recalc, Analysis Services processes calculated columns, calculated tables, user/attribute hierarchies, and relationships.

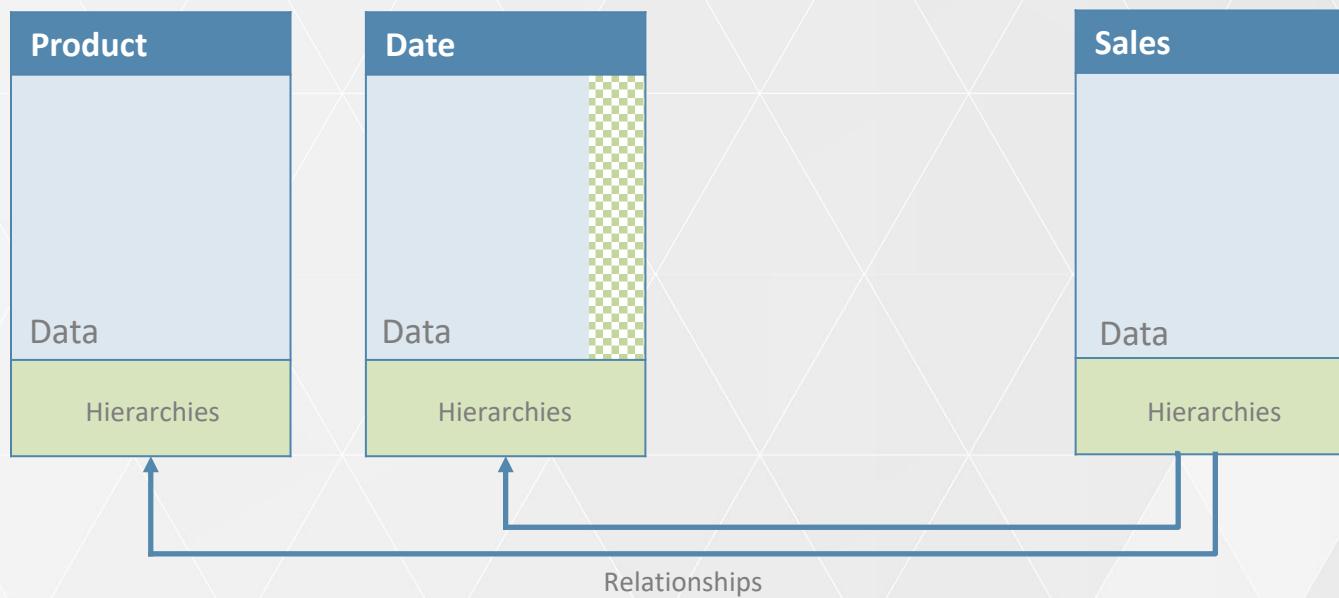


## Deployment and processing

- Changes that invalidate data and require table refresh:
  - Adding/modifying tables/columns
- Changes that invalidate data and require model recalc:
  - Adding/modifying calculated tables/calculated columns
  - Adding relationships
- Changes that do not require data refresh:
  - Adding/removing/changing measures
  - Removing columns/tables/relationships

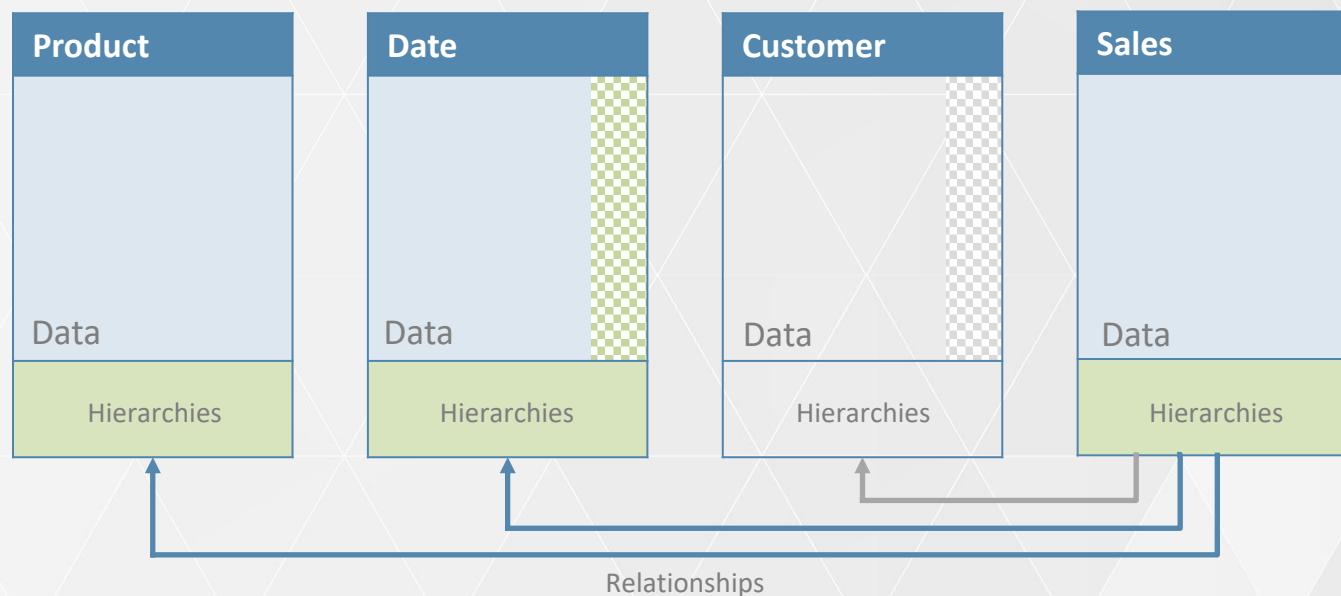
# Add Customer table

The model is already processed, we will add a new Customer table with a relationship with Sales.



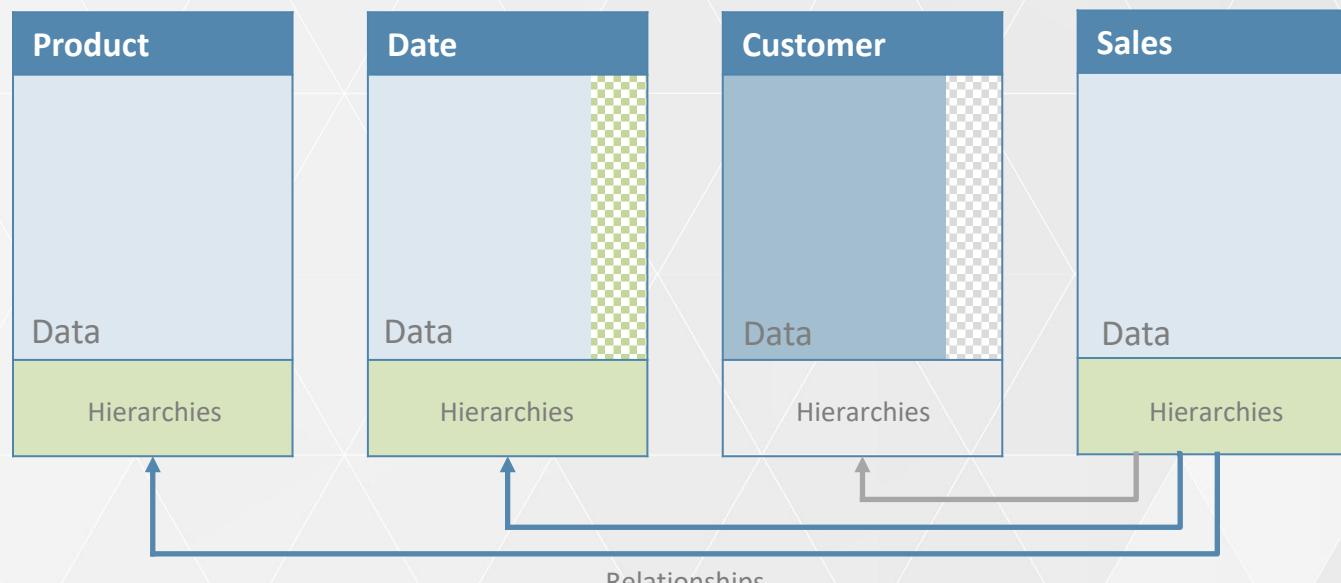
# Add Customer table

Adding a table with a relationship and a calculated column does not populate any data.



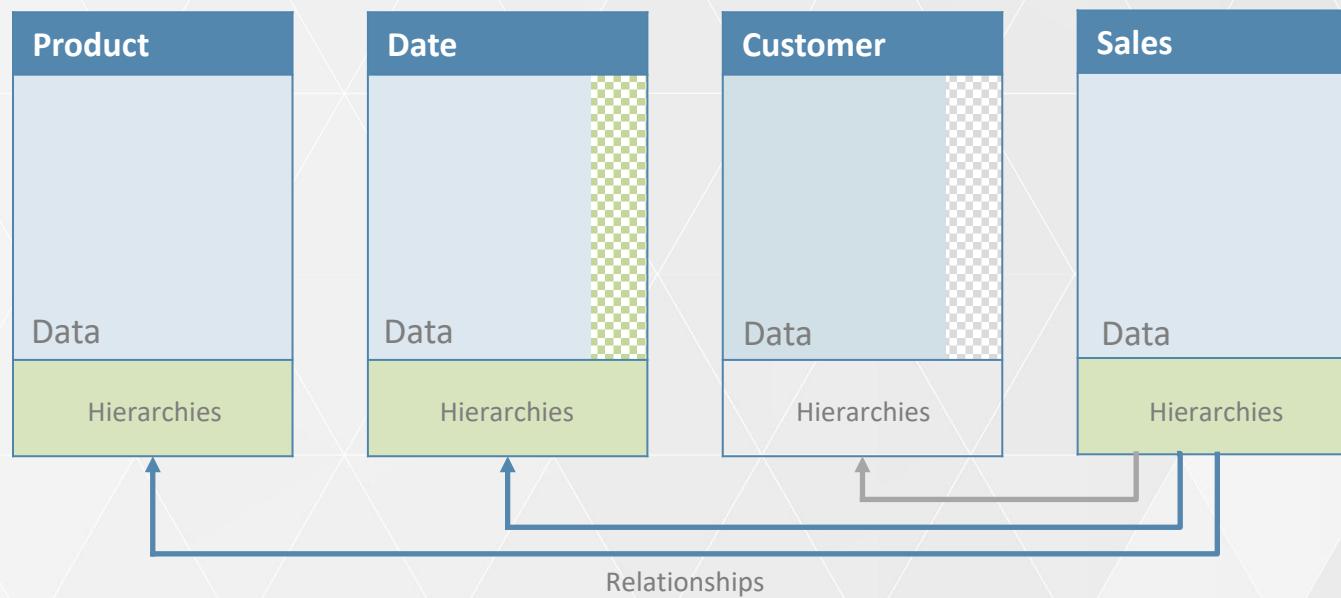
# Add Customer table + data refresh

A data refresh operation is needed on the Customer table to populate the native columns of the newly created table.



# Add Customer table + data refresh

Once the data in Customer is populated, we still need to compute the additional structures: hierarchies, calculated columns and relationships. Process recalc is needed.

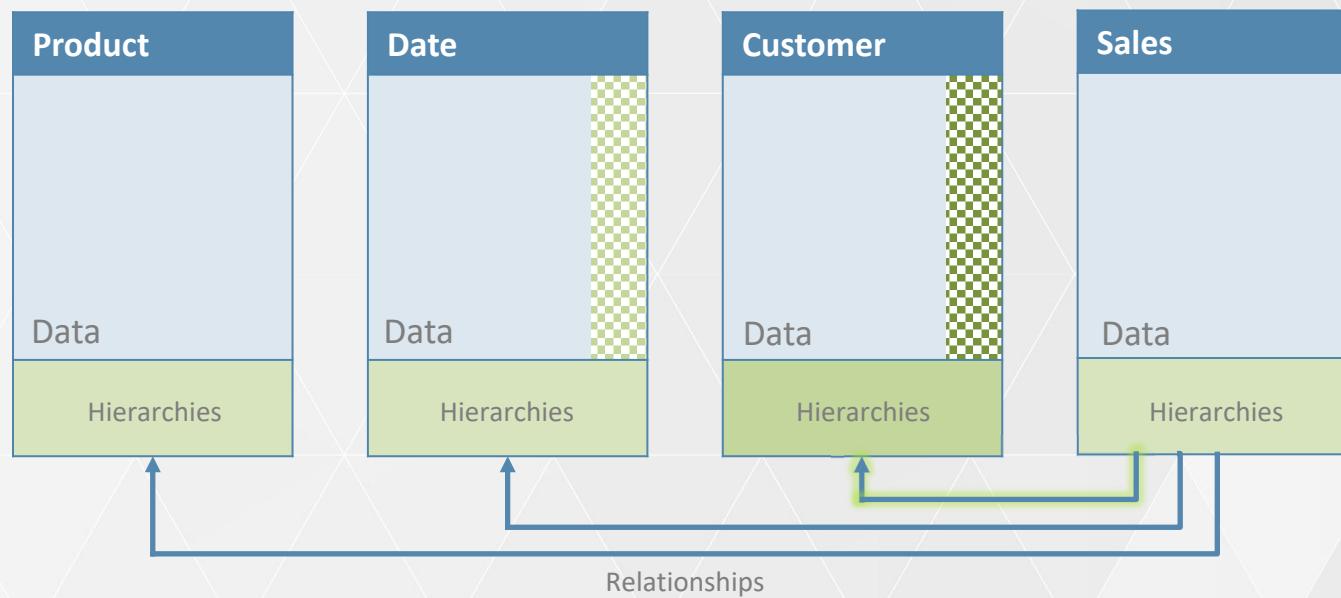


# Add Customer table + data refresh + model recalc

Model refresh populates calculated columns, hierarchies, and relationships.

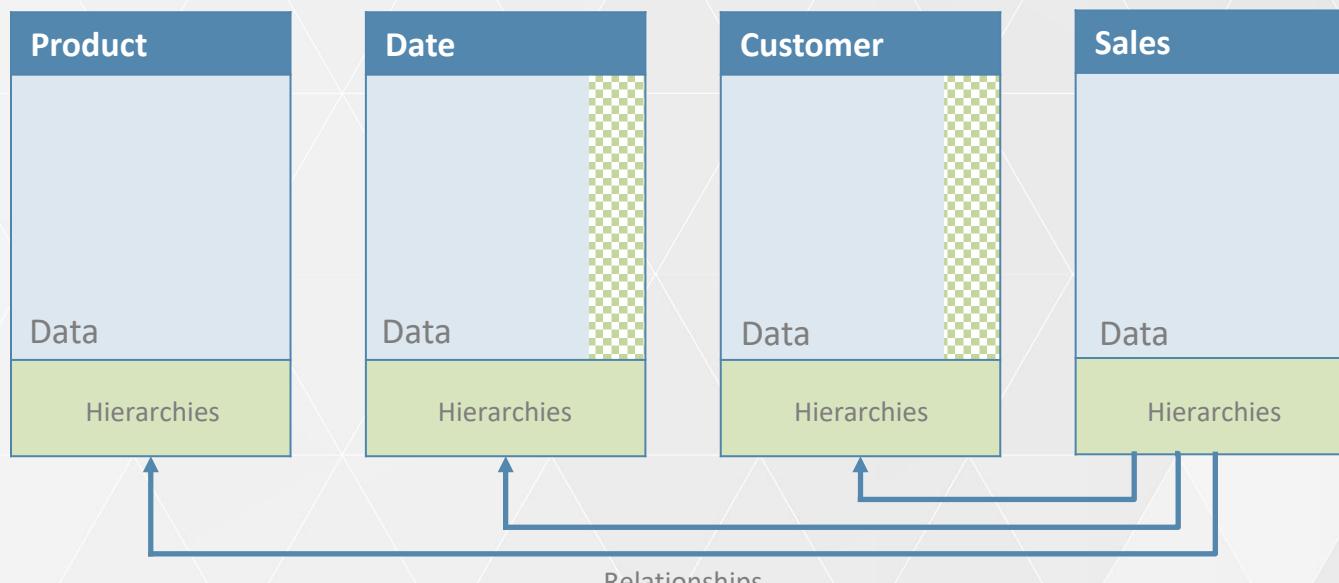
Even though we added a single table, the recalc operation is invoked at the model level.

Model recalc computes only the objects that are not in a valid state already.



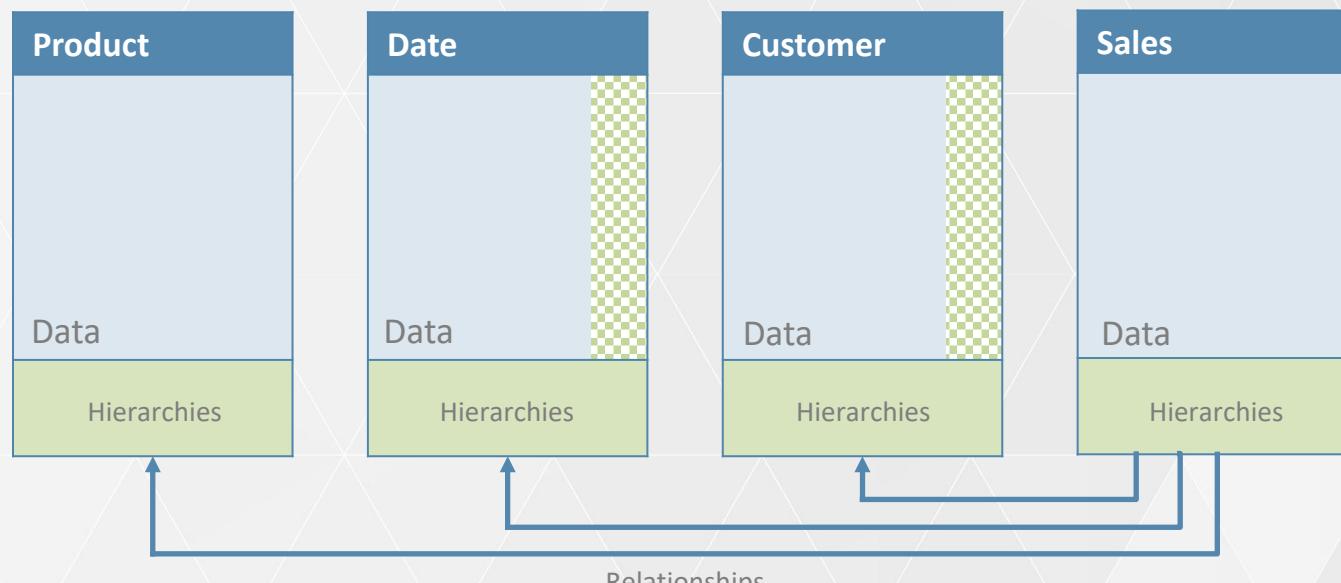
# Add Customer table + data refresh + model recalc

After processing of the table data and model recalc, the model is valid and ready to be queried.



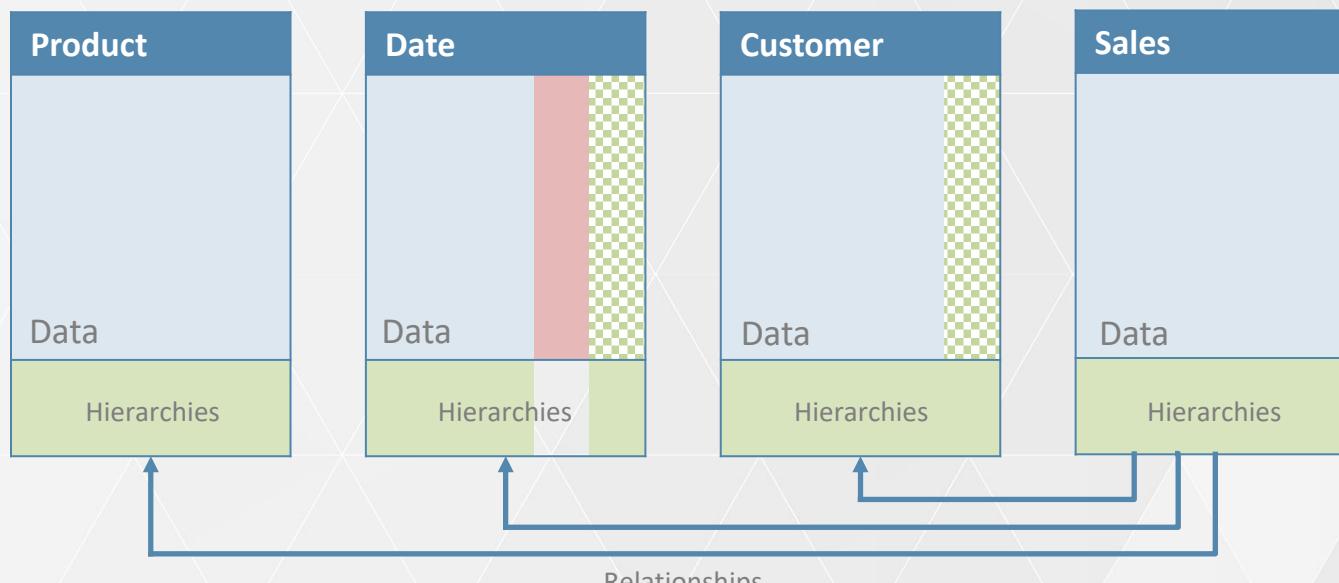
# Add a column in Date

The model is valid, but we deploy a change that creates a new column in the Date table. The newly introduced column needs to be read from the data source.



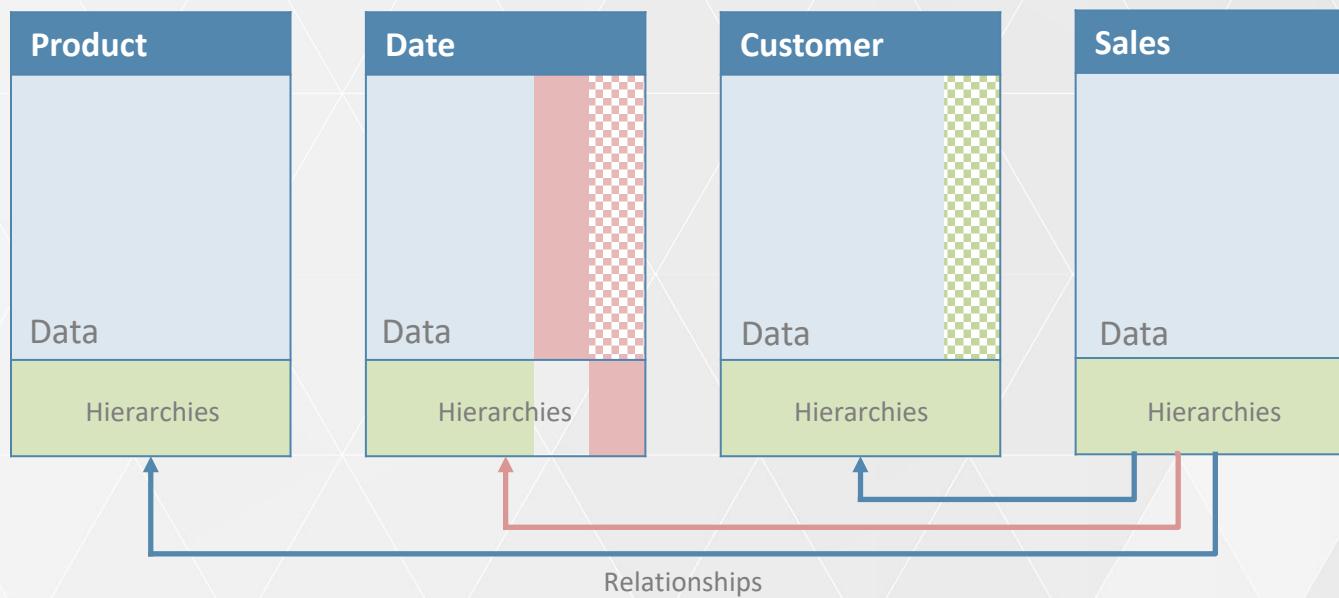
# Add a column in Date

The Date table is still valid, the newly introduced column is empty.



# Add a column in Date

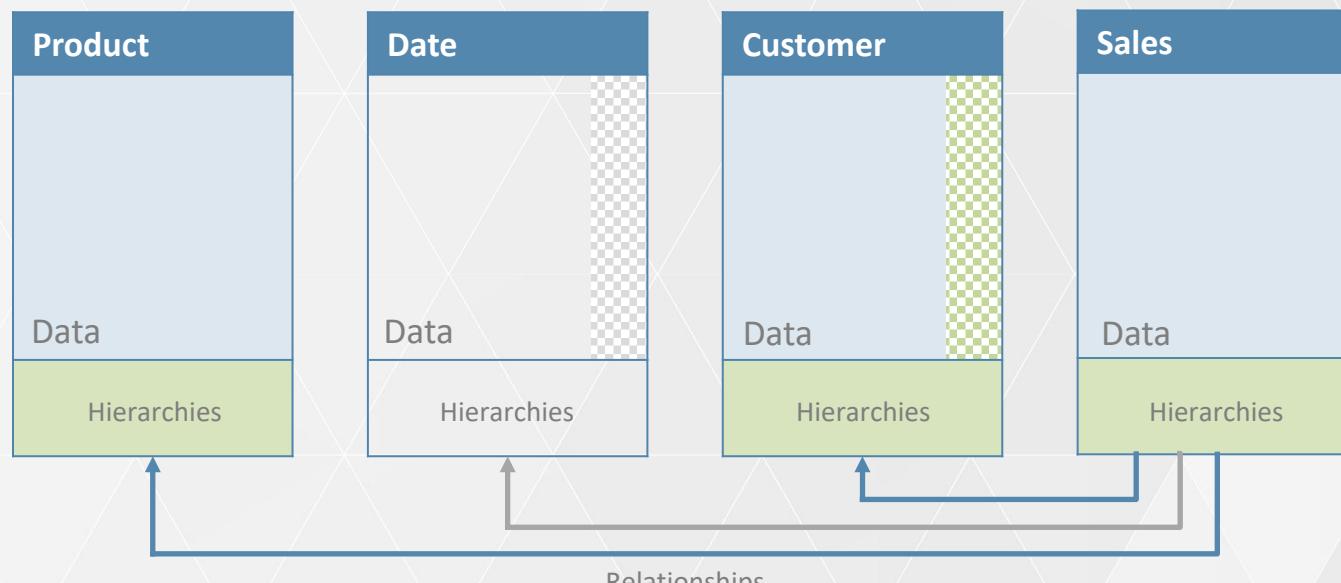
Adding a column in a table might invalidate dependent objects (hierarchies, calculated columns, and relationships).



# Add a column in Date

Adding a column in a table might invalidate dependent objects (hierarchies, calculated columns, and relationships).

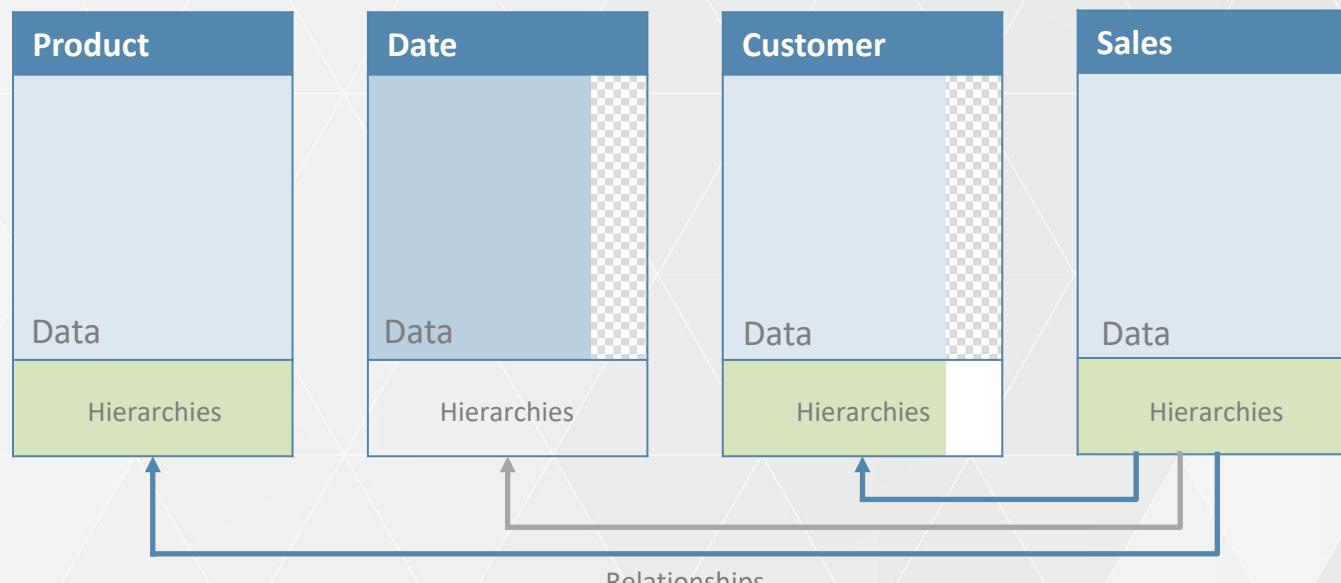
Data refresh required for the entire Date table.



# Add a column in Date

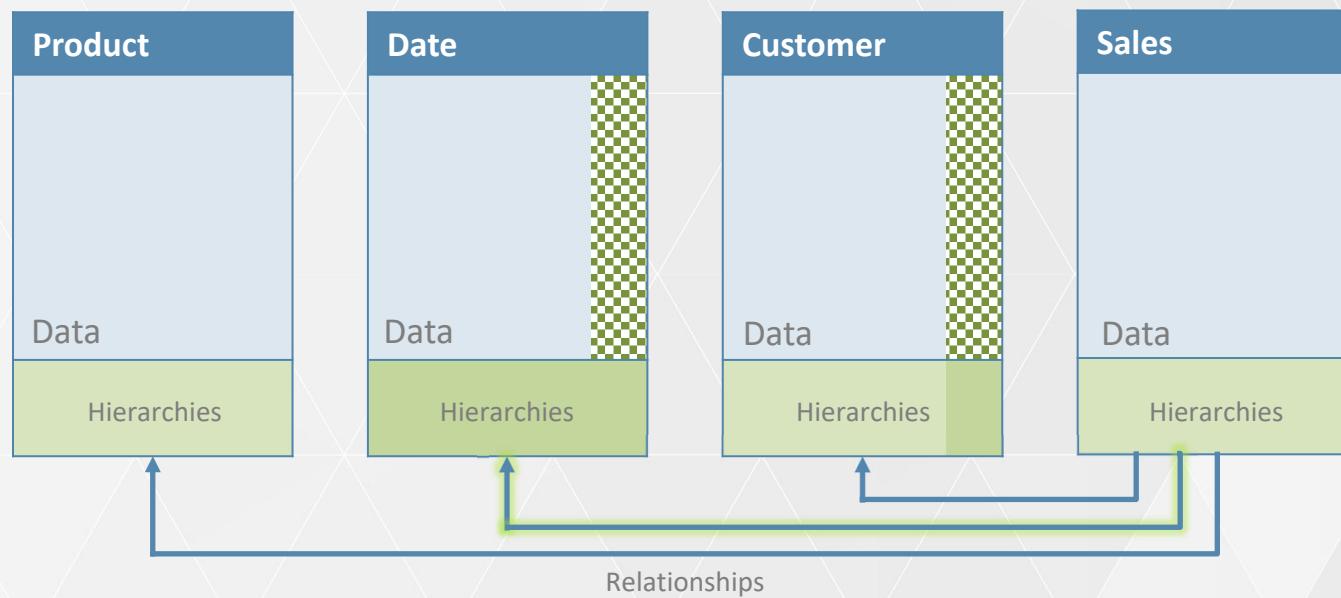
After data refresh, calculated columns and hierarchies are empty.

Moreover, calculated column in other tables might be invalidated, because they depended on the content of Date.



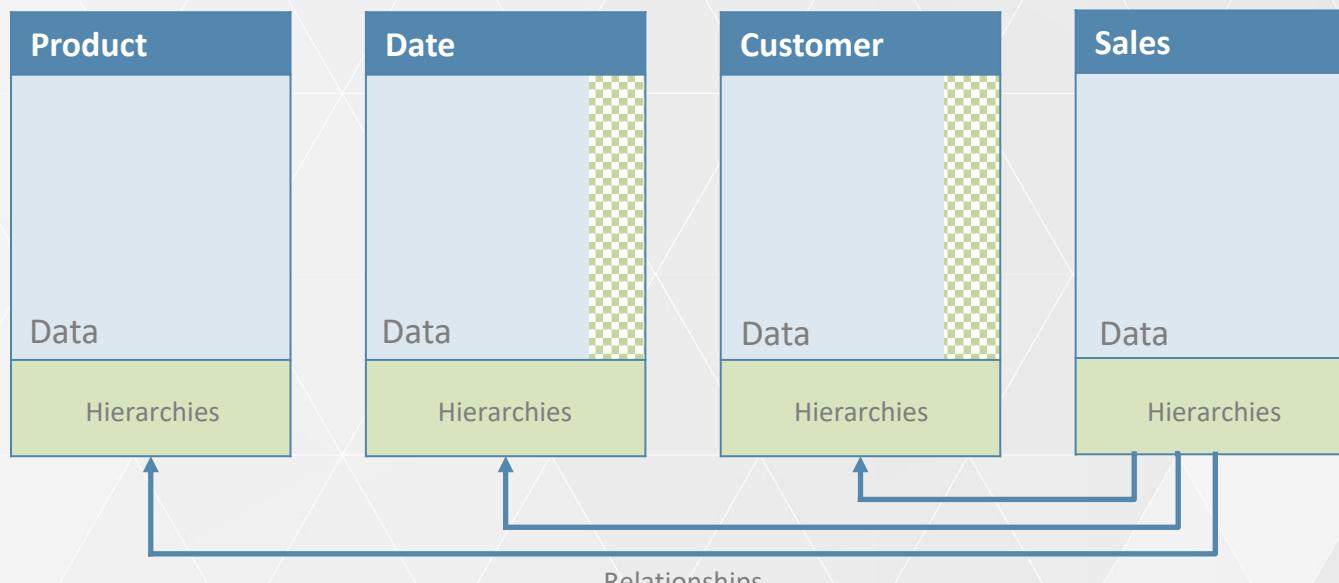
# Add a column in Date

Model recalc is needed to refresh calculated objects



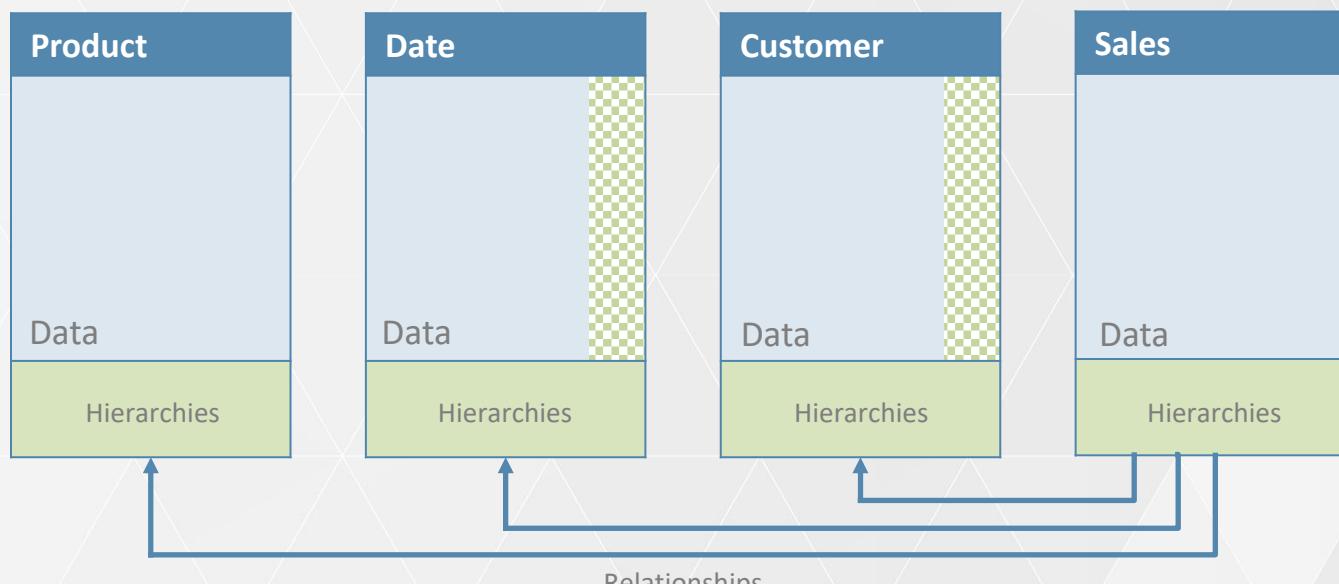
# Add a column in Date

After data processing and model recalc, the model is ready to be queried.



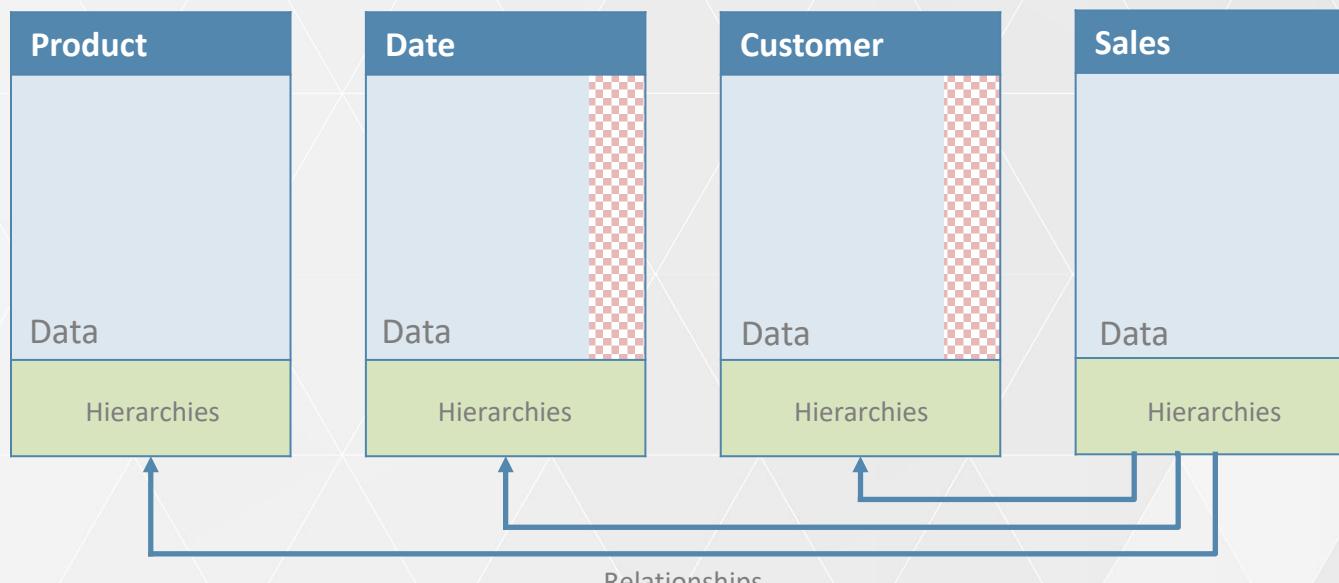
# Add/modify a calculated column in Date

A new or updated calculated column will not require a reprocess of the data. The engine can recompute the column (and all dependent objects) without accessing the data sources.



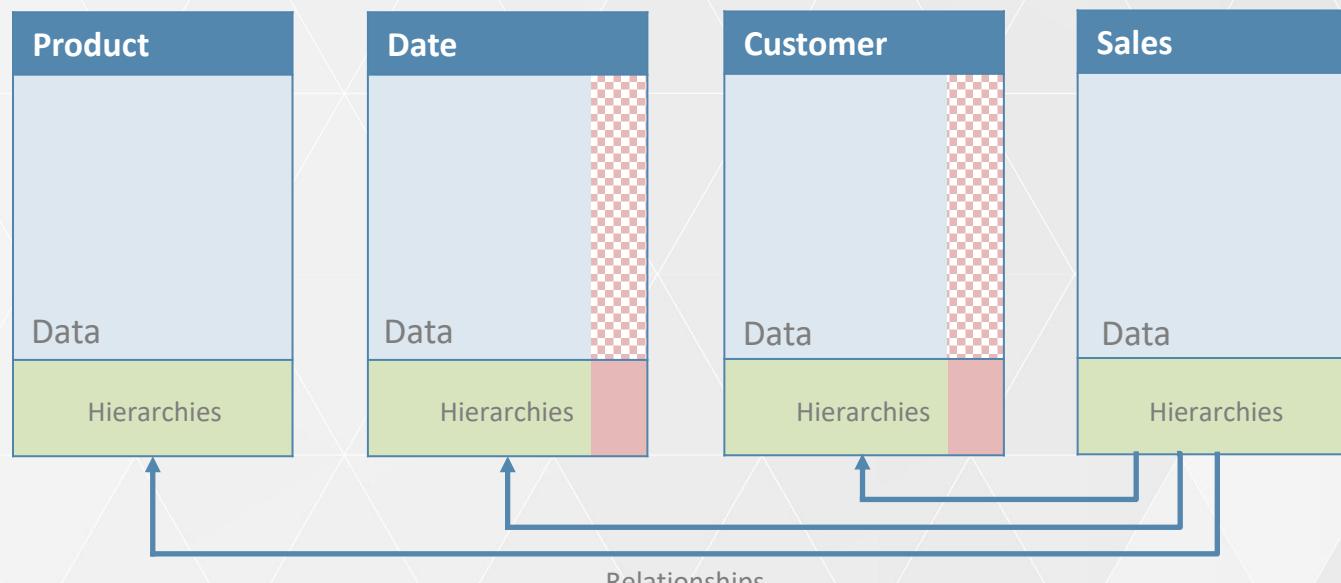
# Add/modify a calculated column in Date

Adding or editing a calculated column invalidates the column itself and all dependent objects. The dependent objects might also be in different tables; for example, a calculated column in Customer depends on the modified calculated column in Date.



# Add/modify a calculated column in Date

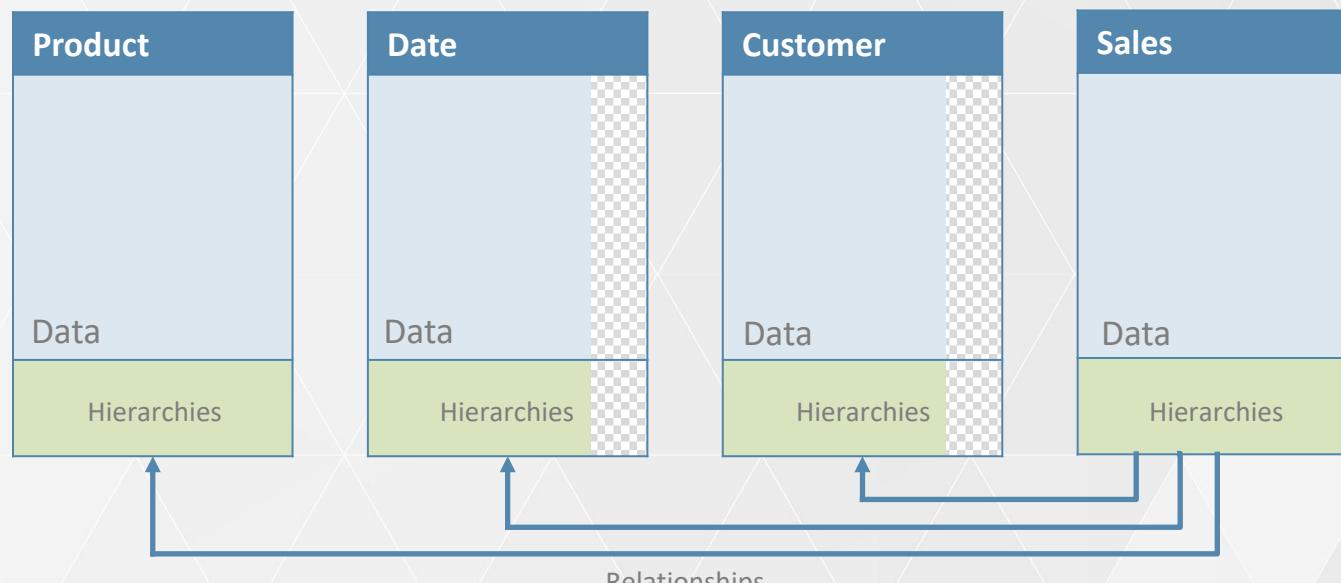
The invalid columns might invalidate hierarchies and/or relationships. We need to recalculate them.



# Add/modify a calculated column in Date

Model recalc recomputes all the invalid calculated objects in the model.

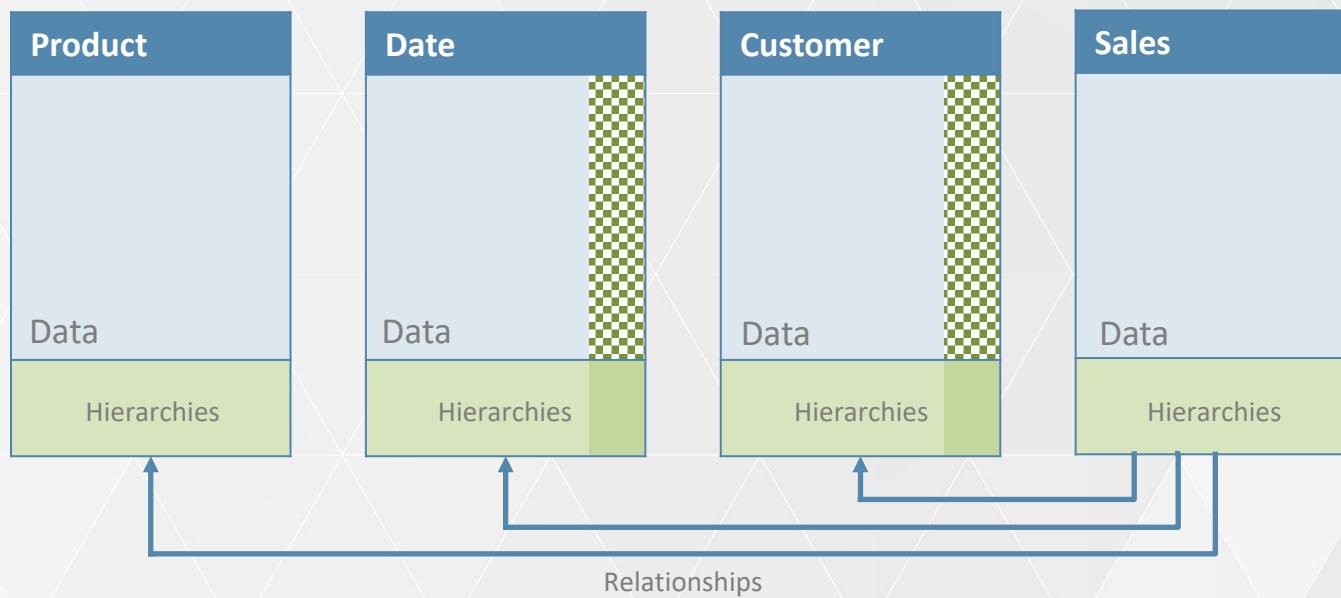
Model recalc works at the model level, we cannot recalculate only objects in a table.



# Add/modify a calculated column in Date

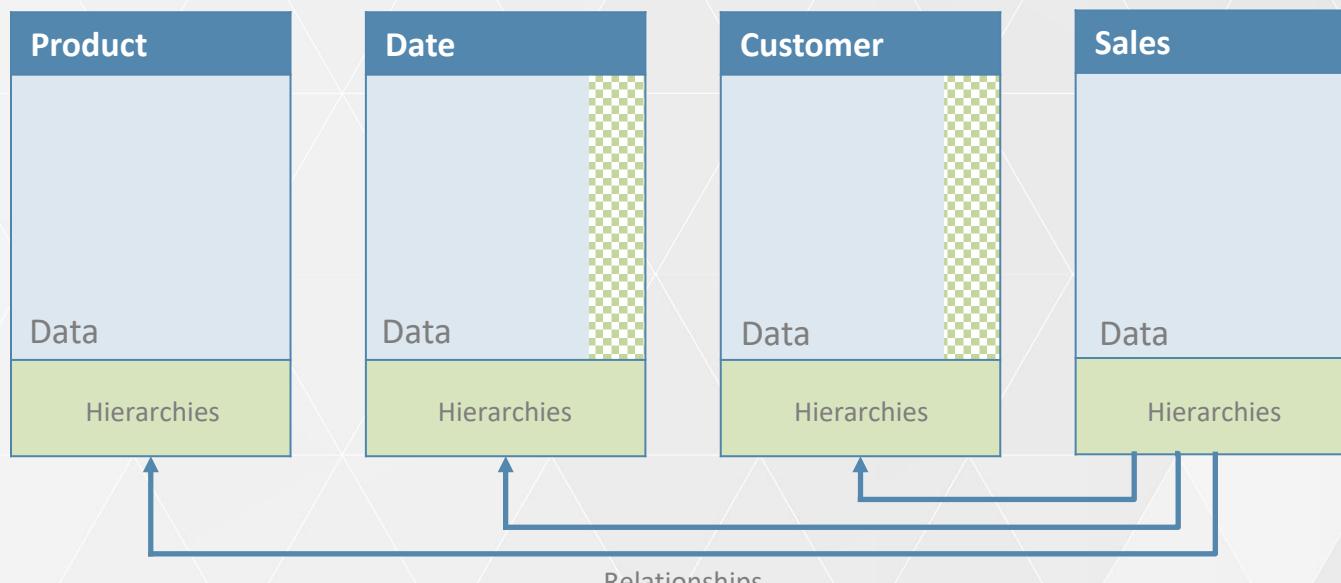
After process recalc, the invalid objects are computed.

Running process default would return the same result.



# Add/modify a calculated column in Date

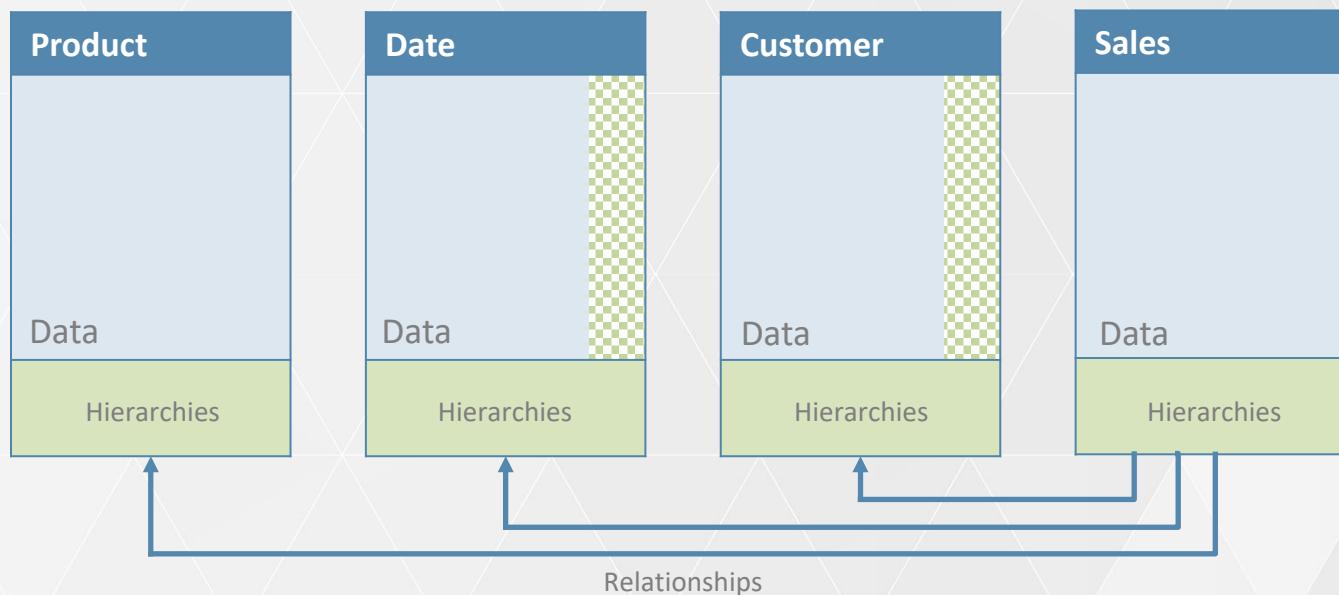
After the invalid objects are recomputed, the model is valid again.



## Processing options after deployment

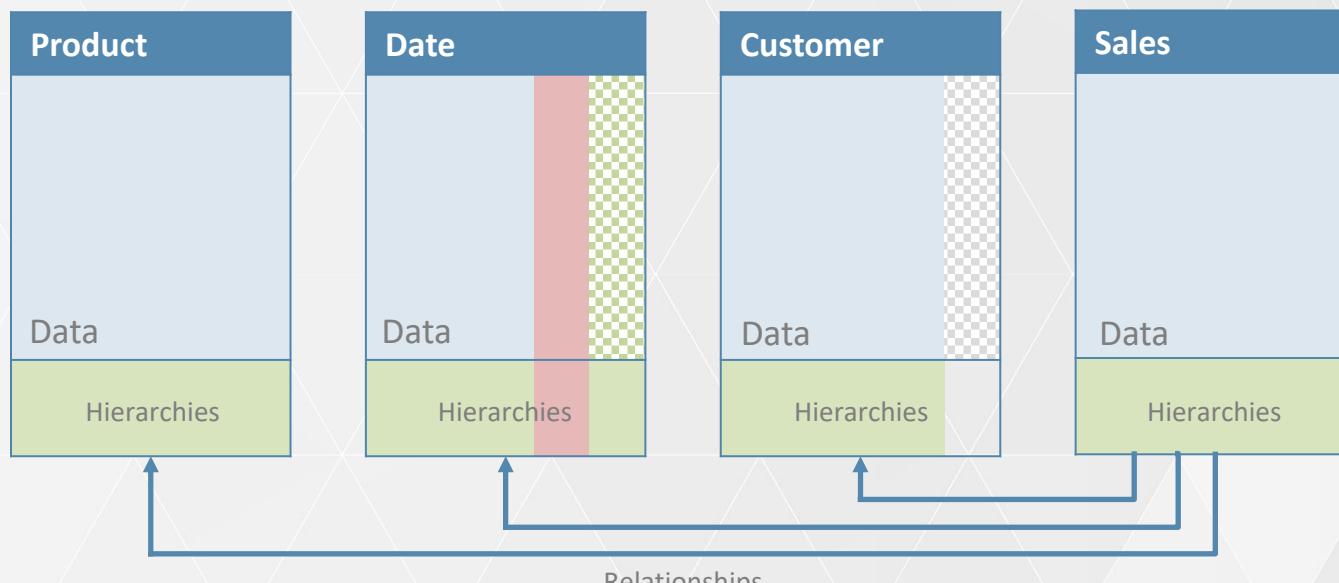
- Full
  - Clear all the tables and process the entire database
- Automatic
  - Data refresh of invalid table
  - Model recalc limited to the invalid/updated objects
- None
  - The model can be fully valid or partially valid
  - It depends on the deployment delivered
  - All the tables are empty when you deploy a new database

# Processing options after deployment



# Processing options after deployment

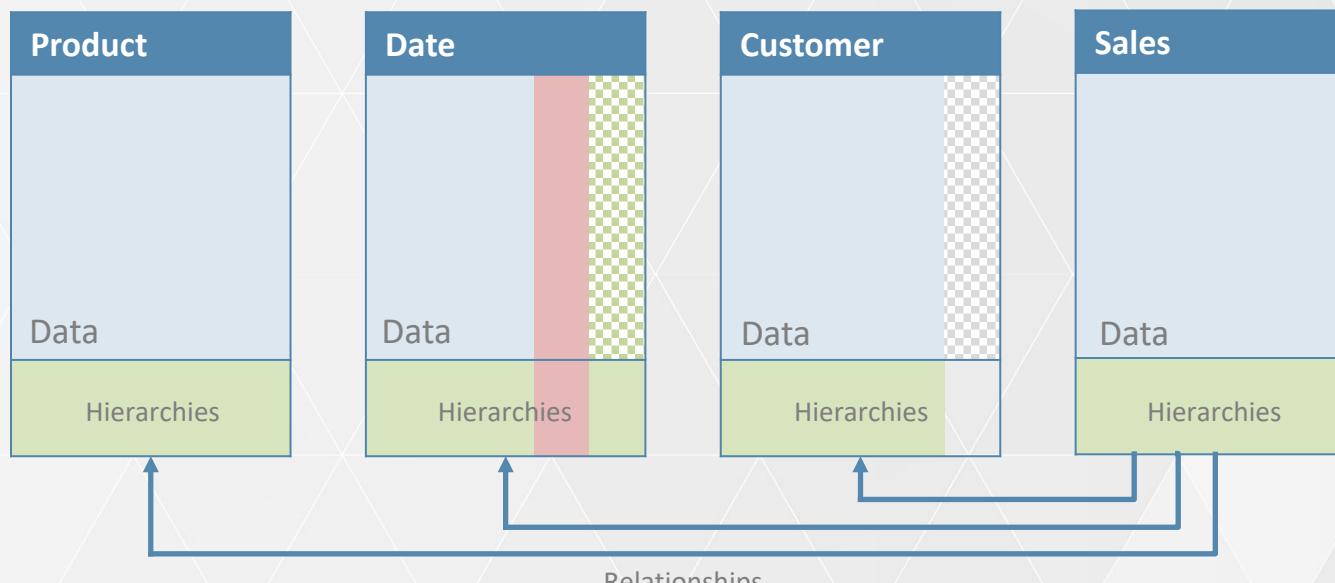
Deployment of modified columns in Date and new calculated column in Customer.



# Processing options after deployment

Deployment of modified columns in Date and new calculated column in Customer.

**Process None:** new and modified columns are not accessible. The remaining model works.

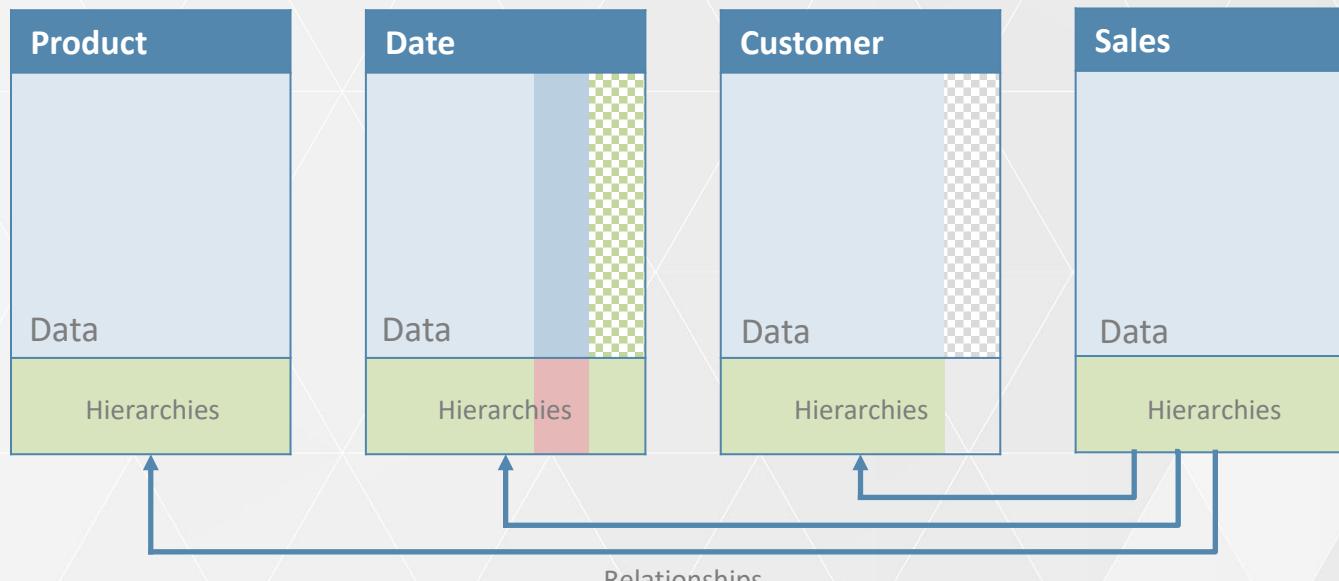


# Processing options after deployment

Deployment of modified columns in Date and new calculated column in Customer.

**Process None:** new and modified columns are not accessible. The remaining model works.

**Process Automatic:** data refresh of Date

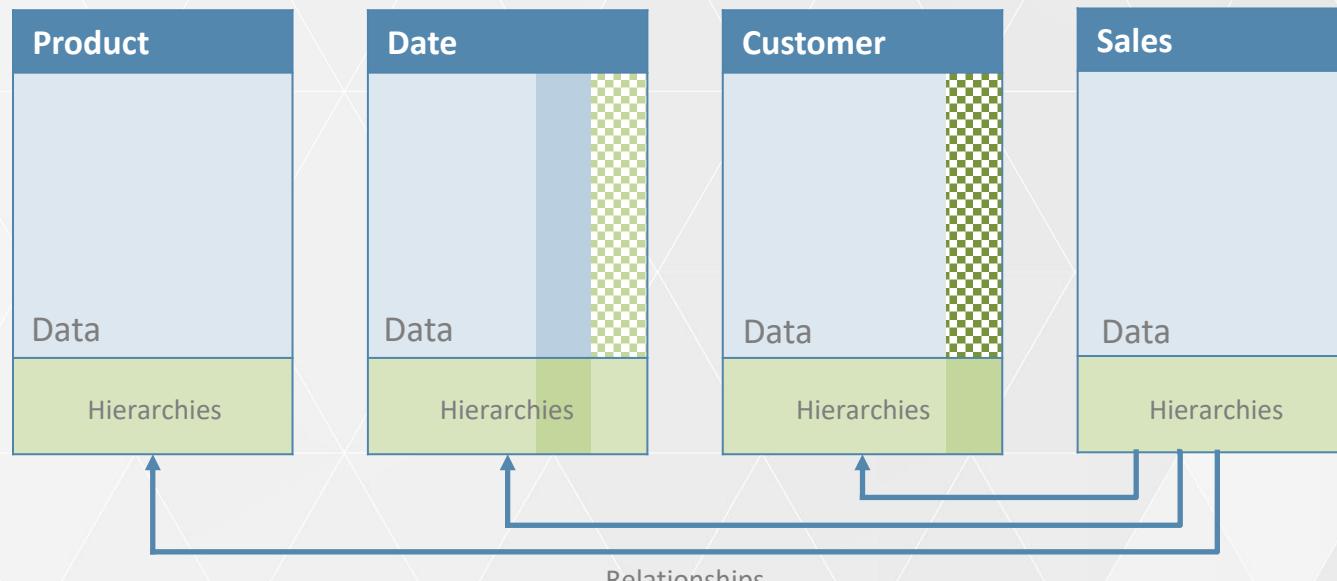


# Processing options after deployment

Deployment of modified columns in Date and new calculated column in Customer.

**Process None:** new and modified columns are not accessible. The remaining model works.

**Process Automatic:** data refresh of Date, recalc Date attribute hierarchy and Customer calculated column.

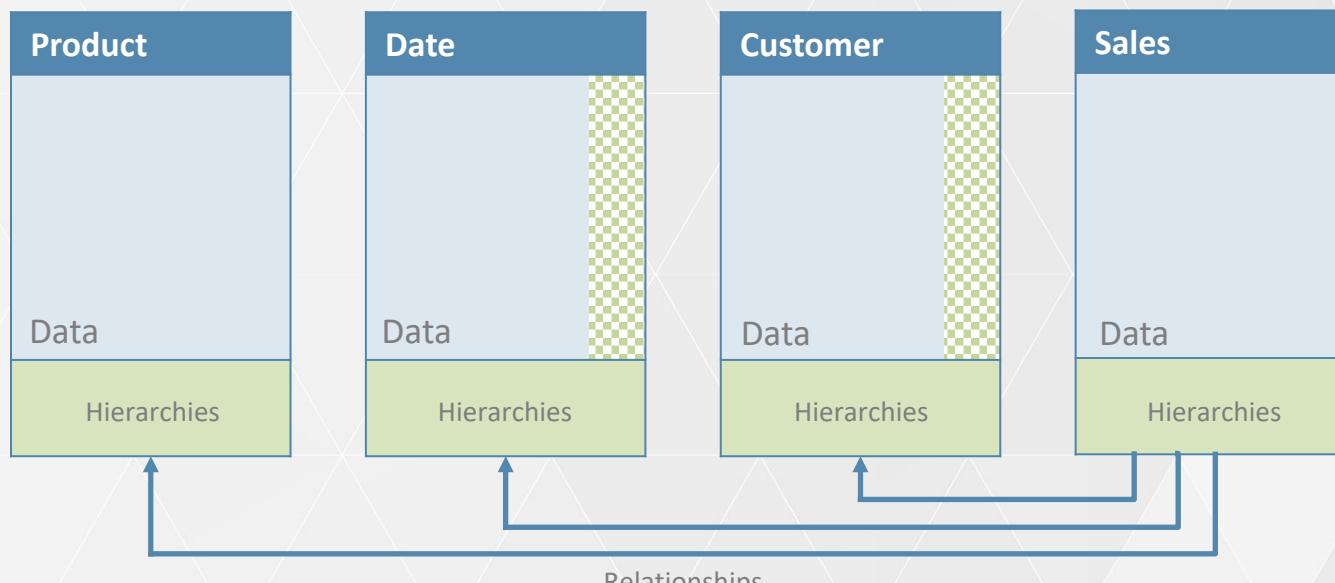


# Processing options after deployment

Deployment of modified columns in Date and new calculated column in Customer.

**Process None:** new and modified columns are not accessible. The remaining model works.

**Process Automatic:** data refresh of Date, recalc Date attribute hierarchy and Customer calculated column.



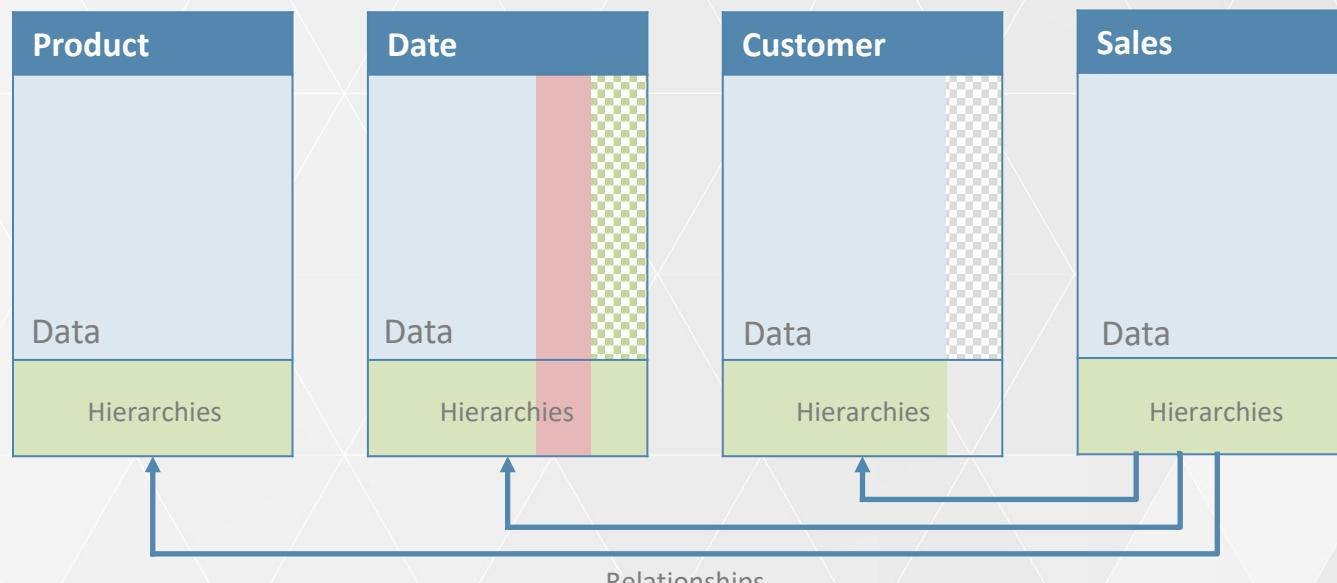
# Processing options after deployment

Deployment of modified columns in Date and new calculated column in Customer.

**Process None:** new and modified columns are not accessible. The remaining model works.

**Process Automatic:** data refresh of Date, recalc Date attribute hierarchy and Customer calculated column.

**Process Full:**



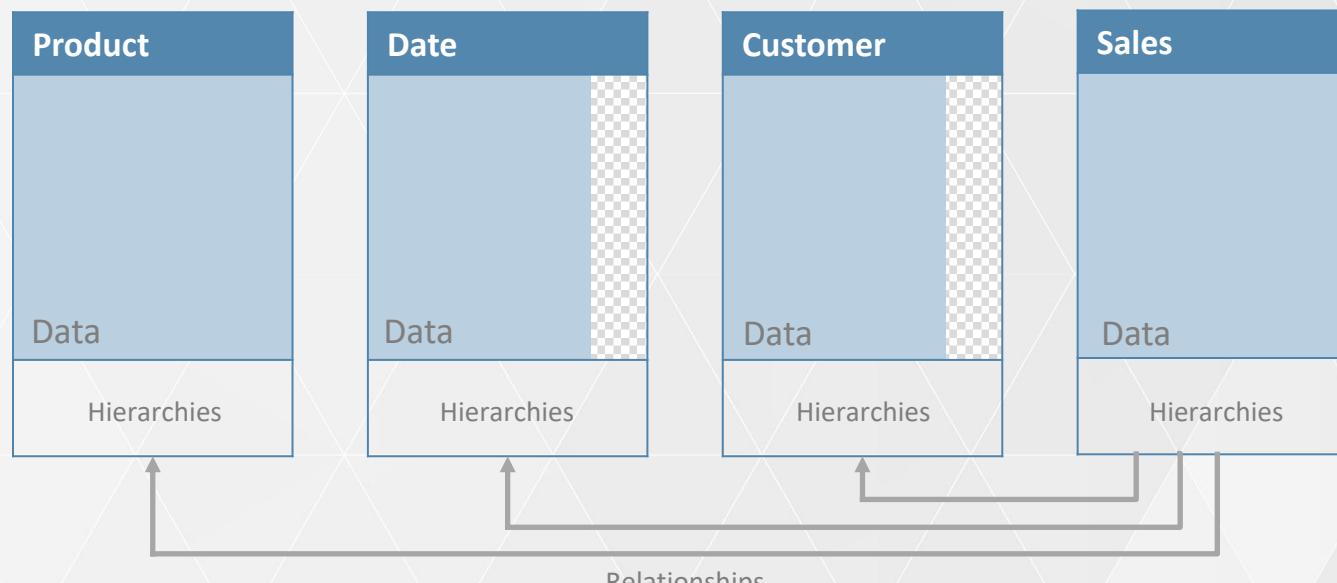
# Processing options after deployment

Deployment of modified columns in Date and new calculated column in Customer.

**Process None:** new and modified columns are not accessible. The remaining model works.

**Process Automatic:** data refresh of Date, recalc Date attribute hierarchy and Customer calculated column.

**Process Full:** data refresh of all the tables



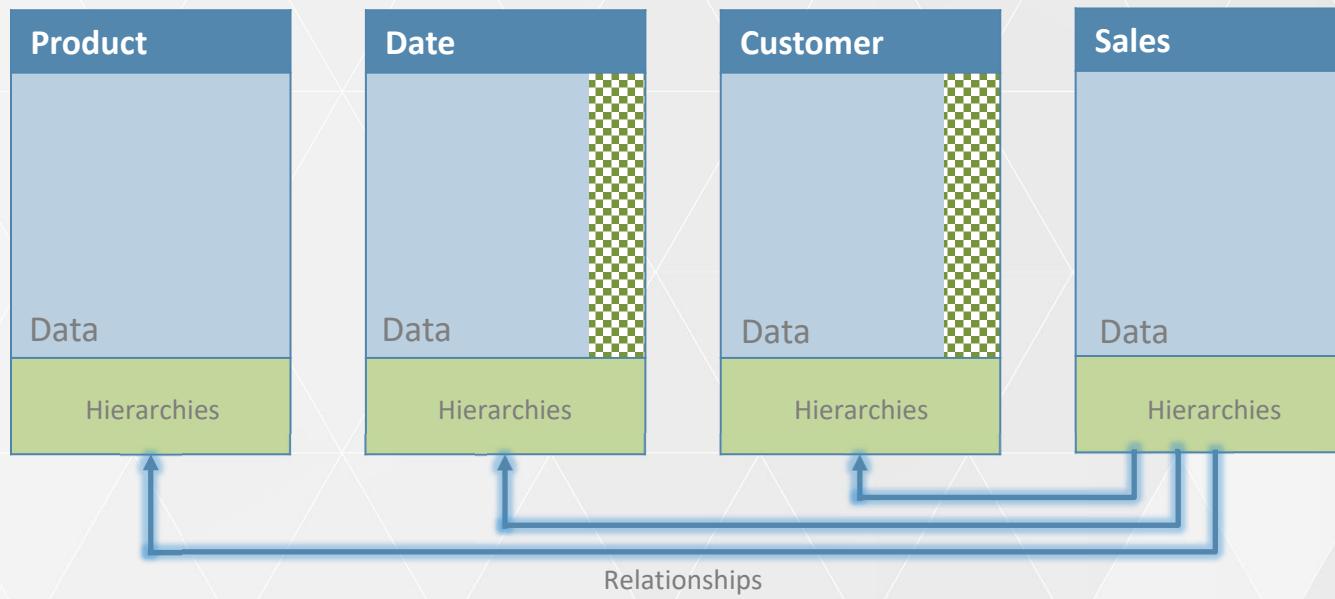
# Processing options after deployment

Deployment of modified columns in Date and new calculated column in Customer.

**Process None:** new and modified columns are not accessible. The remaining model works.

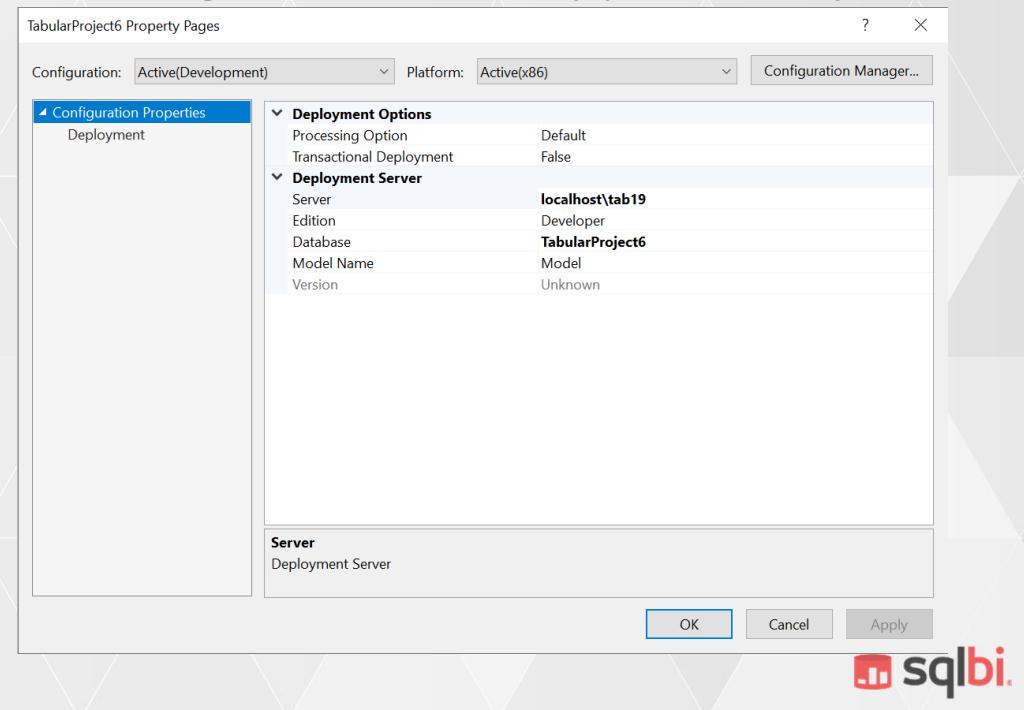
**Process Automatic:** data refresh of Date, recalc Date attribute hierarchy and Customer calculated column.

**Process Full:** data refresh of all the tables, complete model recalc.



# Deploying a solution from Visual Studio

- Deployment properties and configuration manager
  - Visual Studio configuration manager does not support Analysis Services projects
  - Properties like Server and Database name are shared by all configurations



## Deploying security roles and partitions

- Do you want to override objects when you deploy a model?
  - Security roles
  - Partitions
- Visual Studio always overrides existing roles and partitions
- Other tools can deploy keeping existing roles/partitions:
  - Analysis Services Deployment Wizard
  - Tabular Editor

# **Analysis Services Deployment Wizard**

- Converts .ASDATABASE file into TMSL
  - Creates XMLA command to deploy
  - Saves the XMLA file and/or deploys it
  - Modifies the XMLA file according to deployment options

# Deploying a model from Tabular Editor

- Tabular Editor Deployment Wizard
  - Direct deployment (Deploy button)
  - Export build: create .asdatabase file for Analysis Services Deployment Wizard
  - Create TMSL Script
- Available on both Tabular Editor 2 and Tabular Editor 3

# Continuous integration

- CI/CD with Azure DevOps, Git, and Tabular Editor  
<https://tabulareditor.github.io/2019/02/20/DevOps1.html>
- Split model into one file for each object
  - Tabular Editor as a command line tool can rebuild the model file and deploy it
  - Automate Release Pipelines in Azure DevOps
  - Enable multiple developers to work on the same model

## Other deployment techniques

- Backup/Restore
- Attach/Detach
- Manually synchronize differences
  - ALM Toolkit

## ALM Toolkit

- Compare two Analysis Services models
  - Work on both deployed databases and model.bim files
  - Also available as a Power BI external tool
- Merge differences between models
  - You can select the objects and properties to update/skip
  - It does not create data sources / connections
- Deploy options
  - Deploy to target database / updated Model.bim file
  - Create TMSL script for update (XMLA commands)

# Power BI deployment pipelines

The screenshot shows the Power BI Deployment Pipelines interface. On the left, a navigation bar includes Home, Favorites, Recent, Create, Datasets, Goals, Apps, Shared with me, Deployment pipelines (selected), Discover, Learn, Workspaces, and My workspace.

The main area displays a flow from Development to Test to Production:

- Development:** Design, review, and revise your content in a development workspace. When it's ready to test and preview, deploy the content to the test stage.
- Test:** Test and verify your content in a preproduction workspace. When it's ready to distribute, deploy the content to the production stage.
- Production:** Your content has been tested and is ready to distribute to your consumers as an app or by access to the production workspace.

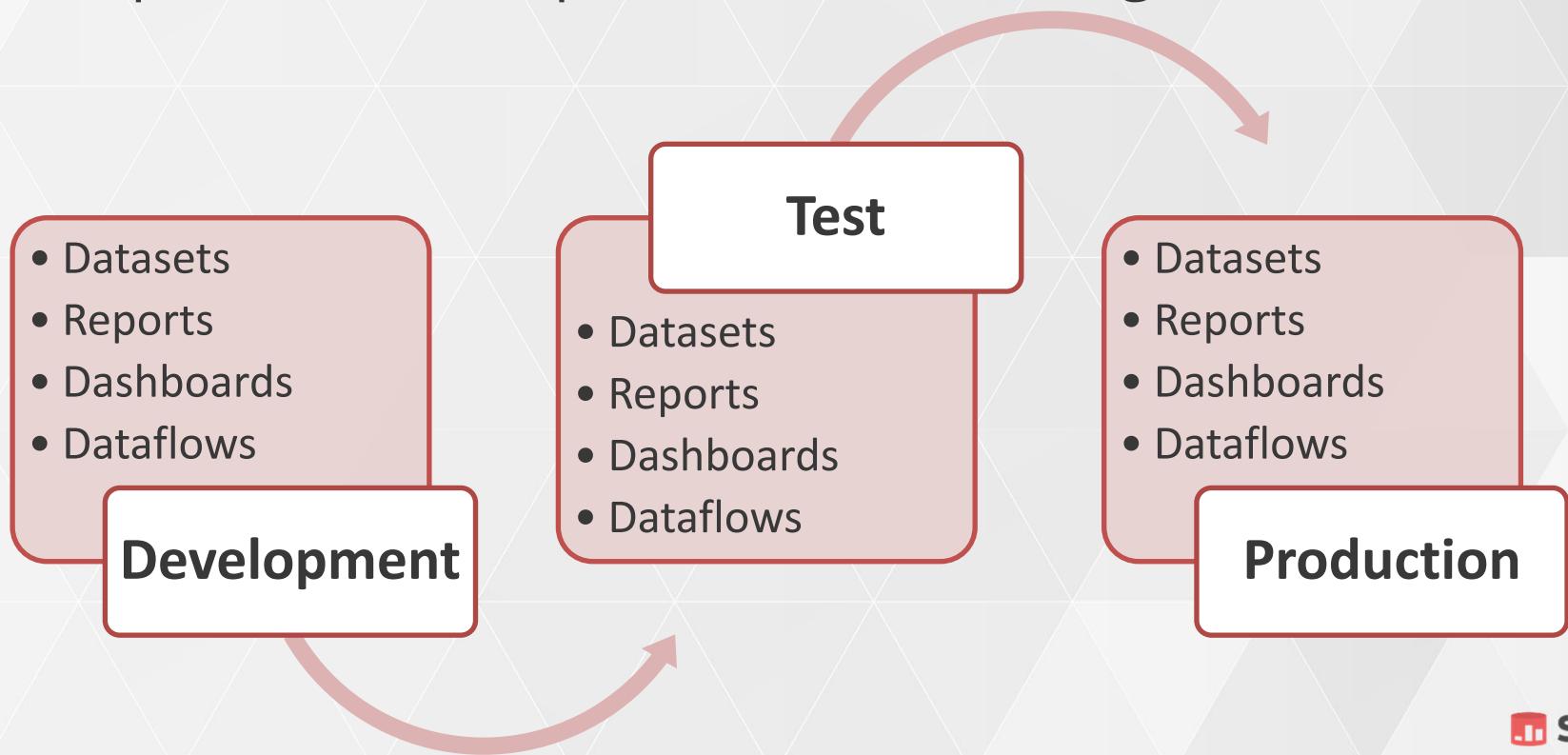
Below the stages, three cards represent the workspace content at each stage:

- Development:** Documentation demo. Contains 1 Dataflows (preview), 2 Datasets, 5 Reports, and 2 Dashboards. Includes a "Deploy to test" button.
- Test:** Documentation demo [Test]. Deployed: 6/7/2021, 10:03:13 AM. Contains 1 Dataflows (preview), 2 Datasets, 5 Reports, and 2 Dashboards. Includes a "Deploy to production" button.
- Production:** Documentation demo [Production]. Deployed: 6/14/2021, 10:49:23 AM. Contains 1 Dataflows (preview), 2 Datasets, 3 Reports, and 1 Dashboard. Includes a "Update app" button.

Each card also includes a "Compare" button and a "Show more" / "Show less" button.

# Power BI deployment pipelines

- Duplicate a workspace in different stages:



# Power BI deployment pipelines

- Dataset rules
  - Modify connection/parameter moving from one stage to another
  - Example: change source database for production stage
- Deployment tool, not a development tool
  - Deployment pipelines do not help to concurrently edit a single data model
  - Different users cannot edit a model at the same time using Visual Studio or Power BI Desktop

## Deployment of Power BI changes

- Local development using PBI Desktop
- Requirement: deploy changes only
  - Regular publish clears existing partitions
  - Long processing time using Incremental Refresh
- Compare local PBIX file with deployed model
  - Use ALM Toolkit
  - Select changes to deploy
  - ALM Toolkit is also useful for Analysis Services

What you need to know after development

# Processing and partitioning Tabular models



# Processing and Partitioning

- Table partitions
  - Creating and managing partitions
  - Side effects on processing parallelism
- Processing Tabular models
  - Processing strategies
  - Near real-time scenarios

# What is a partition?

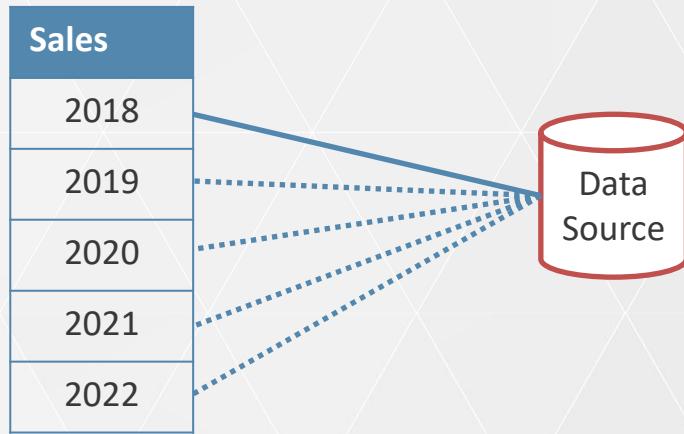
- Part of a table
  - Minimal unit of data refresh
  - Includes one or more segments
- Can reduce processing time, no impact on query time
  - Partitions can be refreshed in parallel
  - Refresh selected partitions
- Flexibility in managing data
  - Each partition can have a different data source
  - Remove selected partitions

# What is a partition?

Every partition has a different data source.

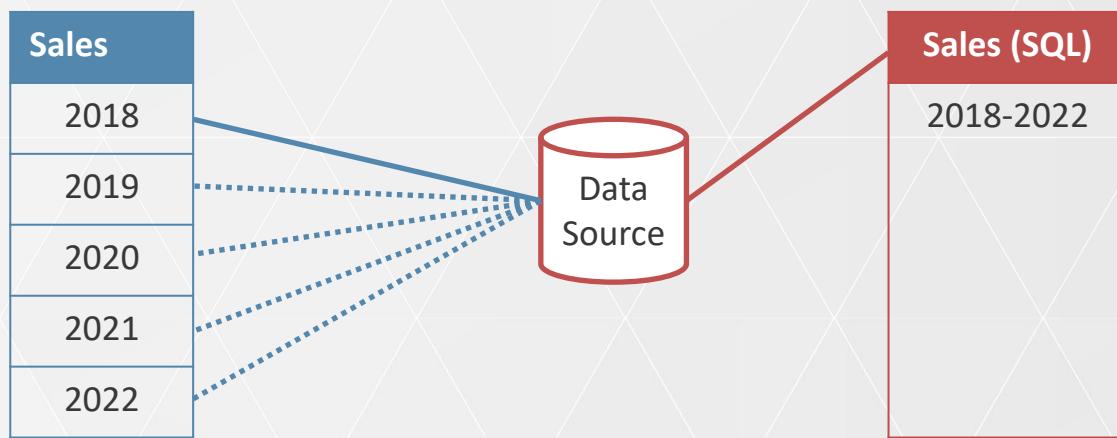
Although the connection may be the same, the query for each partition should be different.

**WARNING:** If you duplicate the query, you load the same data twice.



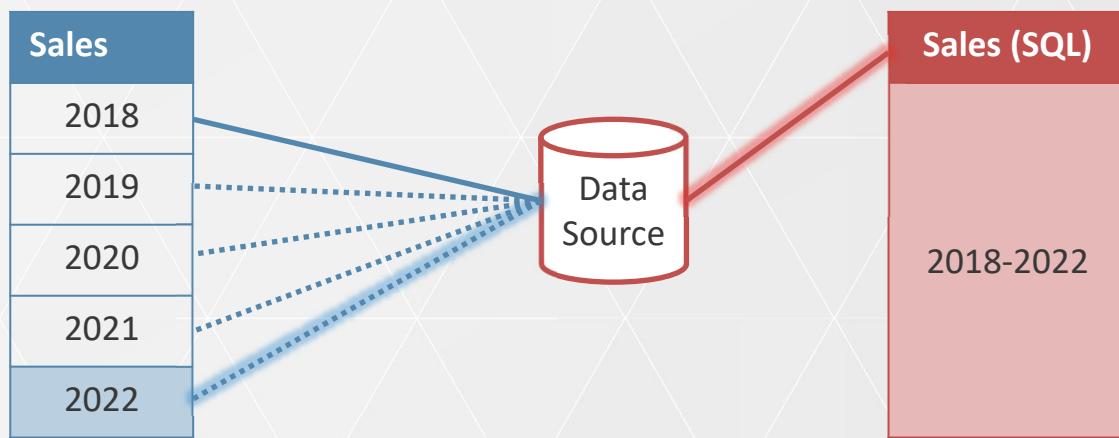
# Partitioning strategies

The data source should provide better performance when the query extracts a single partition.



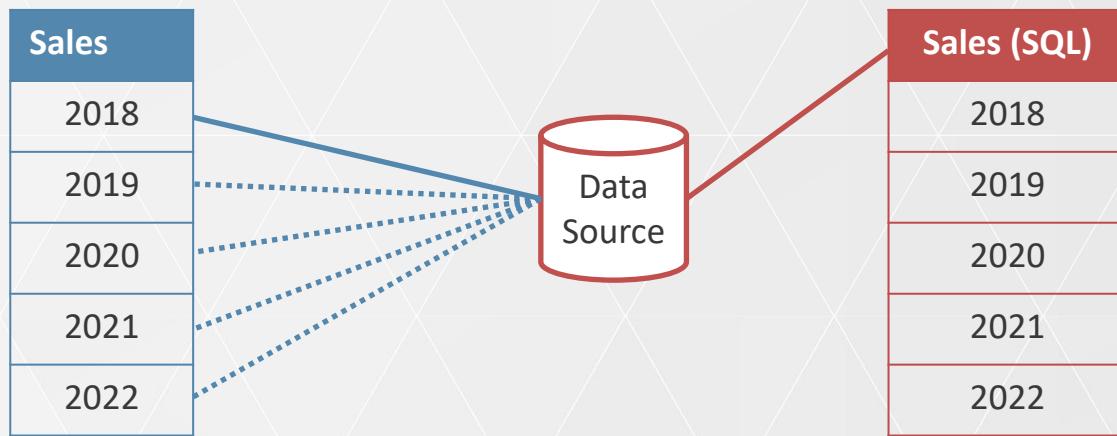
# Partitioning strategies

The data source should provide better performance when the query extracts a single partition.



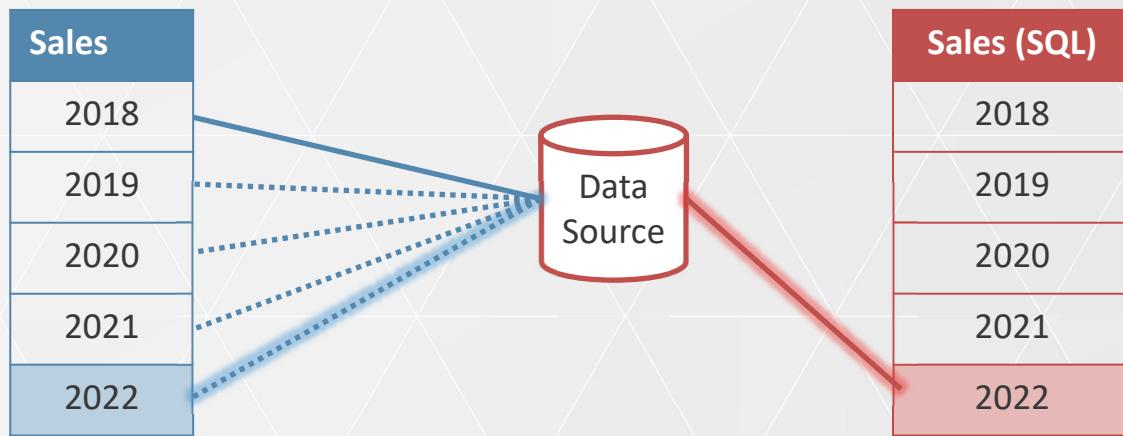
# Partitioning strategies

The data source should provide better performance when the query extracts a single partition. For example, a table in SQL Server should have a similar partitioning scheme.



# Partitioning strategies

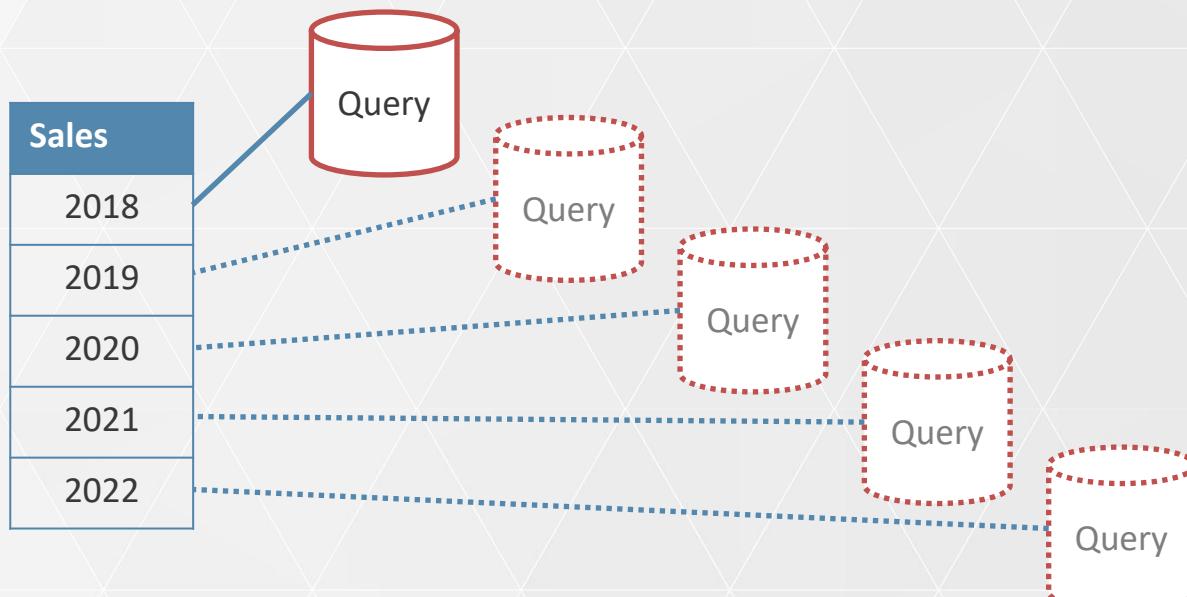
The data source should provide better performance when the query extracts a single partition. For example, a table in SQL Server should have a similar partitioning scheme.



# Partitioning strategies

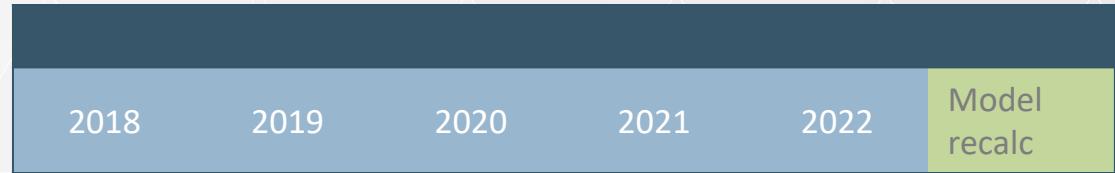
Increase parallelism during processing.

Consider only when the data source can manage a larger concurrent workload.



# Improve processing time by using partitions

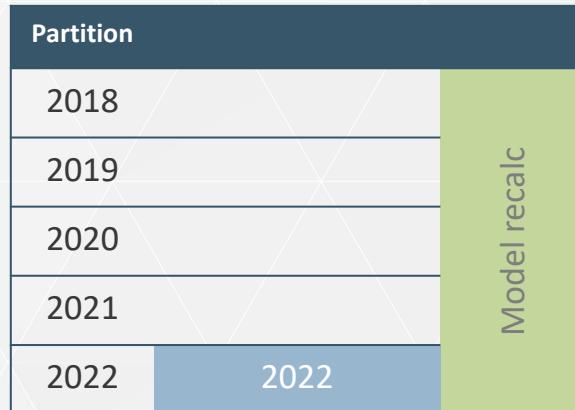
A table without partitions is processed sequentially.



Processing time

# Refresh modified partitions

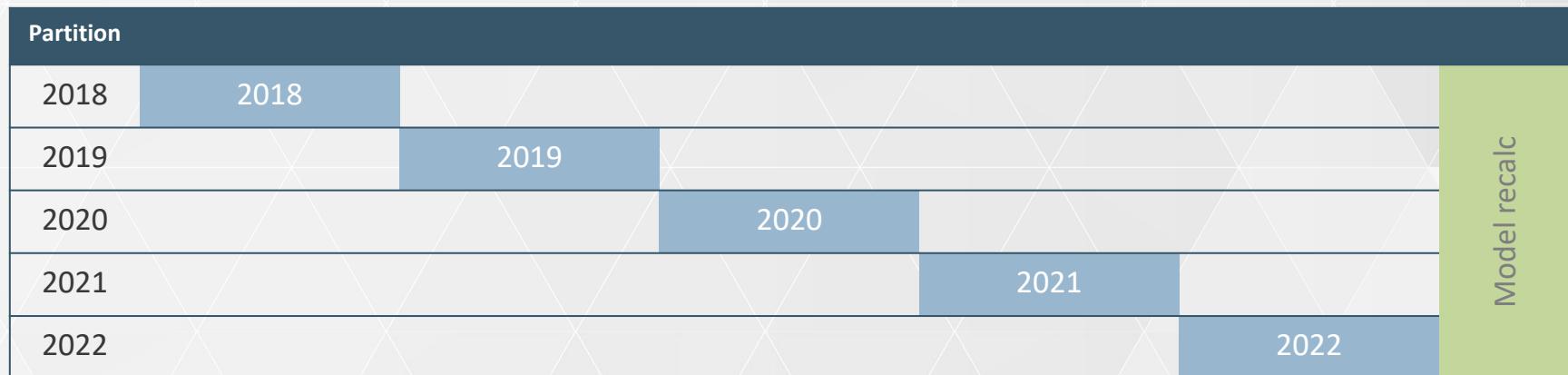
If partition by time, only refresh the last/current partition.



# Increase parallelism during processing

One partition per year can be processed in parallel depending on the maxParallelism setting.

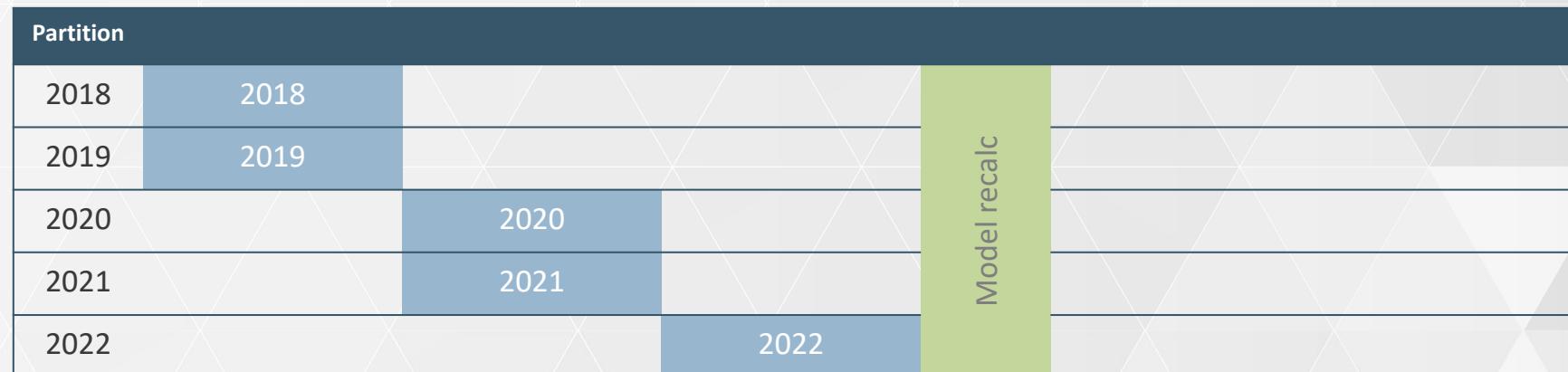
**maxParallelism = 1**



# Increase parallelism during processing

Increasing parallelism reduces processing time and increases memory requirements to refresh the data.

**maxParallelism = 2**



# Increase parallelism during processing

There is one partition per year, and multiple partitions can be processed in parallel depending on maxParallelism.

**maxParallelism = 3**

Partition			
2018	2018		
2019	2019		
2020	2020		
2021		2021	
2022		2022	

Processing time



# Increase parallelism during processing

There is one partition per year, and multiple partitions can be processed in parallel depending on maxParallelism.

**maxParallelism = 5**

Partition		
2018	2018	
2019	2019	
2020	2020	
2021	2021	
2022	2022	Model recalc

Processing time  
→

# Controlling parallelism in TMSL scripts

Use **sequence** to limit parallelism in a single operation.

This script processes no more than two partitions at the same time.

```
{  
  "sequence": {  
    "maxParallelism": 2,  
    "operations": [  
      {  
        "refresh": {  
          "type": "full",  
          "objects": [  
            {  
              "database": "Contoso"  
            }  
          ]  
        }  
      }  
    ]  
  }  
}
```

# Remove older partitions

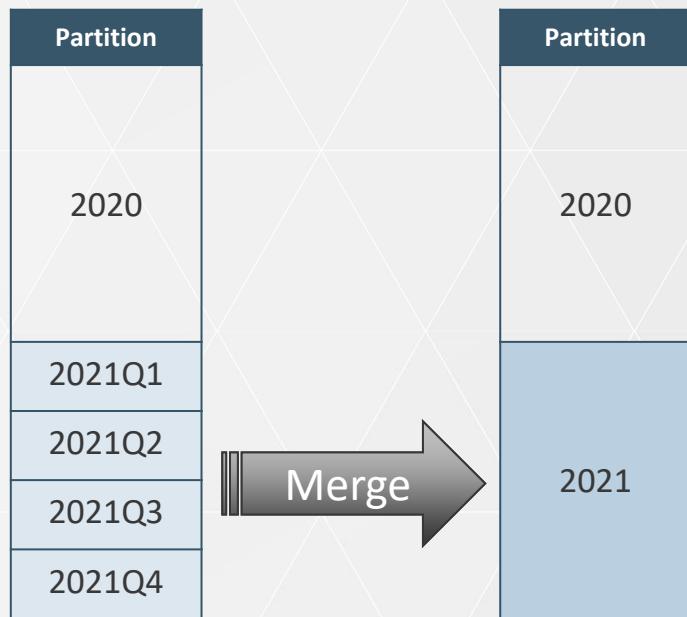
Partitions can be removed without requiring data refresh, to reduce the table size.  
Hierarchies must still be updated (model update required).

Partition	
2018	
2019	
2020	Model recalc
2021	
2022	

Processing time  
→

# Merge partitions

Partitions can be merged without accessing the data source. Segments are merged too.  
Consolidating small partitions reduces the number of small segments.  
No need to run model recalc.



# Incremental refresh in Power BI

- In Power BI, incremental refresh manages table partitions automatically
  - Partitions dynamically created and maintained based on incremental refresh rule
  - Latest partitions automatically refreshed
  - Partitions should not be modified manually
  - Partitions can be processed manually

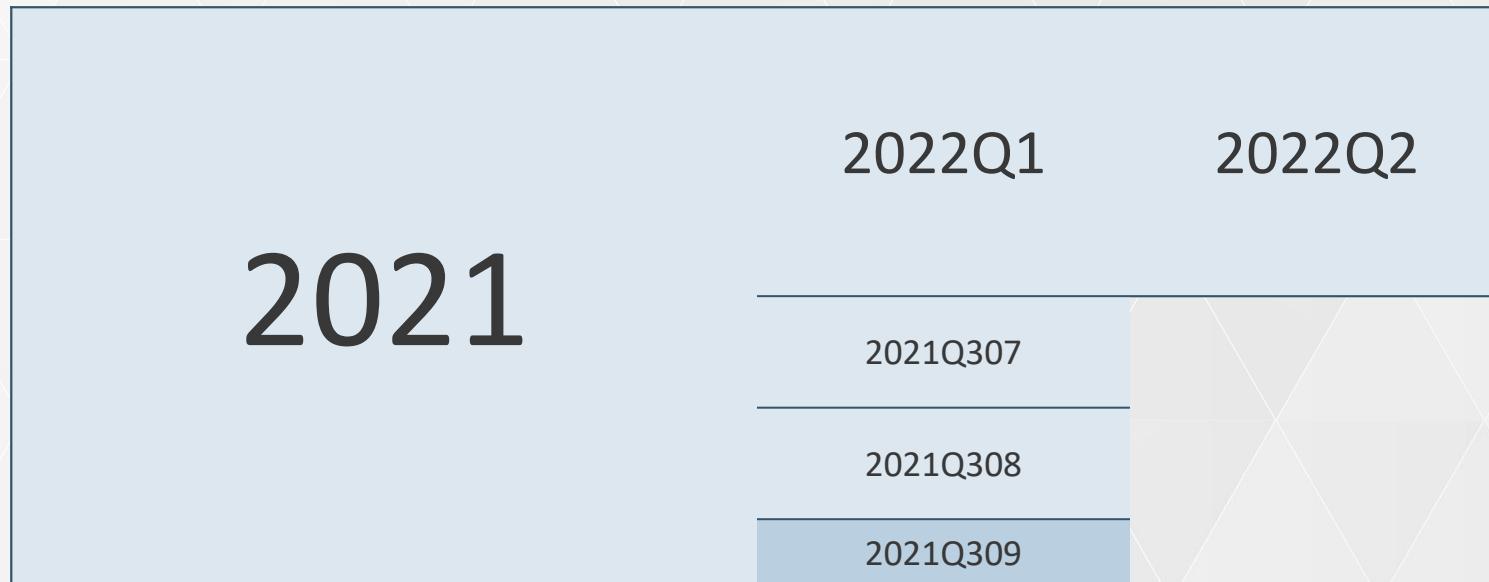
# Incremental refresh in Power BI

The incremental refresh automatically creates/merges/removes partitions according to the policy defined.



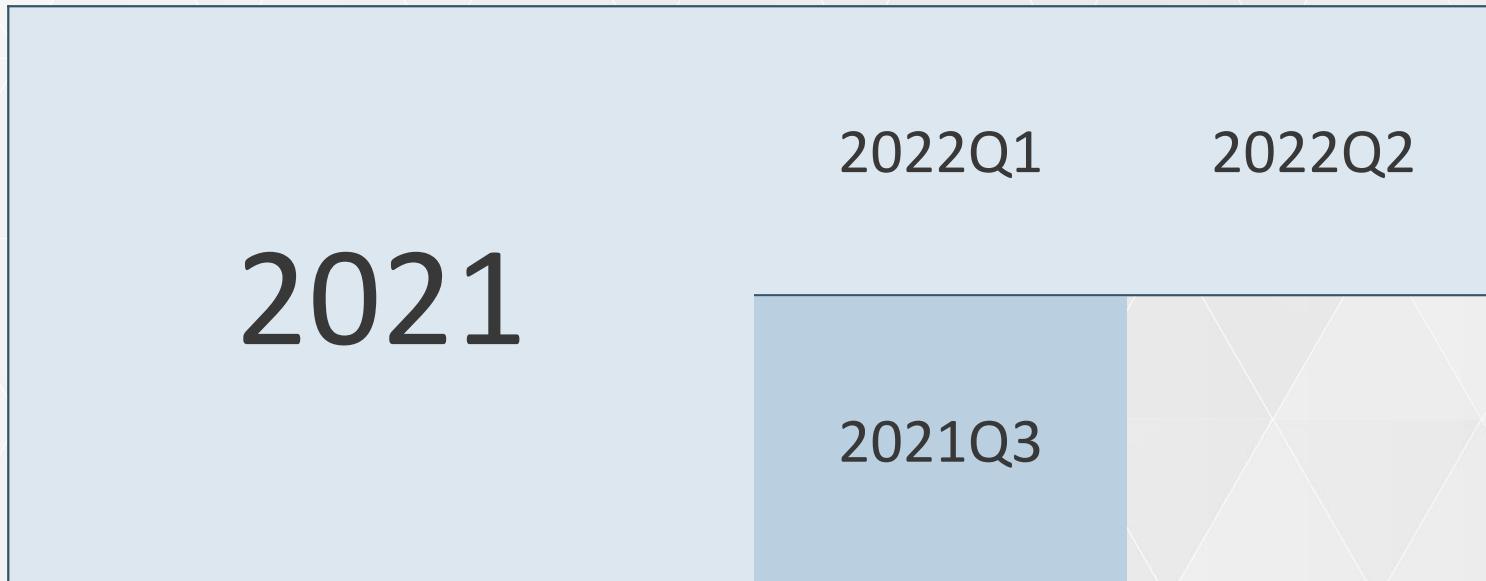
# Incremental refresh in Power BI

The incremental refresh automatically creates/merges/removes partitions according to the policy defined.



# Incremental refresh in Power BI

The incremental refresh automatically creates/merges/removes partitions according to the policy defined.



# Automate partition management

- GitHub open-source project:  
<https://github.com/Microsoft/Analysis-Services/tree/master/AsPartitionProcessing>
- Works for Azure AS and SSAS
- Starting point for incremental refresh implementation in Power BI
- Alternative: create your own script in C# / PowerShell (TOM)

## Partitioning points of attention

- Increase memory required to process a model
  - Parallelism requires more buffers to read data
- Increase the number of small segments
  - Small partitions generate small segments
  - Consider merging partitions
- Load duplicated data
  - No control over duplicated rows across partitions
  - Unique / Key constraints guarantee uniqueness
  - Fact tables should not have keys (storage/processing cost)

# Processing options

Option	Database	Table	Partition
Process Clear	Available	Available	Available
Process Default	Available	Available	Available
Process Full	Available	Available	Available
Process Recalc	Available	<b>Not in UI</b>	
Process Defrag	<b>Not in UI</b>	Available	
Process Data	<b>Not in UI</b>	Available	Available
Process Add			<b>Not in UI</b>

# Processing options

Process Table(s)

Select a page

General

Script ▾ Help

Select the processing mode and the objects to process.

Mode: Process Default

- Process Default
- Process Full
- Process Data
- Process Clear
- Process Defrag
- Name

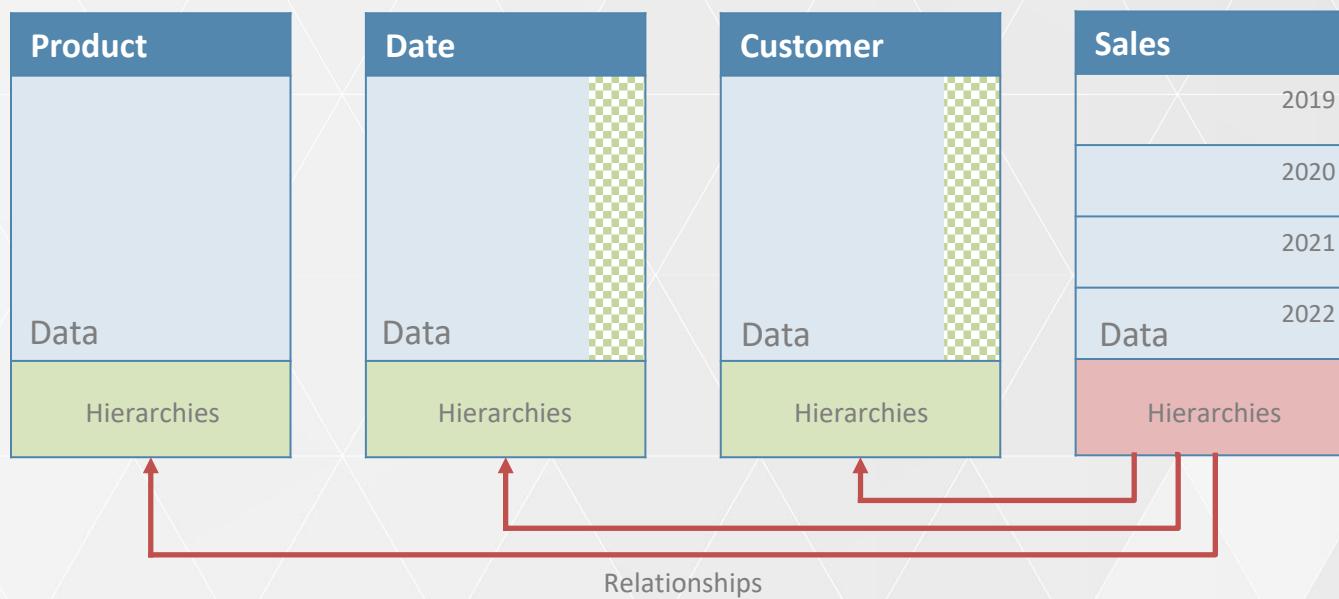
- Customer
- Date
- Sales
- Product
- Currency

Connection

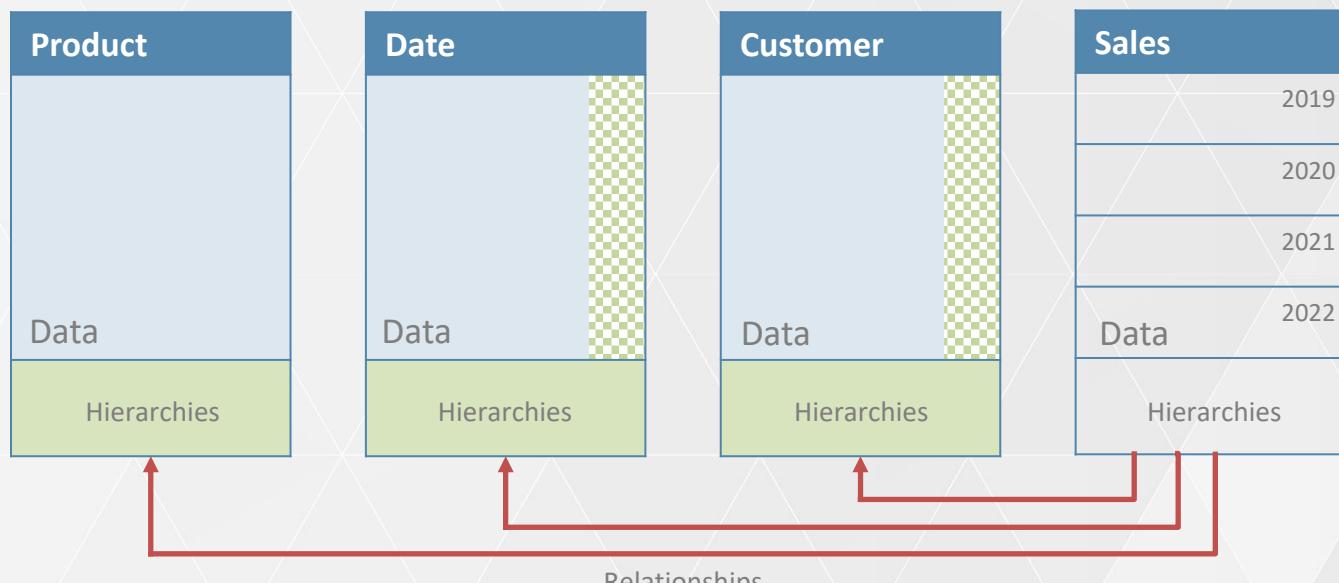
Server: .\tab19

Connection:

# Process Clear: partition 2019 (Sales)

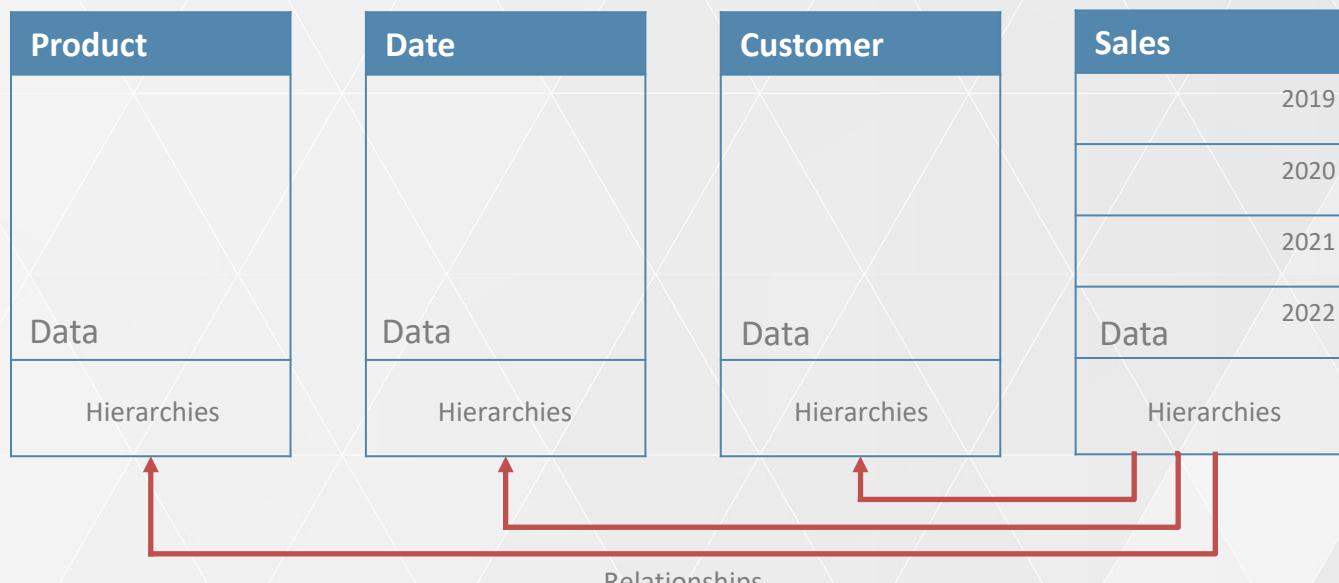


# Process Clear: Sales table

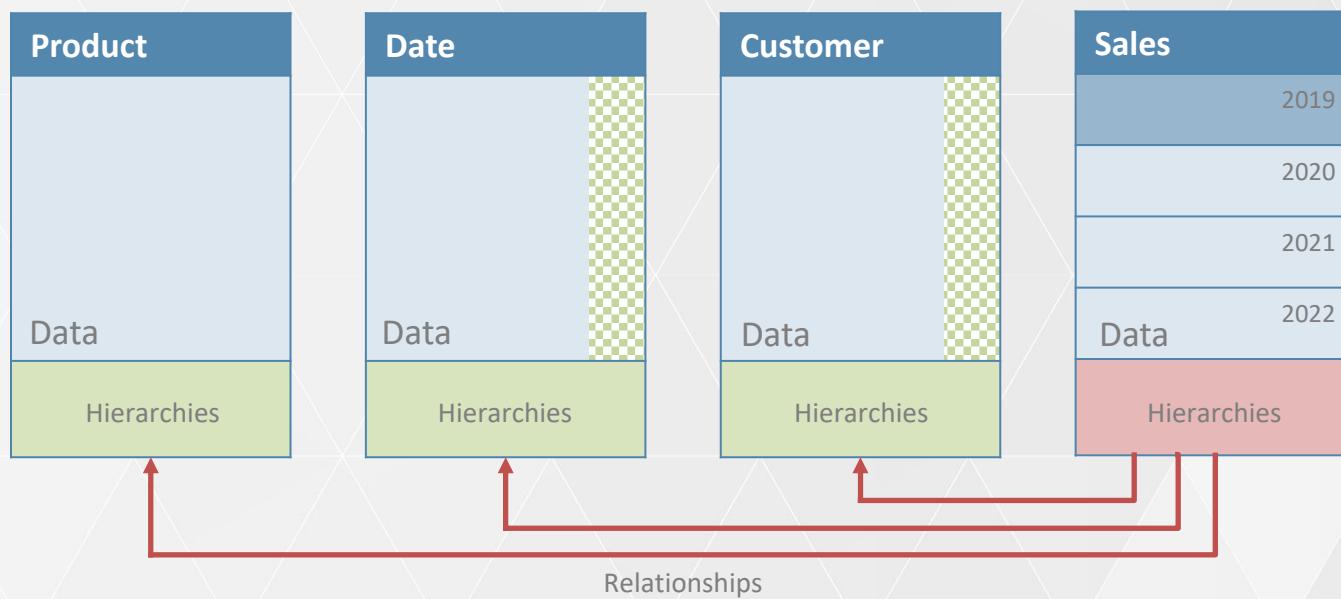


# Process Clear: database

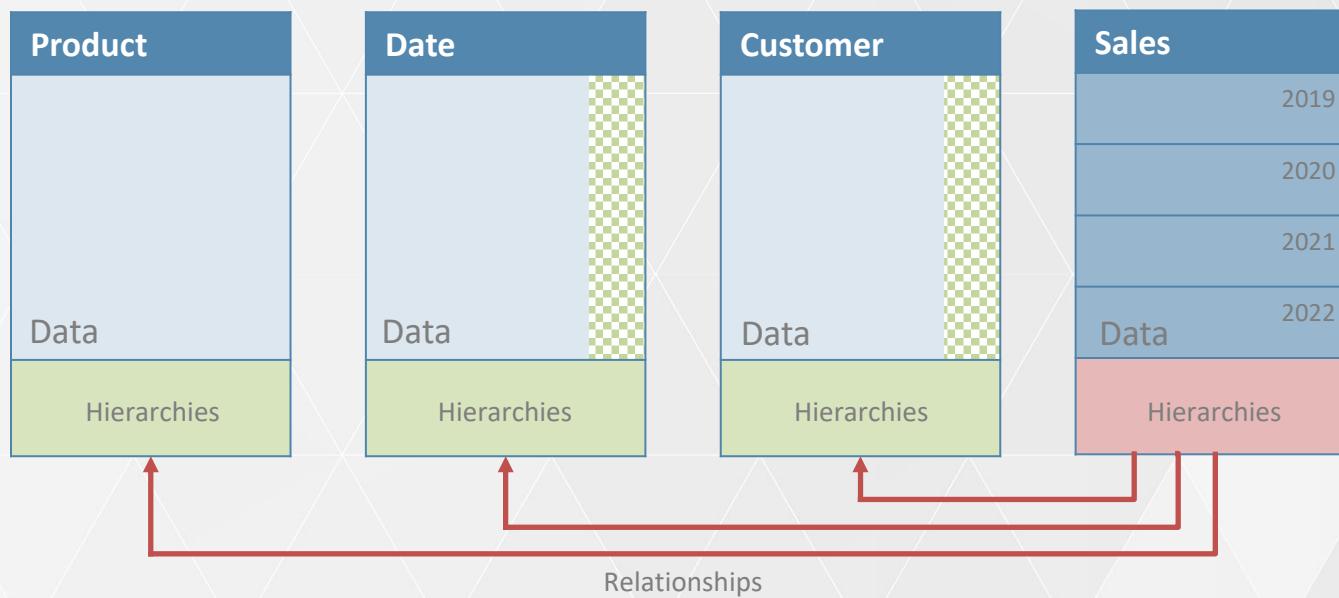
When you deploy a model with no processing, all the partitions of all the tables are empty, just like after a clear command.



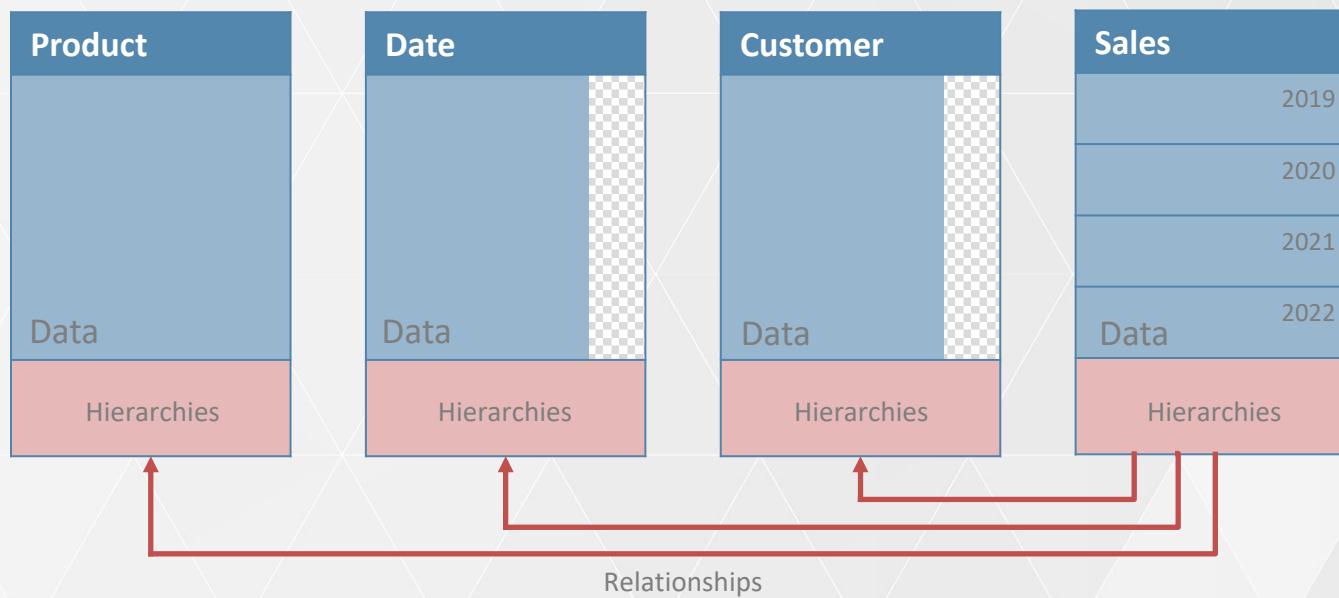
# Process Data: partition 2019 (Sales)



# Process Data: Sales table



# Process Data: database



## Process Recalc

- Compute the attribute hierarchies and the user hierarchies
- Compute the calculated columns and calculated tables
- Compute the relationships

## Process Default

- Process Data for the objects that are in clear state
  - Partition: only the partition requested if it is in clear state
  - Table: all the “clear” partitions of the table
  - Database: all the “clear” partitions of the database
- Perform a process recalcl limited to the tables affected by the previous step (or all the tables if database):
  - Compute attribute hierarchies and user hierarchies
  - Compute calculated columns and calculated tables
  - Compute relationships

## Process Full

- Equivalent to Process Data + Process Default
- Possible overhead when applied to different objects
  - Partitions:
    - Recompute attribute and user hierarchies
    - Recompute calculated columns and calculated tables
    - Recompute relationships
  - Tables:
    - Recompute calculated columns and calculated tables
    - Recompute relationships

## Process Defrag

- Rebuild the dictionary of the specified tables
  - The dictionary is always good if you always refresh data at the table level
  - The dictionary is updated incrementally when you add/remove/update partitions
  - Unused entries in the dictionary consume memory and slow down performance
- Requires memory to reprocess the dictionary
- Schedule a periodic Process Defrag for tables with partitions

# Process Add

- Adds rows to an existing partition
  - Loads data in a temporary partition
  - Merges the temporary partition to the existing partition
  - Use an ad-hoc partition to add rows
- Not available in the user interface
  - Specify an alternate data source to read additional data
  - Legacy data sources:  
<https://www.sqlbi.com/articles/using-process-add-in-tabular-models/>
  - Structured data sources:  
<https://powerbi.microsoft.com/en-us/blog/using-out-of-line-bindings-in-power-bi/>

# TMSL commands – sequence and parallelism

The **sequence** command runs a consecutive set of operations in batch mode on an instance of Analysis Services. The entire command and all of its component parts must complete in order for the transaction to succeed.

The tasks specified in each **refresh** operation are executed in parallel. The default behavior is to use as much parallelism as possible.

The **maxParallelism** setting specifies a limit to the parallelism used in the Refresh command.

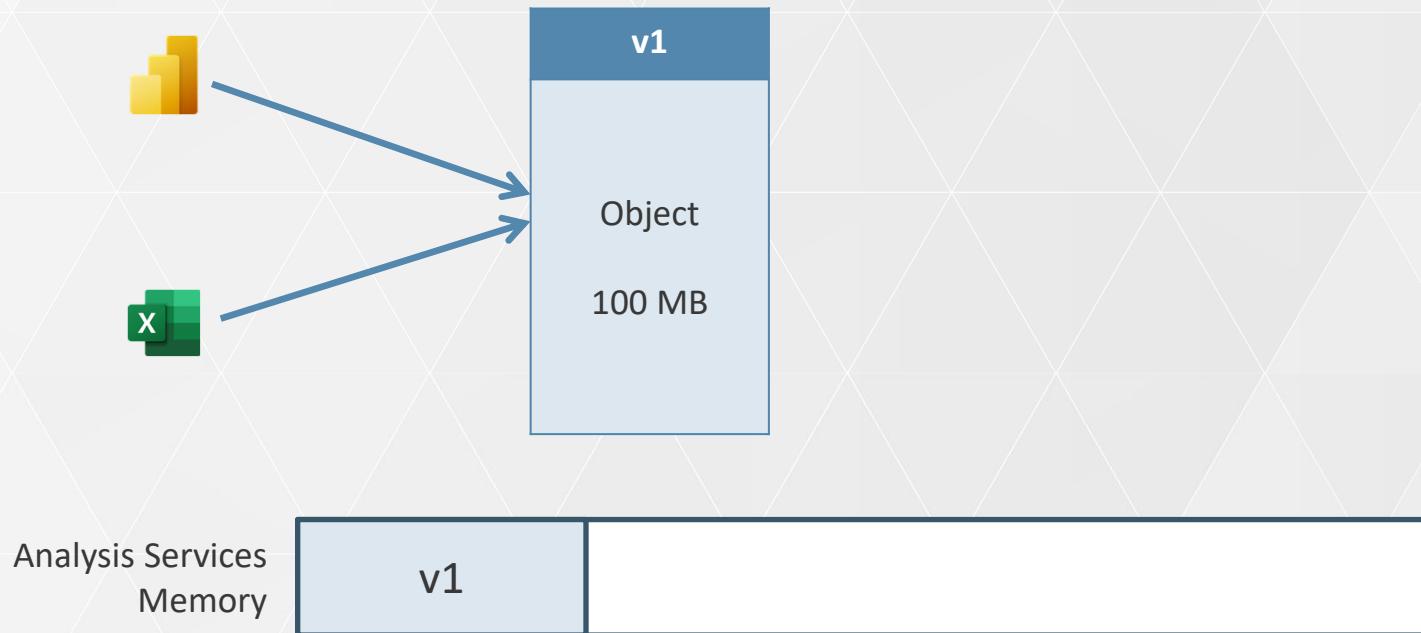
```
{ "sequence": { "maxParallelism": 10, "operations": [ { "refresh": { "type": "dataOnly", "objects": [ { "database": "Contoso", "table": "Sales" }, { "database": "Contoso", "table": "Customer" } ] }, { "refresh": { "type": "calculate", "objects": [ { "database": "Contoso" } ] } } ] } }
```

# Processing in Analysis Services

- Processing objects inside a transaction
  - Previous versions kept in memory
  - New versions processed
  - Switch-in takes place after commit
- Database can be queried during processing
- Memory needed to store old versions

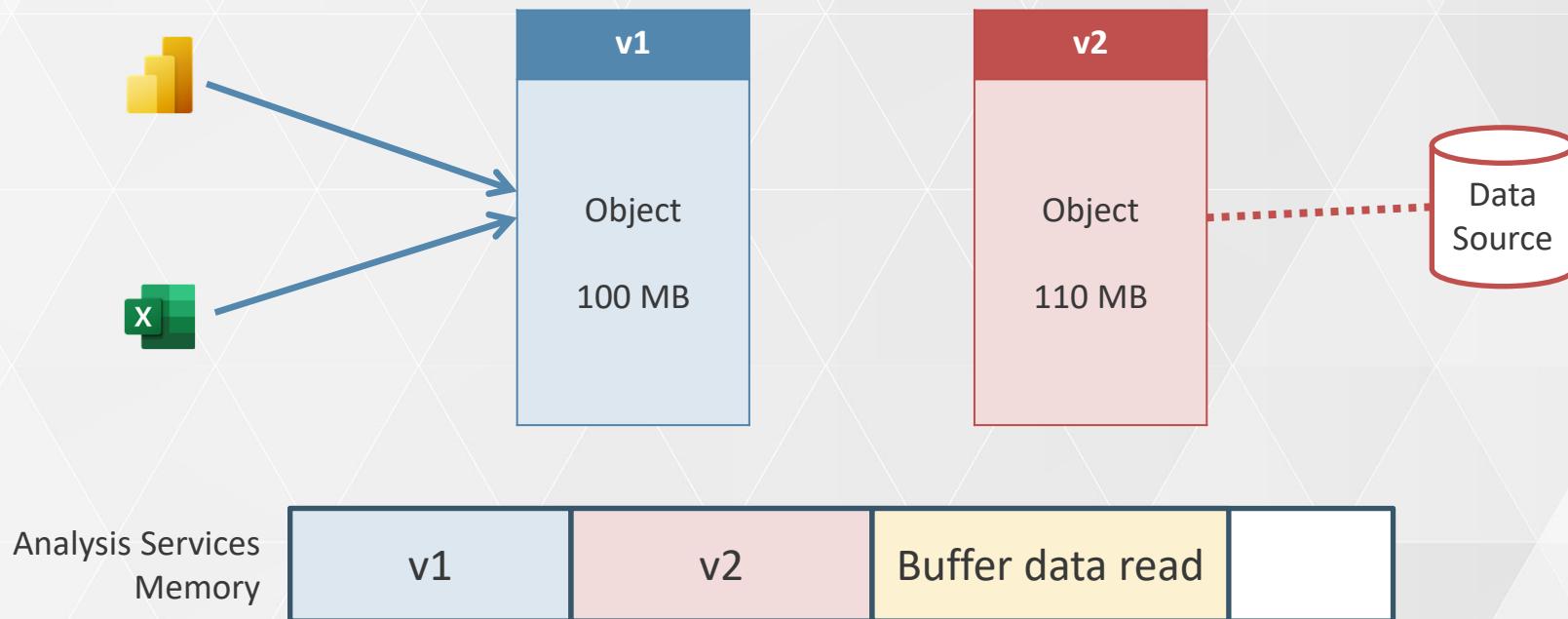
# Processing in Analysis Services

The new version of the object(s) processed requires a separate area of memory during data refresh.



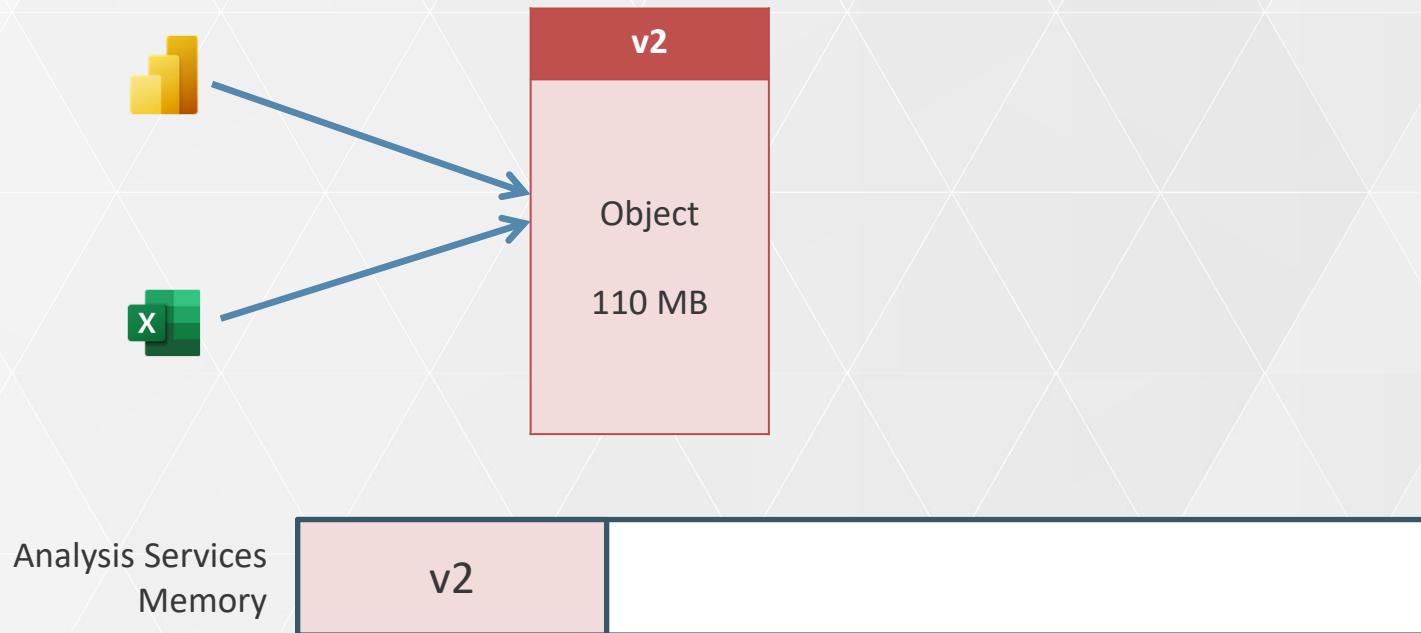
# Processing in Analysis Services

The new version of the object(s) processed requires a separate area of memory during data refresh.



# Processing in Analysis Services

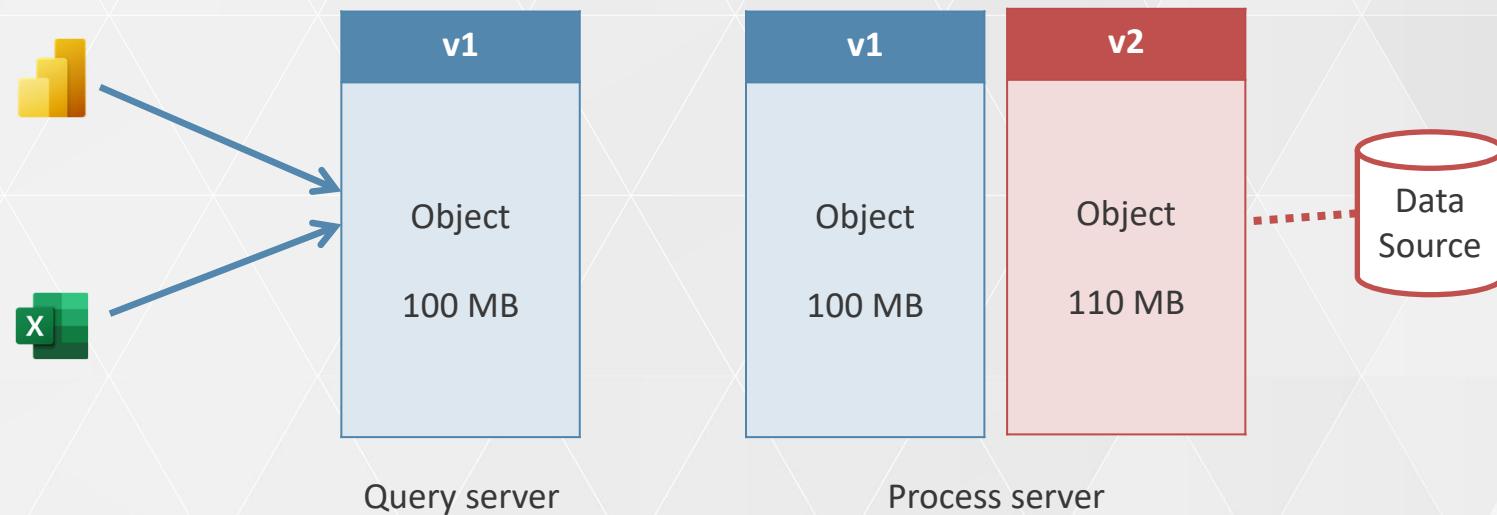
The new version of the object(s) processed requires a separate area of memory during data refresh.



# Processing in Power BI

Power BI (Gen2) uses an architecture where the process occurs on a separate server.

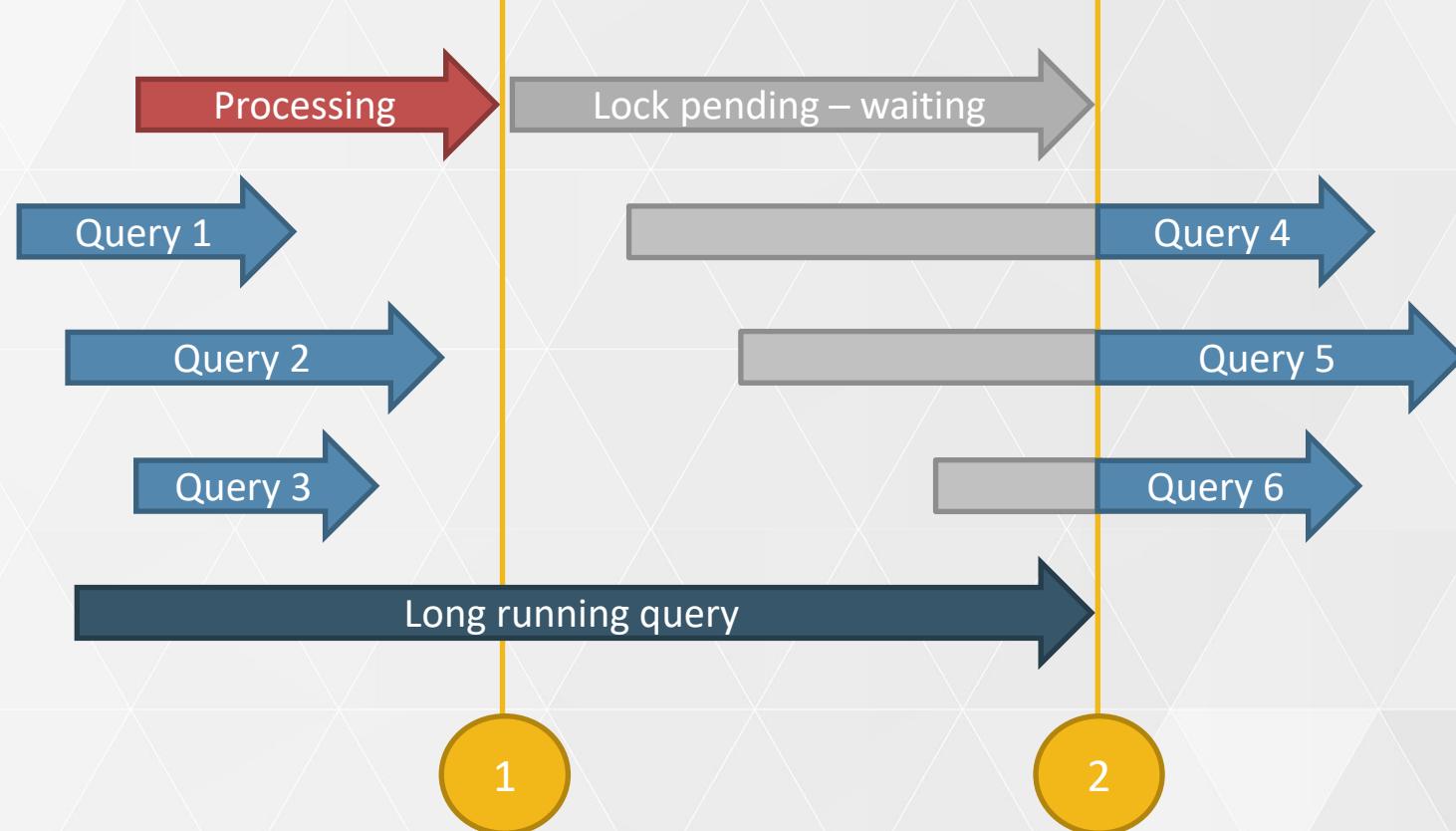
Memory pressure during processing only affects process servers, without impacting query performance.



# Real-time and near real-time updates

- Real-time:
  - DirectQuery
  - Push Dataset
  - \* *Hybrid tables (Power BI, 2022 or later?)*
- Near real-time
  - Small processing window (seconds / minutes)
  - Standard partitioning for old data (yesterday)
  - One partition with the latest data (today)
- Very frequent processing of the latest data
  - Consider **ProcessAdd**

# Long-running queries



## Tips for Near real-time

- After process data, process recalc is needed:
  - Calculated columns and calculated tables
  - Attribute hierarchies and user hierarchies
  - Relationships
- Database storage engine cache is cleared after any process
- ProcessAdd on a single partition (or add/merge partitions)
- Reduce recalc cost by using a separate table
  - Use DAX measures to sum historical and current values
  - Not compatible with DISTINCTCOUNT

# Processing strategies

- Always choose the simplest technique
  - **Process Full** whenever possible
  - Otherwise, tune your processing strategy
- Several points of attention
  - Available processing window
  - Availability of data during processing
  - Rollback in case of errors
  - Available memory

# Automating data refresh (AS)

- Using TMSL scripts in SQL Server Analysis Services
  - Using ASCMD
  - Using SQL Server Agent
  - Using SSIS
- Azure Analysis Services
  - [REST API](#): Json request (not TMSL)
  - [Azure Logic Apps](#): internal calls to REST API
  - [Azure Automation](#) (PowerShell)

# Automating data refresh (Power BI)

- XMLA endpoint in Power BI Premium
  - TMSL
  - PowerShell / .NET Scripts
  - Control partition processing
- Power BI REST API
  - Full dataset refresh (including incremental refresh policy)
  - No control over partitions

What you need to know to keep a check on your Tabular instance

# Monitoring Tabular



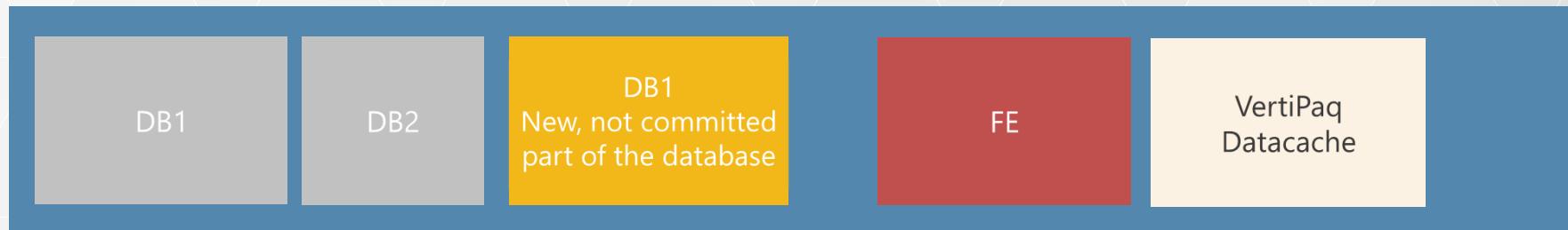
# Monitoring Tabular

- Memory configuration
- Monitor process operations
- Monitor query requests
- Differences between on-premises and cloud services

# Memory use of Analysis Services

- Processing Time
  - Table segments loading
  - Create dictionaries
  - Compress segments
  - Keep previous versions of items processed
- Querying Time
  - Cached results
  - Temporary results spooled
  - Might consume memory for materialization

# Analysis Services memory map



VertiPaqMemoryLimit

LowMemoryLimit

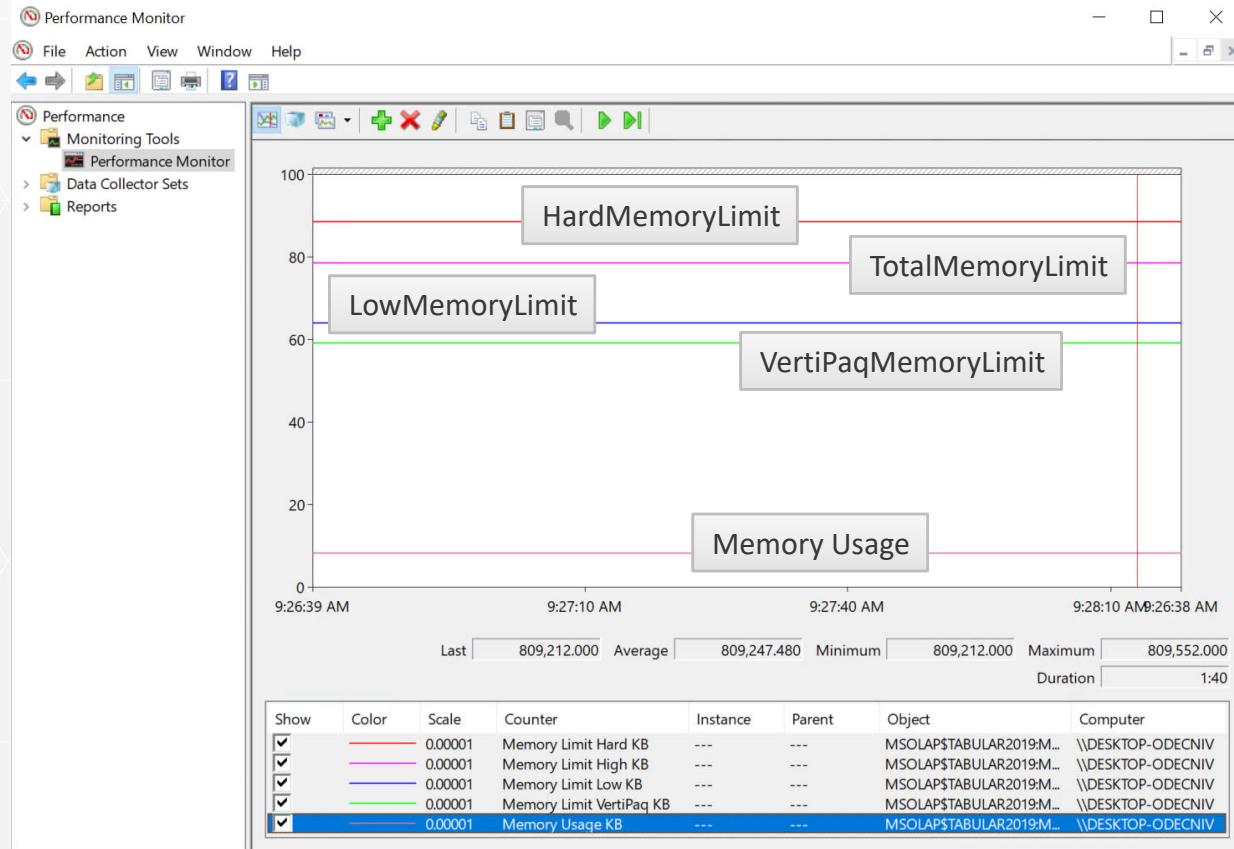
TotalMemoryLimit

HardMemoryLimit

## Memory performance counters

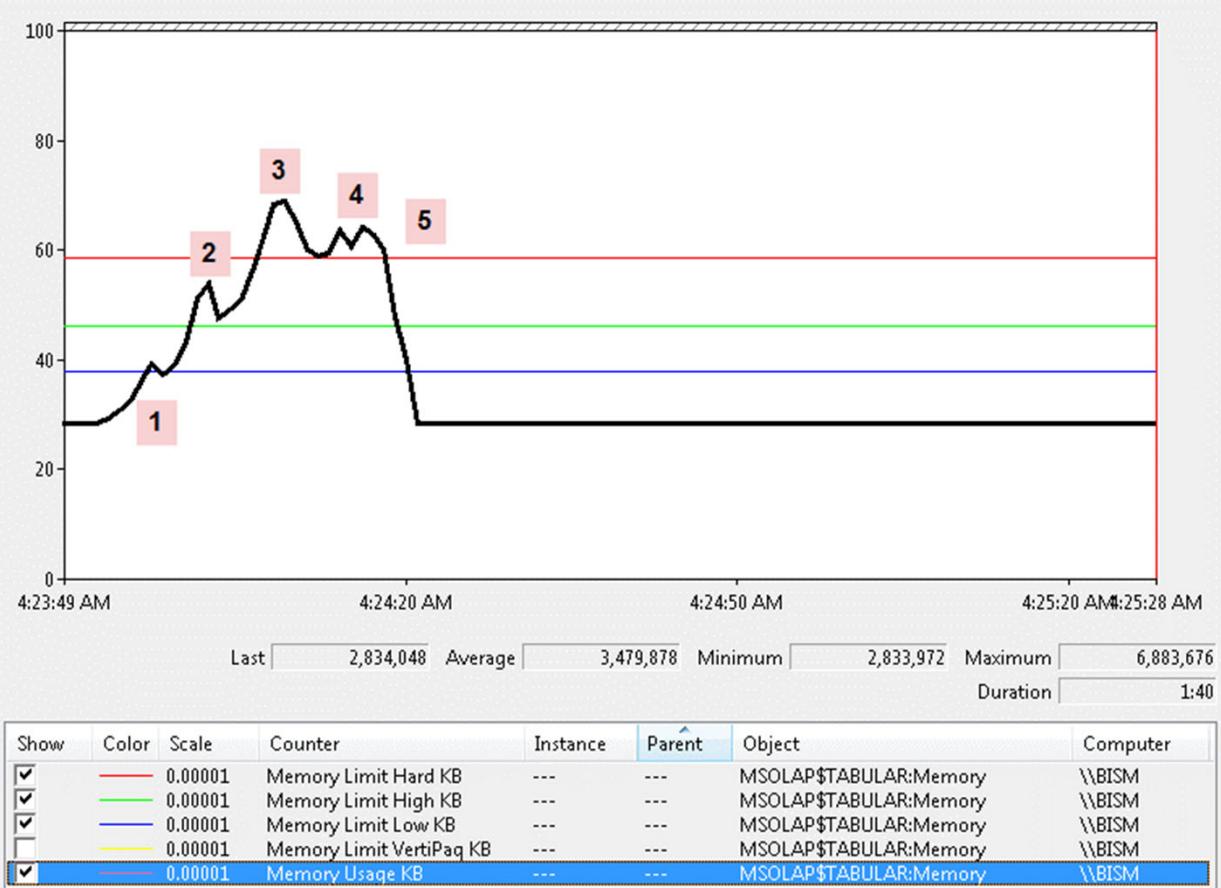
- Under the instance name: Memory (*configuration setting*)
  - Memory Limit VertiPaq – *VertiPaqMemoryLimit*
  - Memory Limit Low – *LowMemoryLimit*
  - Memory Limit High – *TotalMemoryLimit*
  - Memory Limit Hard – *HardMemoryLimit*
  - Memory Usage KB
- Compare memory usage with memory limits

# Performance Monitor



# Expensive request going out of memory

1. LowMemoryLimit passed, the cleaner starts to free up memory.
2. TotalMemoryLimit passed, the cleaner is very aggressive.
3. The cleaner tries to free up memory to complete the request.
4. The cleaner is unable to free memory.
5. Memory allocation fails above HardMemoryLimit, the connection is closed.



# Analysis Services memory configuration

Analysis Server Properties

Select a page: Information, General, Language/Collation, Security.

Script ▾ Help

Name	Value	Current Value	Default Value	Restart	Type	Units	Category
MaxIdleSessionTimeout	0	0	0	yes	int	Sec	Advanced
MDataEngine \ Processing \ MaxContainerCount	-1	-1	-1	yes	int		Advanced
MDataEngine \ Processing \ MinContainerCount	-1	-1	-1	yes	int		Advanced
Memory \ HardMemoryLimit	0	0	0	yes	double		Basic
Memory \ HeapTypeForObjects	-1	-1	-1	yes	int		Advanced
Memory \ LowMemoryLimit	65	65	65	yes	double		Basic
Memory \ MemoryHeapType	-1	-1	-1	yes	int		Advanced
Memory \ QueryMemoryLimit	0	0	0	yes	double		Advanced
Memory \ TotalMemoryLimit	80	80	80	yes	double		Basic
Memory \ VertiPaqMemoryLimit	60	60	60	yes	double		Basic
Memory \ VertiPaqPagingPolicy	1	1	1	yes	int		Advanced
MinIdleSessionTimeout	2700	2700	2700	yes	int	Sec	Advanced
Network \ Listener \ IPV4Support	2	2	2	yes	int		Advanced
Network \ Listener \ IPV6Support	2	2	2	yes	int		Advanced
Network \ Listener \ MaxAllowedRequestSize	0	0	0	yes	int	Bytes	Advanced
Network \ ListenOnlyOnLocalConnections	false	false	false	yes	bool		Basic
Network \ ListenOnTCPConnections	1	1	1	yes	int		Basic
Network \ Requests \ EnableBinaryXML	false	false	false	yes	bool		Basic
Network \ Requests \ EnableCompression	false	false	false	yes	bool		Basic
Network \ Responses \ EnableBinaryXML	true	true	true	yes	bool		Basic
Network \ Responses \ EnableCompression	true	true	true	yes	bool		Basic
OLAP \ LazyProcessing \ Enabled	true	true	true	yes	bool		Basic
OLAP \ LazyProcessing \ MaxCPUUsage	0.5	0.5	0.5	yes	double		Advanced
OLAP \ LazyProcessing \ MaxObjectsInParallel	2	2	2	yes	int		Advanced
OLAP \ LazyProcessing \ SleepIntervalSecs	5	5	5	yes	int	Sec	Basic
OLAP \ Process \ AggregationMemoryLimitMax	80	80	80	yes	double		Advanced

Show Advanced (All) Properties

Save Only Modified Properties

Reset default

OK Cancel

## Memory settings

- Settings that can be modified only in SSAS on-premises:
  - TotalMemoryLimit
  - LowMemoryLimit
  - VertiPaqMemoryLimit
- Decrease LowMemoryLimit on shared servers
- Increase TotalMemoryLimit on dedicated servers

## VertiPaq settings

- **VertiPaqPagingPolicy**
  - **0** - Paging not allowed – default on Azure Analysis Services
  - **1** - Paging allowed – default on SQL Server Analysis Services
- **VertiPaqMemoryLimit:** Physical RAM available to VertiPaq

# VertiPaqPagingPolicy for Azure Analysis Services

- Azure AS sets **VertiPaqPagingPolicy** to 0
  - Setting applied by template creating AAS instance
  - It can be changed by using SSMS
  - In **Mode 0**, it can prevent database full refresh for lack of available memory.
  - Switch to **Mode 1** to complete database refresh without having to scale up Azure AS tier.
- Optimizing memory settings in Analysis Services:  
<https://www.sqlbi.com/articles/optimizing-memory-settings-in-analysis-services/>

## Memory settings for Power BI service

- Every dataset is a separate AS database
- A workspace seems like an Analysis Services instance
  - The correspondence is “virtual”
  - Every database could be in a separate instance
  - Memory limits are by dataset/database in Power BI
- No direct access to Analysis Services memory settings

# Data Management Views (DMV)

- DMV to monitor the current state of Analysis Services
  - DISCOVER\_COMMANDS
  - DISCOVER\_CONNECTIONS
  - DISCOVER\_SESSIONS
  - DISCOVER\_TRANSACTIONS
- SSAS Activity Monitor (free open-source tool):  
<https://www.sqlbi.com/tools/ssas-activity-monitor/>

## Trace events and extended events

- Analysis Services generates diagnostic events
- Events exposed through:
  - SQL Server Profiler events
    - Used by interactive tools (SQL Server Profiler, DAX Studio)
    - Computationally expensive – don't use in production
  - Extended events
    - Lightweight infrastructure
    - No generic client tools
    - Used by cloud services and on production servers

# Event types

- Command
- Discover
- Discover Server State
- Errors and Warnings
- File Load and Save
- **Job Graph**
- M Data Provider
- Notification
- **Progress Reports**
- **Queries**
- **Query Processing**
- Resource Governance
- Security Audit
- Session Events

Analysis Services Trace Events

<https://docs.microsoft.com/en-us/analysis-services/trace-events/analysis-services-trace-events>

# Events to monitor process operations

- Job Graph
  - Visualizing Azure Analysis Services processing tasks with the Job Graph events sample  
<https://blog.crossjoin.co.uk/2020/11/15/visualising-azure-analysis-services-processing-tasks-with-the-job-graph-events-sample/>
  - Analysis-Services/ASJobGraphEvents  
<https://github.com/microsoft/Analysis-Services/tree/master/ASJobGraphEvents>
- Progress Reports
  - Multiple subclass events
  - Begin/End in interactive analysis
  - Collect only End events in production

## Events to monitor query requests

- Queries
  - Requests received from clients (DAX/MDX/DMV)
  - QueryEnd event includes total execution time
  - Collect End events in production
- Query Processing
  - Internal events executing a query
  - Includes storage engine requests and cache hit
  - Collect interactive events – DAX Studio

## Interactive analysis

- DAX Studio
  - Analyze details of single query execution (Query Processing)
  - Possible Job Graph visualization in future versions
- SQL Server Profiler
  - Not recommended for SQL Server, supported for Analysis Services
  - Analyze any trace event
  - Complex correlation between events
  - Export captured events

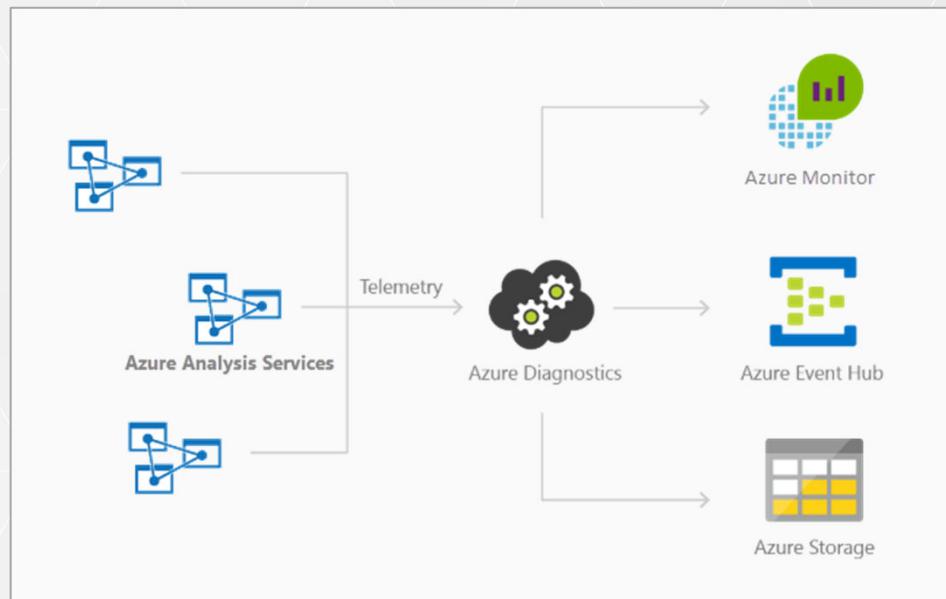
## SQL Server Analysis Services on-premises

- Collect extended events
- Automate collection of QueryEnd events to collect slower queries
- Automate collection of Progress Report End events to analyze scheduled data refresh
- SSAS Events Analyzer (free open-source):  
<https://marcosqlbi.github.io/SsasEventsAnalyzer/>

# Azure Analysis Services

- Extended events available in Diagnostic Logs of Azure Analysis Services:

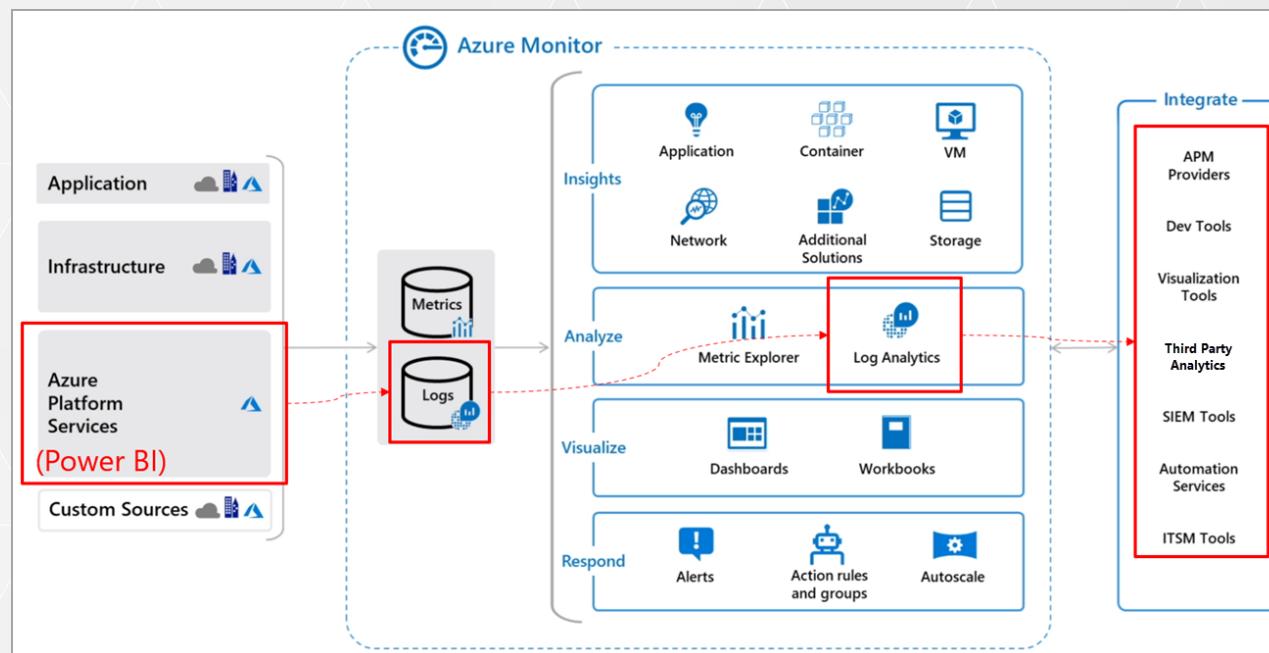
<https://docs.microsoft.com/en-us/azure/analysis-services/analysis-services-logging>



# Power BI Premium

- Integration with Azure Log Analytics:

<https://docs.microsoft.com/en-us/power-bi/transform-model/log-analytics/desktop-log-analytics-overview>



Moving data between servers

# Backup and Restore strategies



## Do you need backups?

- You should preserve the structure (model.bim), not the data
- A full backup of the database includes structure & data
  - Restore time is faster than full process
  - Useful to move a database to a different server
  - Backup not needed if full process time is acceptable
- Two techniques corresponding to XMLA or [TMSL](#) commands:
  - Backup/Restore
  - Attach/Detach

# Database structure

Analysis Server Properties

Select a page: Information, General, Language/Collation, Security.

Script, Help.

Name	Value	Current
BackupDir	C:\Program Files\Microsoft SQL Server\MSAS15.TABULAR2019\OLAP\Backup	C:\Program
CommitTimeout	0	0
CoordinatorExecutionMode	-4	-4
DataDir	C:\Program Files\Microsoft SQL Server\MSAS15.TABULAR2019\OLAP\Data	C:\Program
DataMining \ AllowAdHocOpen...	false	false
DataMining \ AllowSessionMini...	false	false
DataMining \ MaxConcurrentDr...	0	0

File | Contoso.0.db

File Home Share View

Local Disk (C:) > Program Files > Microsoft SQL Server > MSAS15.TABULAR2019 > OLAP > Data > Contoso.0.db >

MSAS15.TABULAR2019

OLAP

Backup

bin

Config

Data

CleanupDir

Contoso.0.db

ExcelMDX.0.asm

EXCELXLINTERNAL.0.asm

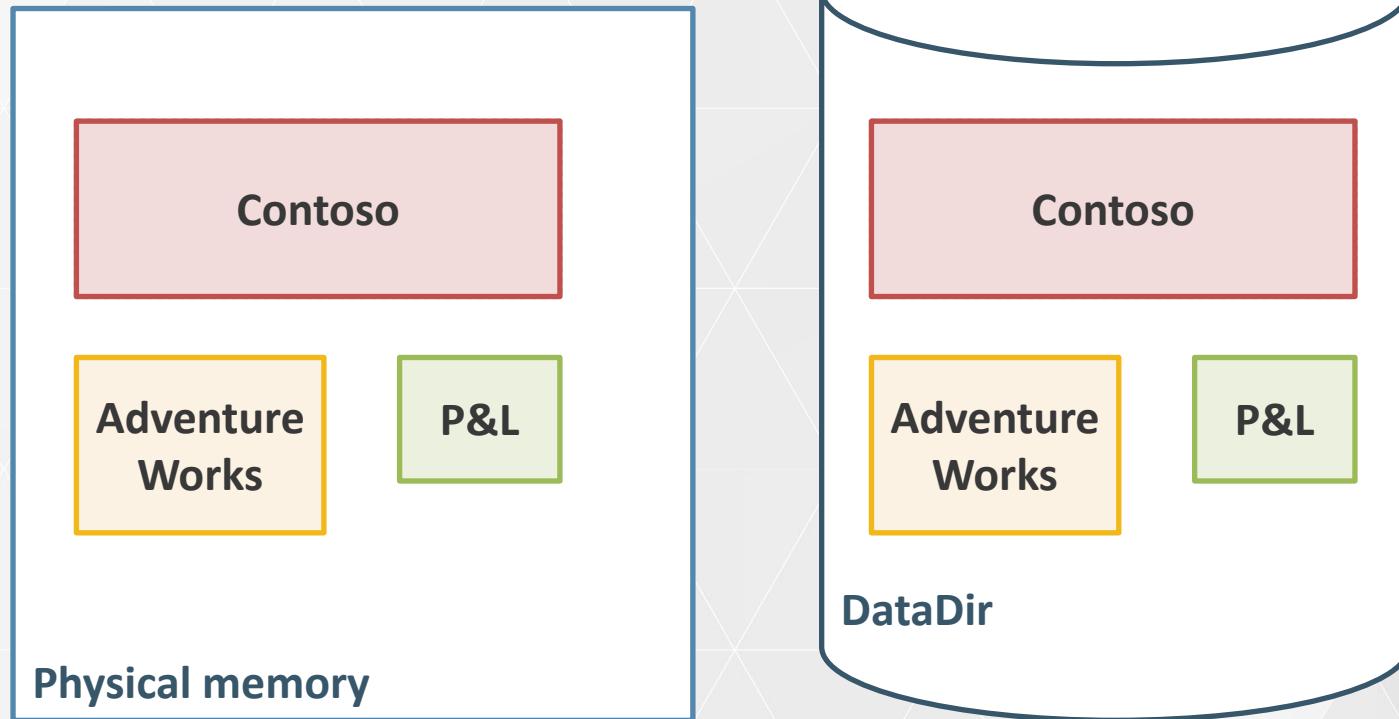
Name Date modified Type

Name	Date modified	Type
Customer (10).tbl	7/9/2021 10:25 AM	File folder
Date (19).tbl	7/9/2021 10:25 AM	File folder
H\$Customer (10)\$Address (40)\$(3938).tbl	7/9/2021 10:25 AM	File folder
H\$Customer (10)\$Age (48)\$(4074).tbl	7/9/2021 10:25 AM	File folder
H\$Customer (10)\$Birthday (47)\$(4057).tbl	7/9/2021 10:25 AM	File folder
H\$Customer (10)\$City (41)\$(3955).tbl	7/9/2021 10:25 AM	File folder
H\$Customer (10)\$Country (46)\$(4040).tbl	7/9/2021 10:25 AM	File folder
H\$Customer (10)\$Country Code (45)\$(4023).tbl	7/9/2021 10:25 AM	File folder
H\$Customer (10)\$CustomerKey (37)\$(3887).tbl	7/9/2021 10:25 AM	File folder
H\$Customer (10)\$Gender (38)\$(3904).tbl	7/9/2021 10:25 AM	File folder

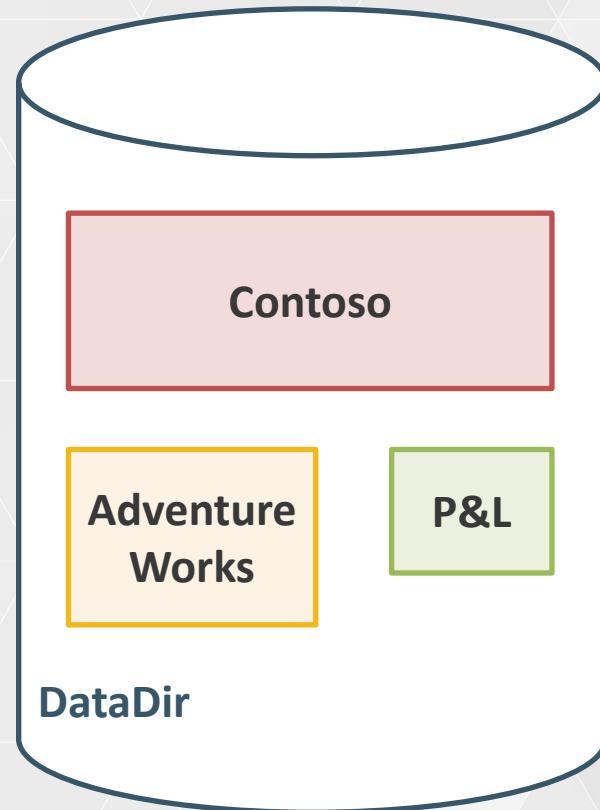
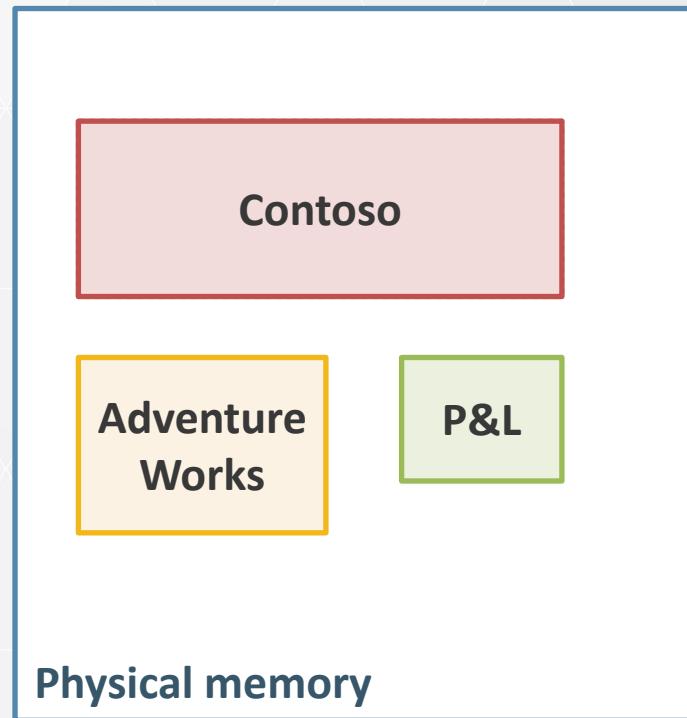
78 items



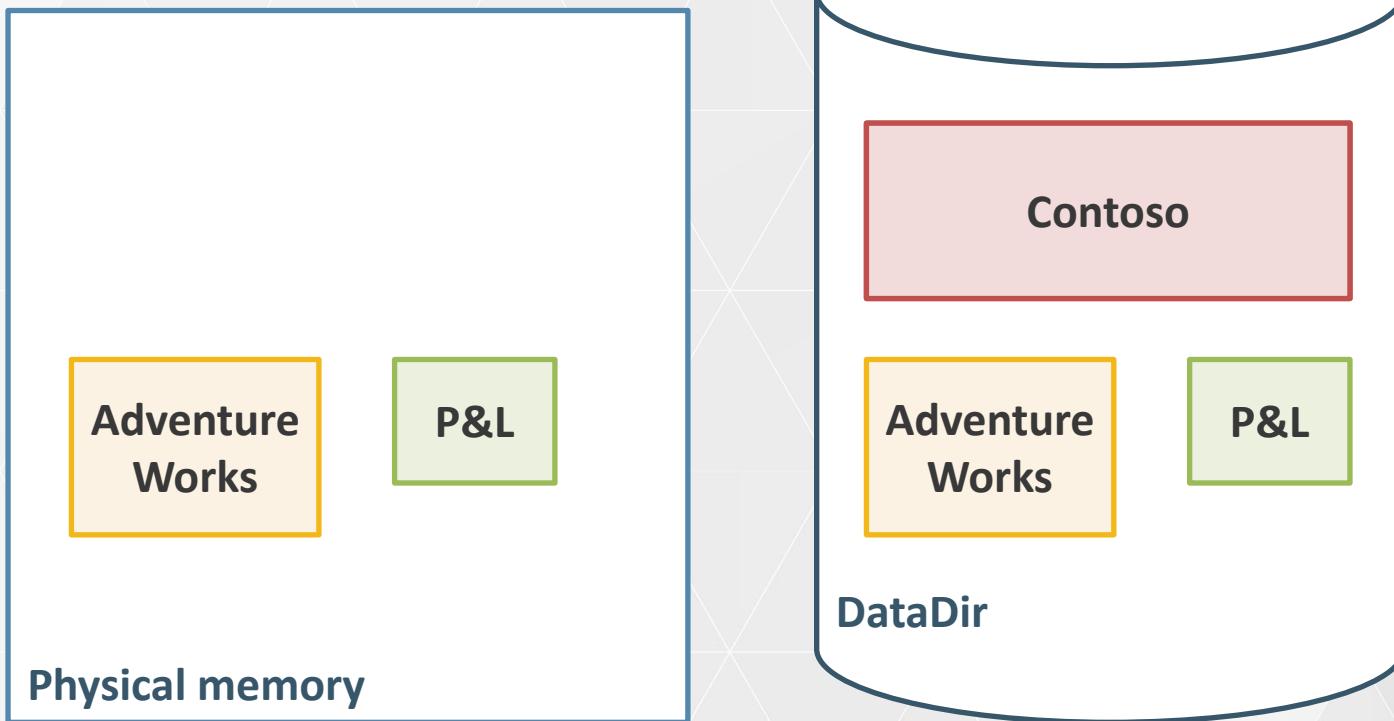
# Delete database



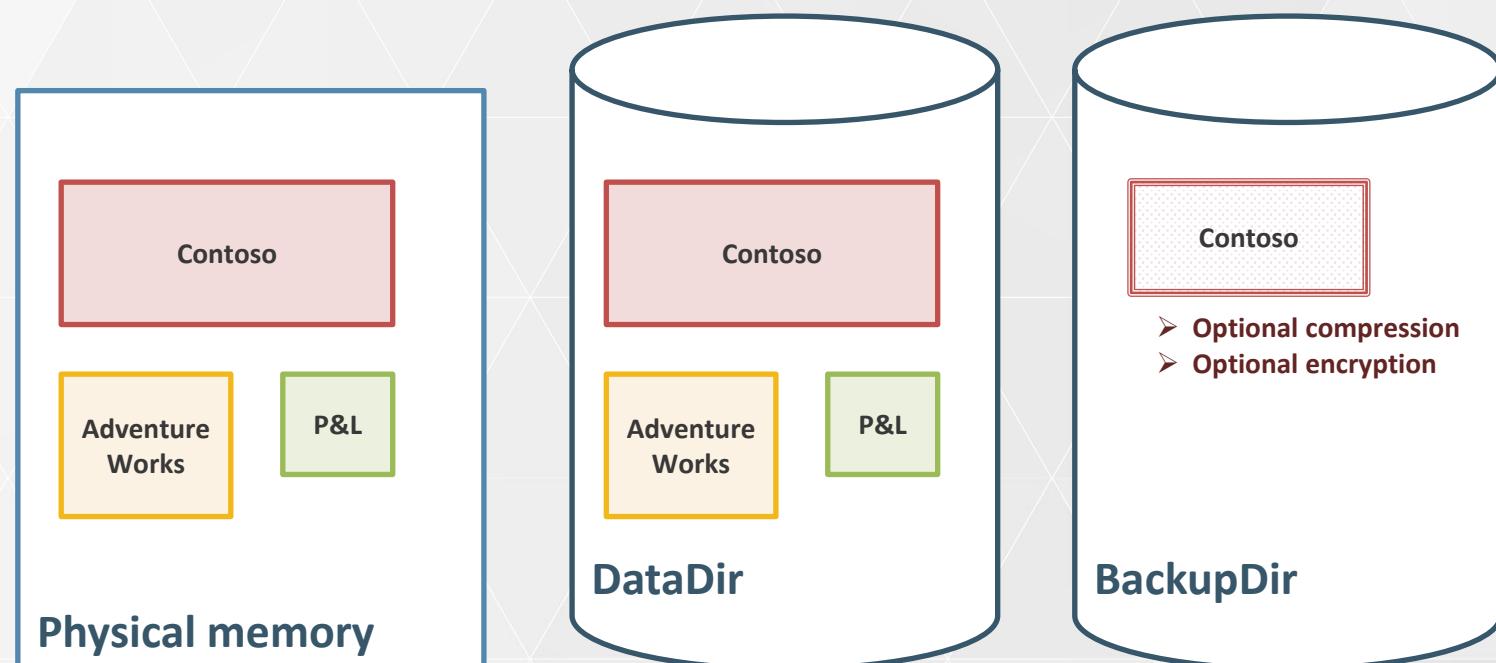
# Detach database



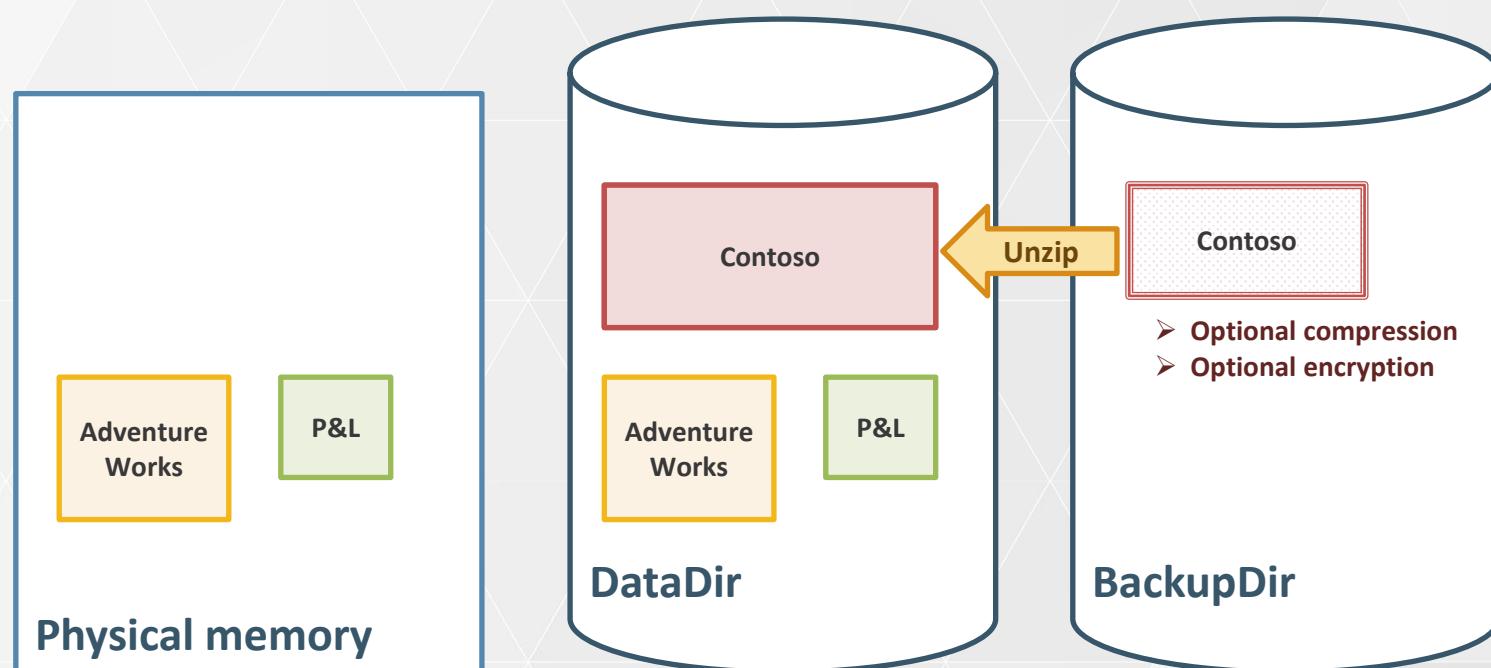
# Attach database



# Backup database



# Restore database



## Attach and detach database – SSAS on-premises

- Detach: removes the database from memory
  - The objects are still available in the data folder
  - The database is stored in a corresponding folder
  - Internal proprietary formats (XML, JSON, binary)
- Attach: loads database into memory
  - Loads files from data folder in memory
  - SSMS show folders only in *AllowedBrowsingFolders*
- Not supported by Azure AS and Power BI

## Backup and restore on-premises (SSAS)

- Backup made to *BackupDir*
- Restore possible from *AllowedBrowsingFolders* (SSMS UI)
- Restore may or may not include members in security roles

# Backup/restore on Azure Analysis Services

- Define storage in Backup Storage Settings  
<https://docs.microsoft.com/en-us/azure/analysis-services/analysis-services-backup>
- Can move on-premises backups to Azure
  - Copy the backup file using Azure Storage Explorer
  - The opposite is possible, but compatibility level could be incompatible on premises

## Backup/restore on Power BI Premium

- Configure ADLS Gen2 storage account for Premium Capacity to store backup files  
<https://docs.microsoft.com/en-us/power-bi/admin/service-premium-backup-restore-dataset>
- Can move on-premises and Azure AS backups to Power BI
  - Copy the backup file using Azure Storage Explorer
  - Database restored as a Large Model (Premium) database
  - Password encryption in backup is not supported

Increase the number of concurrent users

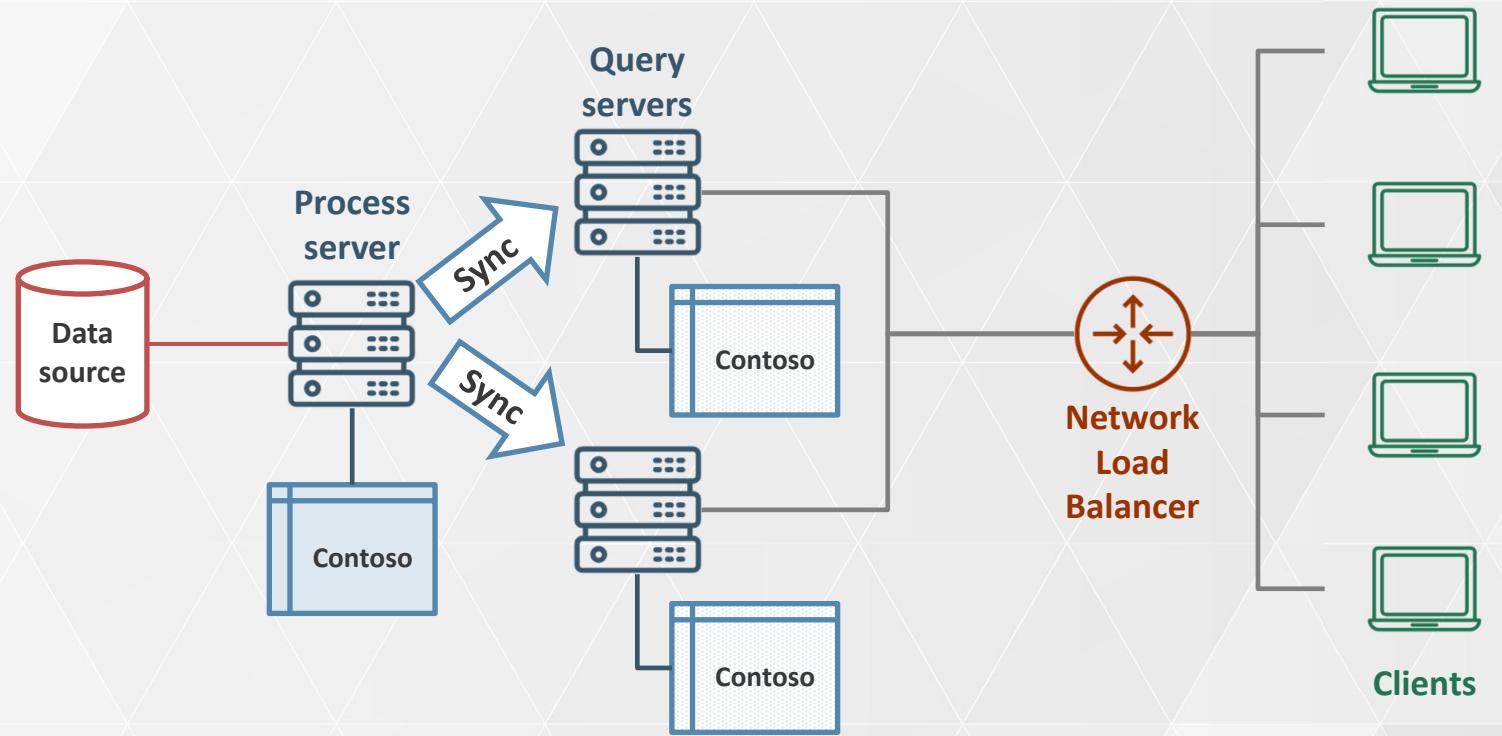
## Scaling out



# Scaling out vs. scaling up

- Multiple users querying at the same time
  - CPU consumption
  - RAM can be the bottleneck
- Scale up
  - More cores, more RAM, more NUMA nodes
  - Physical limits of a single server
- Scale out
  - Distribute the query on multiple query servers
  - Use a separate server as process server

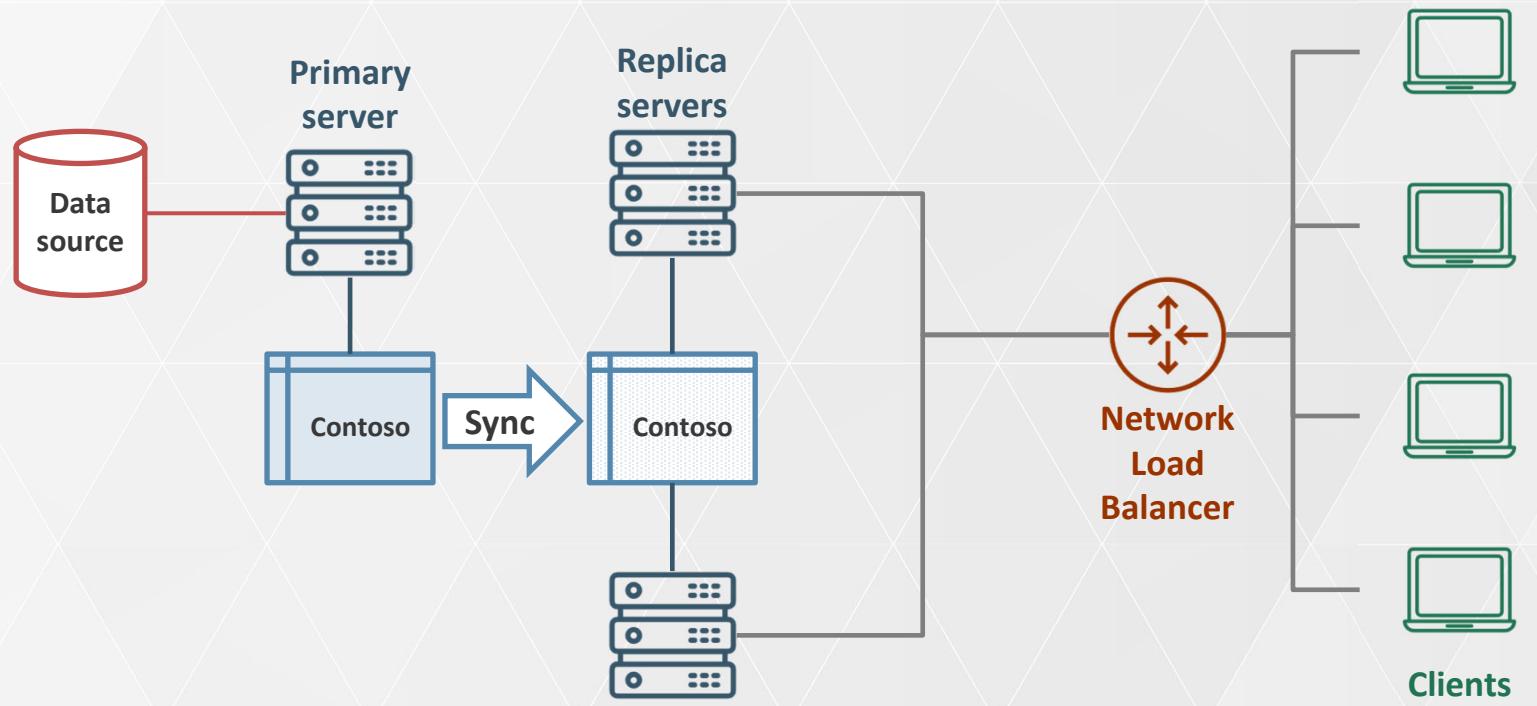
# On-premises Analysis Services scale-out architecture



# Scaling out with on-premises Analysis Services

- Manually manage multiple SSAS servers
  - Process model on a process server
  - Deploy model on query server
  - Two options to manage client connections:
    - Statically distribute client connection to query servers
    - Implement and maintain a network load balancer
- Manage database synchronization (on-premises SSAS only)
  - Synchronize Database Wizard (in SSMS)
  - XMLA or [TMSL](#) Synchronize command

# Azure Analysis Services scale-out architecture



# Scaling out with Azure Analysis Services

- Query pool
  - One primary server (process/query)
  - Up to 7 query replica resources
- Synchronization mode (within the same region)
  - 1 (default): Full replica database rehydration in stages
    - Detach/attach databases on query replicas making sure at least one is always available (with old data)
  - 2: Optimized synchronization in parallel
    - Lower latency, but requires more memory
- <https://docs.microsoft.com/en-us/azure/analysis-services/analysis-services-scale-out>

## Scaling out with Power BI service

- Power BI services does not scale out instances
- Automatic **scale-up** for cores available is possible
- Capacity is limited to a single database instance with the highest number of cores available
  - P3: 16 vCores for backend
  - P5: 64 vCores for backend
  - Auto-scale: additional v-cores for 24 hours
- <https://docs.microsoft.com/en-us/power-bi/admin/service-premium-auto-scale>