**Data Modeling**

# **What is data modeling?**

Data modeling is the process of analyzing and defining all the different data your business collects and produces, as well as the relationships between those bits of data.

Modeling data is about establishing and maintaining relationships so that you can effectively visualize the data in the form that your business requires.

## **Why data modeling is important?**

By modeling your data, you’ll document what data you have, how you use it, and what your requirements are .

* Creates a structure for collaboration between your IT team and your business teams.
* Reduces errors (and error-prone redundant data entry), while improving data integrity.
* Increases the speed and performance of data retrieval and analytics by planning for capacity and growth

**What is Data Model?**

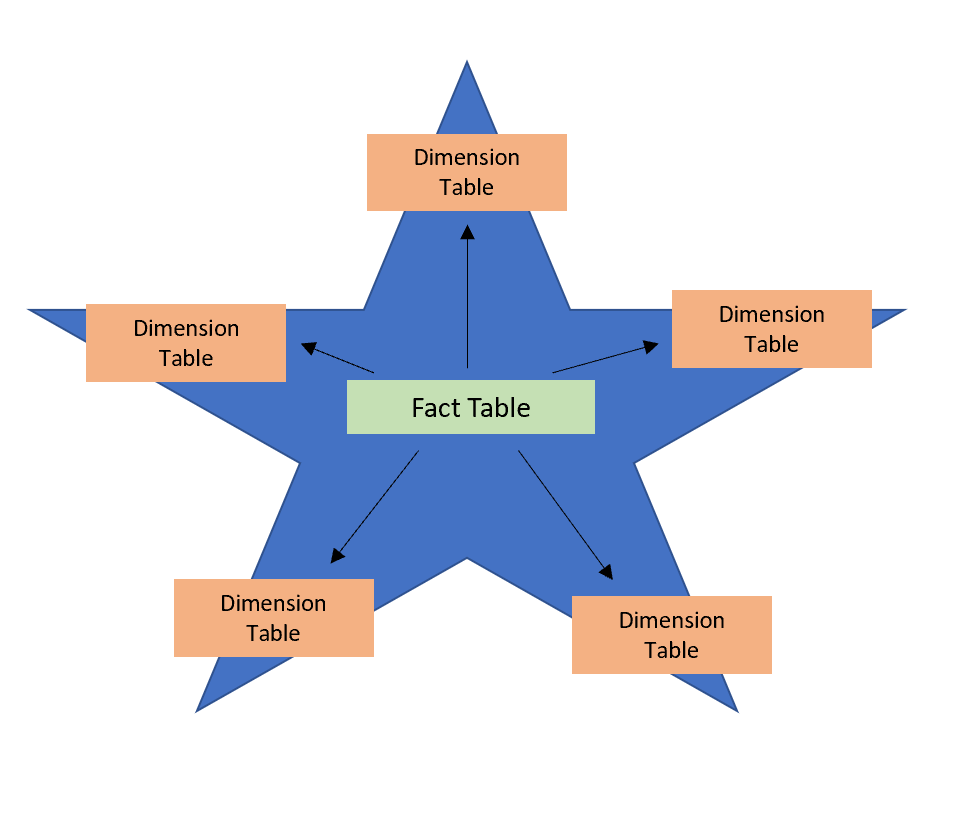
The term “data model” refers to **the way data is organized, documented, and defined within a database**.

A good data model offers the following benefits:

* Data Exploration is faster
* Aggregation are simpler to build.
* Reports are more accurate.
* Writing reports takes less time.

## **Star schemas**

You can design a star schema to simplify your data. It's not the only way to simplify your data, but it is a popular method.



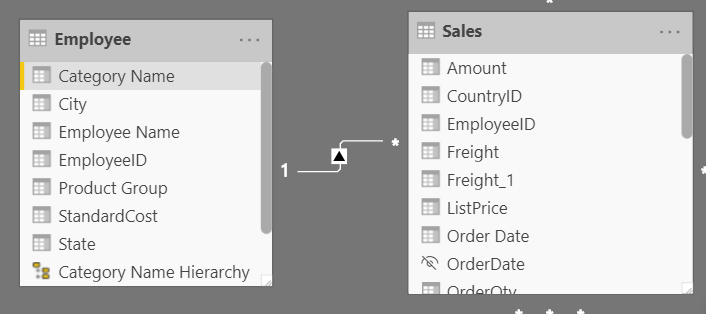
****Fact tables**** :

* contain observational or event data values.
* can contain several repeated values
* Ex. sales orders, product counts, prices

****Dimension tables**** :

* contain the details about the data in fact tables.
* connected to the fact table through key columns.
* Generally contain unique values.

Ex. products, locations, employees, and order types.



* Because the Sales table contains the sales order values, which can be aggregated, it is considered a fact table.
* The Employee table contains the specific employee name, which filters the sales orders, so it would be a dimension table.
* establish a relationship by **EmployeeID.**

# **Work with tables**

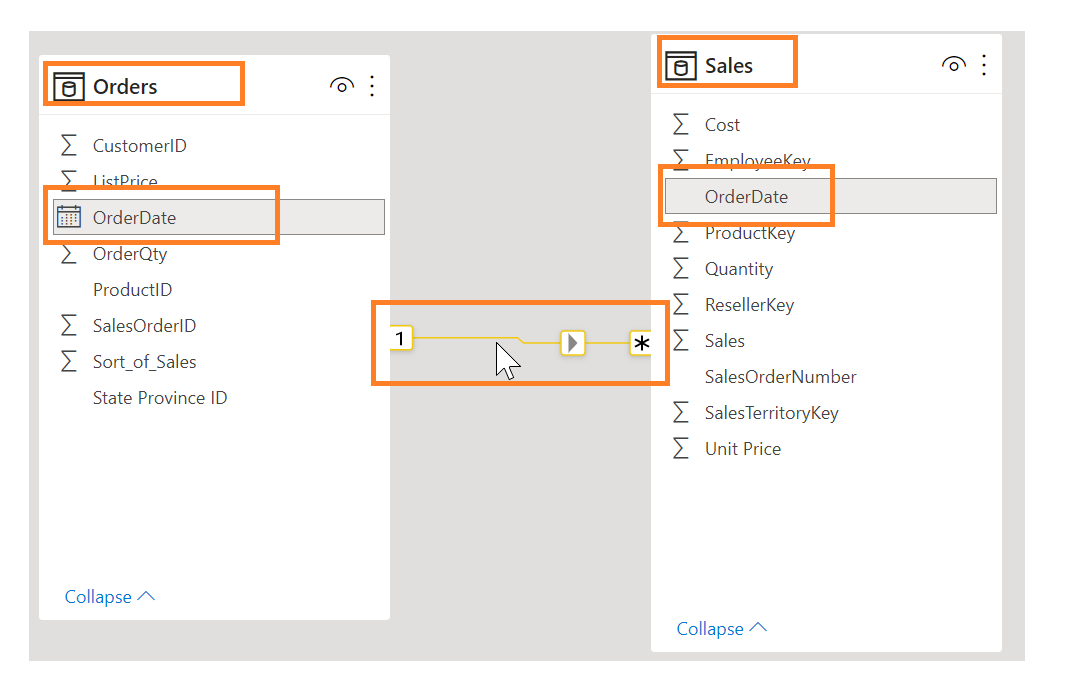
A good table structure will:

* Be simple to navigate
* Have merged or appended tables to simplify the tables within your data structure.
* Have good-quality relationships between tables that make sense.

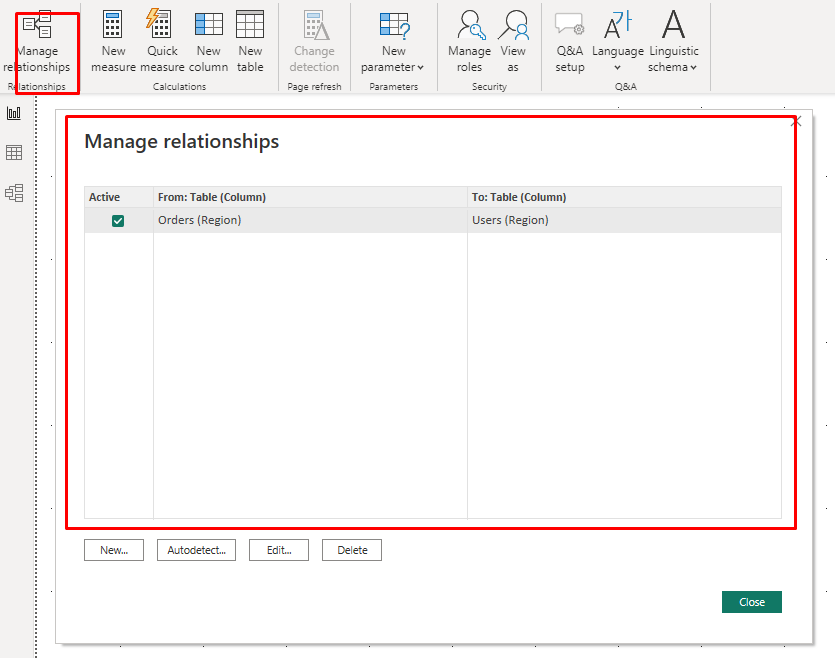
## **Configure data model and build relationships between tables**

**Steps :**

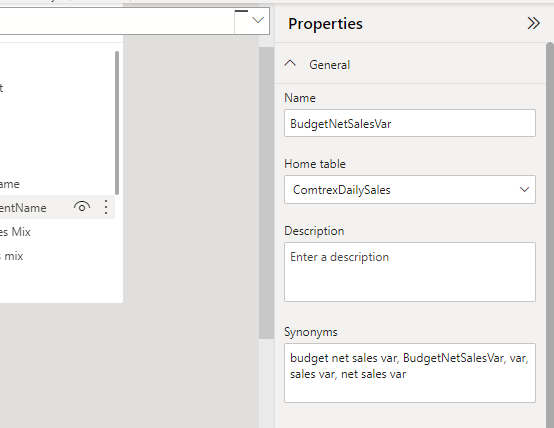
* **Retrieve your data**
* **Clean it in Power Query**
* **Go to the model tab**

****

To manage these relationships, go to ****Manage Relationships**** on the ribbon,

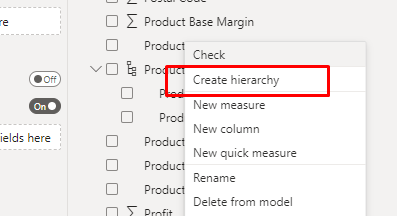


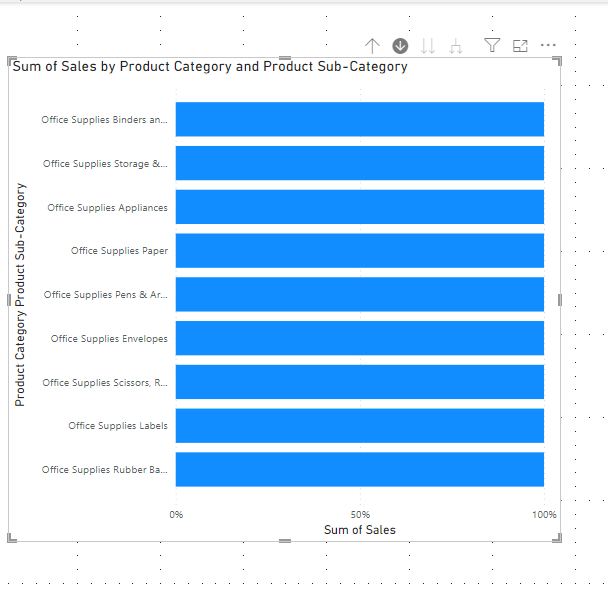
**Configure table and column properties :**



# **Work with dimensions**

# **Hierarchies**



**Relationships**

**Many-to-one (\*:1) / one-to-many (1: \*) relationship**

Describes a relationship in which you have many instances of a value in one column that are related to only one unique corresponding instance in another column.

**One-to-one (1:1) relationship:**

Describes a relationship in which only one instance of a value is common between two tables.

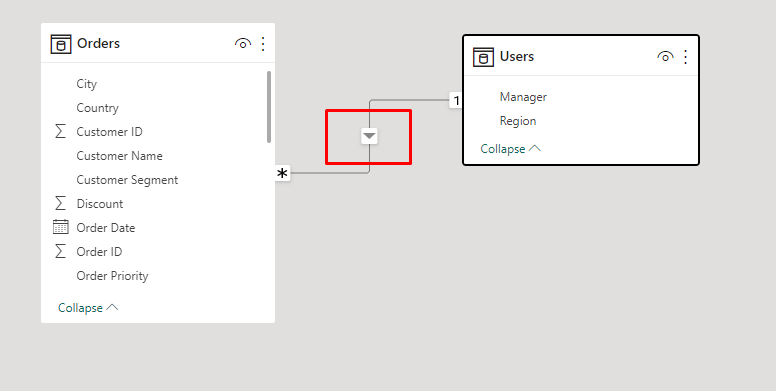
**Many-to-many (.) relationship:**

Describes a relationship where many values are in common between two tables.

## **Cross-filter direction**

Data can be filtered on one or both sides of a relationship.

Follow the direction of the arrow on the relationship between your tables to know which direction the filter will flow. You typically want these arrows to point to your fact table.



# **Introduction to DAX**

Data Analysis Expressions (DAX) is a programming language that is used throughout Microsoft Power BI for creating calculated columns, measures, and custom tables.

**Use calculated columns**

DAX allows you to create a calculated column that didn't originally exist in the data source.

**Use measures**

Calculated columns are useful, when you are required to operate row by row

## **Differences between a calculated column and a measure**

* calculated column creates a value for each row in a table. For example, if the table has 1,000 rows, it will have 1,000 values in the calculated column.
* Calculated column values are stored in the Power BI .pbix file. Each calculated column will increase the space
* Measures are calculated on demand.
* Power BI calculates the correct value when the user requests it.
* Measures do not add to the overall disk space of the Power BI .pbix file.

# **Use the Calculate function**

The CALCULATE function is your method of creating a DAX measure that will override certain portions of the context that are being used to express the correct result.

Total Sales for 2015 = CALCULATE(SUM('Sales OrderDetails'[Total Price]), YEAR('Sales OrderDetails'[orderdate]) = 2015)

# **Use relationships effectively**

Another DAX function that allows you to override the default behavior is USERELATIONSHIP.

This function is used to specify a relationship to be used in a specific calculation and is done without overriding any existing relationships.

Sales by Ship Date = CALCULATE(Sum(Sales[TotalPrice]), USERELATIONSHIP(Sales[ShipDate],'Calendar'[Date]))

# **Work with time intelligence**

YTD Total Sales = TOTALYTD ( SUM('Sales OrderDetails'[Total Price]) , Dates[Date] )

Total Sales Previous Month = CALCULATE ( sum('Sales OrderDetails'[Total Price]) , PREVIOUSMONTH(Dates[Date]) )

# **Optimize a model for performance in Power BI**

Performance optimization, also known as performance tuning, involves making changes to the current state of the data model so that it runs more efficiently.

Ensuring that the correct data types are used.

Deleting unnecessary columns and rows.

Avoiding repeated values.

Reducing cardinalities.

Analyzing model metadata.

Summarizing data where possible.

## **Identify report performance bottlenecks**

To achieve optimal performance in your reports, you need to create an efficient data model that has fast running queries and measures.

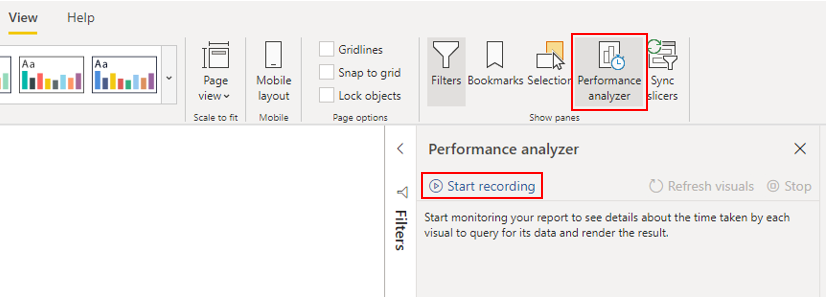
### **Analyze performance**

You can use ****Performance analyzer**** in Power BI Desktop to help you find out how each of your report elements are performing .

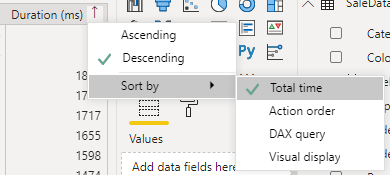
Before you run ****Performance analyzer****, Clear Cache :

****Visual cache :**** (Restart the power BI Desktop )

****Data engine cache**** :  restart Power BI Desktop or connect DAX Studio to the data model and then call Clear Cache.



### **Review results**



****DAX query**** - The time it took for the visual to send the query, along with the time it took Analysis Services to return the results.

Ex: Count Customers =CALCULATE (

DISTINCTCOUNT ( Order[ProductID] ),

FILTER ( Order, Order[OrderQty] >= 5 ))

****Visual display**** - The time it took for the visual to render on the screen, including the time required to retrieve web images .

****Other**** - The time it took the visual to prepare queries, wait for other visuals to complete, or perform other background processing tasks.

## **Resolve issues and optimize performance**

## **Visuals**

* fewer visuals means better performance.
* Rather than using multiple visuals on the page, consider other ways to provide additional details,

such as drill-through pages and report page tooltips.

* The upper limit for visuals is 100 fields (measures or columns), so a visual with more than 100 fields will be slow to load.
* Can reduce unnecessary fields from the visual

### **DAX query**

* Which measure taking so long to process.
* use DAX Studio to investigate your queries in more detail.
* Check that relationship cardinality properties are correctly configured.
* not import columns of data that you do not need

# **Use variables to improve performance and troubleshooting**

You can use variables in your DAX formulas to help you write less complex and more efficient calculations.

The use of variables in your data model provides :

* ****Improved performance****
* ****Improved readability****
* ****Simplified debugging****
* ****Reduced complexity****

Sales YoY Growth = DIVIDE ( ( [Sales] - CALCULATE ( [Sales], PARALLELPERIOD ( 'Date'[Date], -12, MONTH ) ) ), CALCULATE ( [Sales], PARALLELPERIOD ( 'Date'[Date], -12, MONTH ) ) )

Sales YoY Growth = VAR SalesPriorYear = CALCULATE ( [Sales], PARALLELPERIOD ( 'Date'[Date], -12, MONTH ) ) VAR SalesVariance = DIVIDE ( ( [Sales] - SalesPriorYear ), SalesPriorYear ) RETURN SalesVariance

## **Reduce relationship cardinality**

* always ensure that both of the columns in a relationship are sharing the same data type.

# **Optimize DirectQuery models with table level storage**

# **Implications of using DirectQuery**

* suitable in cases where data changes frequently
* can handle large data

### **Behavior of DirectQuery connections**

* When you load the data, no data is imported into the Power BI Desktop, only the schema is loaded.
* If changes are made to the underlying data, they won't be immediately reflected in the existing visuals in Power BI due to caching. You need to carry out a refresh to see those changes.

### **Limitations of DirectQuery connections**

* ****Performance****

our overall user experience depends heavily on the performance of the underlying data source.

* ****Data transformation****

Compared to imported data, data that is sourced from DirectQuery has limitations when it comes to applying data transformation techniques within Power Query Editor.

* ****Modeling****

Some of the modeling capabilities that you have with imported data aren't available, or are limited, when you use DirectQuery.

* ****Reporting****

Almost all the reporting capabilities are supported

But, when the report is published in Power BI service, the Quick Insights and Q&A features are not supported.

## **Optimize performance**

### Optimize data in Power BI Desktop using **Performance analyzer.**

* identify the queries that are taking a long time to load.

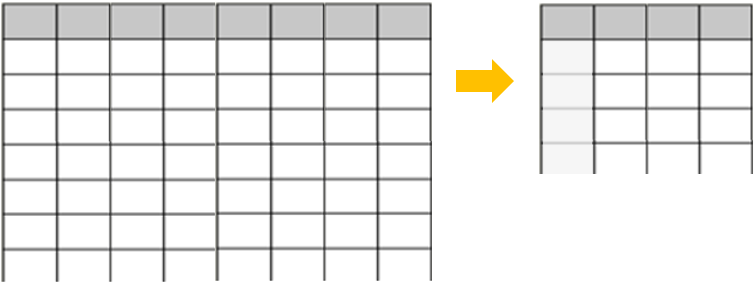
### Optimize the underlying data source

* Avoid the use of complex calculated columns because the calculation expression will be embedded into the source queries.

### Customize the Query reduction options(****File**** ****>**** ****Options and settings**** ****>**** ****Options>Query reduction****)

# **Create and manage aggregations**

The aggregation process reduces the table sizes in the data model, allowing you to focus on important data and helping to improve the query performance.



* provide better query performance and help you analyze and reveal the insights of this large data.
* help you speed up the refresh process
* can help you reduce and maintain the size of your model.

--point to discuss in practical work :

* How to get data from data source.
* Import/Direct Query
* Transform & Load

In power query Editor

* Go to view
* column distribution ,column quality , column profile
* Explain transform and add column tab

In Model view tab

* Manage Relationship (Types of Relation ship)
* Cross-filter direction
* Create a Date Table
* Active / Inactive Relationship
* table and column properties
* Hierarchies

In DAX

* Calculated column
* Calculated Measure
* Difference in column /measure
* Calculate function use
* USERELATIONSHIP. Function
* Time Intelligence function
* variables

Optimize model

* **Performance analyzer**
* Query reduction options