Intermediate Data Modeling in Power BI

Table of Content

1. Date Dimensions and Relationships

Date and time dimensions

- **Date dimensions** provide an in-built calendar and help minimize complex date operations
  - e.g. match fiscal year with calendar year
  - e.g. slice by quarter, month, week
- **Time dimensions** handle times of the day: hour, minute, second
- **Time dimensions** tend to be much less common than date dimensions

Options for creating a date dimension

Method	Advantages	Disadvantages
Host in a database	Great if you pull data from a warehouse!	Requires a database
	Easiest to share with multiple services, updating is easy	
Store data in a file	No database required, create one time	Need to create the file
	Power BI support for text files is great	Updating is not as easy as hosting in a database
Create using DAX	Allows for further customization than the prior two options	Need to write custom code
	Does not require external prep work	Some functionality may be more difficult to accomplish here

Creating a simple date dimension with DAX

```
Month_Year =


        CALENDAR(DATE(1950, 1, 1),
                 TODAY()),
```

- `CALENDAR()` is a built-in function to return all dates in a range

Creating a simple date dimension with DAX

```
Month_Year =

        CALENDAR(DATE(1950, 1, 1),
                 TODAY()),
```

- `CALENDAR()` is a built-in function to return all dates in a range
- Creates `[Date]` field with each date between 1950-01-01 and today

[Date]
1950-01-01
1950-01-02
...
2021-06-30

Creating a simple date dimension with DAX

```
Month_Year =

    SELECTCOLUMNS(
        CALENDAR(DATE(1950, 1, 1),
                 TODAY()),
        "Month", MONTH([Date]),
        "Year", YEAR([Date])
    )
```

- `CALENDAR()` is a built-in function to return all dates in a range
- Creates `[Date]` field with each date between 1950-01-01 and today
- Select the columns you want to add

Month	Year
01	1950
01	1950
...	...
06	2021

Creating a simple date dimension with DAX

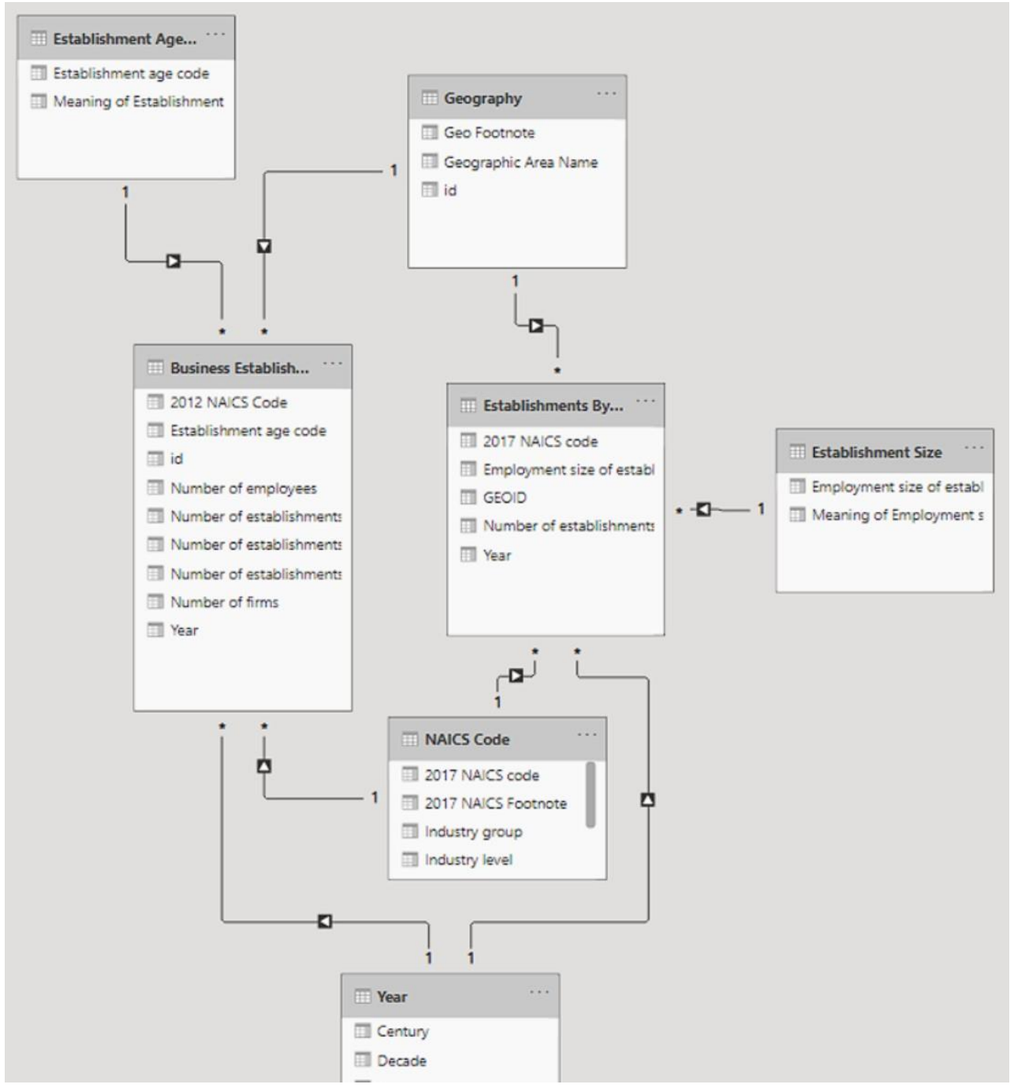
```
Month_Year =
DISTINCT(
    SELECTCOLUMNS(
        CALENDAR(DATE(1950, 1, 1),
                 TODAY()),
        "Month", MONTH([Date]),
        "Year", YEAR([Date])
    )
)
```

- `CALENDAR()` is a built-in function to return all dates in a range
- Creates `[Date]` field with each date between 1950-01-01 and today
- Select the columns you want to add

Month	Year
01	1950
02	1950
...	...
06	2021

Defining relationships

- Relationships allow you to **link tables** in Power BI
  - Propagate filters across tables
  - Allow for cross-table calculations
- Ways to manage relationships
  - Autodetect based on column names
  - Manually customization



Relationship keys

- Relationships are based on keys
  - One or more columns which guarantee a row is unique
- Two types of keys:
  - **Natural key**: existing column *(e.g. email)*
  - **Surrogate key**: artificial column *(e.g. ID)*
- Power BI requires **single column** relationships

Relationship keys

- Relationships are based on keys
  - One or more columns which guarantee a row is unique
- Two types of keys:
  - **Natural key**: existing column *(e.g. email)*
  - **Surrogate key**: artificial column *(e.g. ID)*
- Power BI requires **single column** relationships
- **Composite key**: a key made up of at least two columns

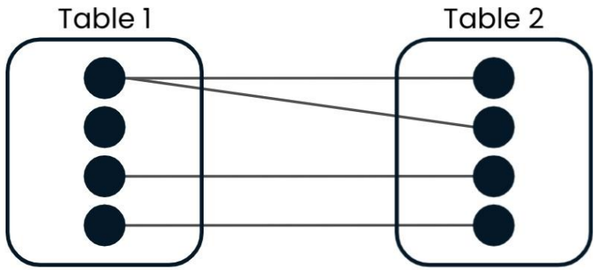
First Name	Last Name	Birth year	Value
Chris P	*Bacon*	*1996*	599
Jane	*Bonds*	*1998*	523
Dwayne	*Pipe*	*1988*	-566

Composite Key	Value
Chris P-Bacon-1996	599
Jane-Bondts-1998	523
Dwayne-Pipe-1988	-566

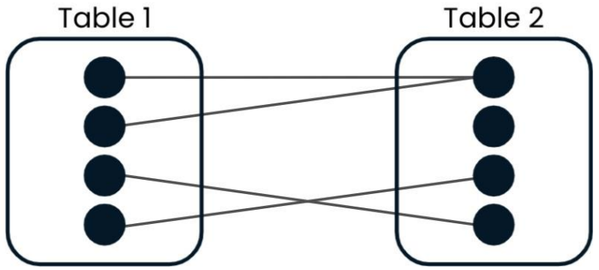
Cardinality

- A measure of the relationship between rows of two given tables
- **Many-to-one/One-to-many**: most commonly used
  - Connect **one** row from the dimension to **one or more** rows in the fact table

One-to-many



Many-to-one

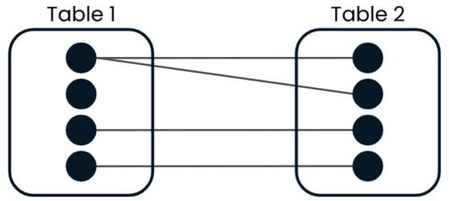


Cardinality

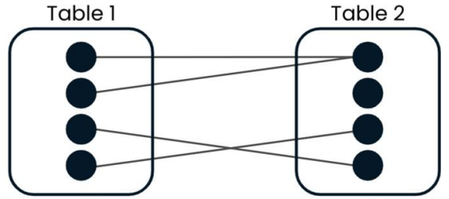
- Less common:
  - **One-to-one**

  - **Many-to-many**

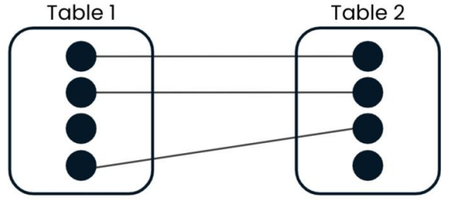
One-to-many



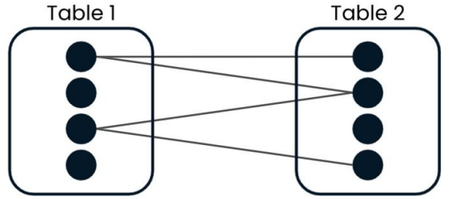
Many-to-one



One-to-one



Many-to-many



Demo

2. Granularity, Measures, and Hierarchies

Understanding granularity

- **Granularity**: at what level is the data stored with respect to dimensions?
- The minimum level of detail to query on
- Define granularity with *"by"* statements:
  - E.g. by customer, by product, by day
  - E.g. by id, by NAICS[1] code, by establishment age, by year

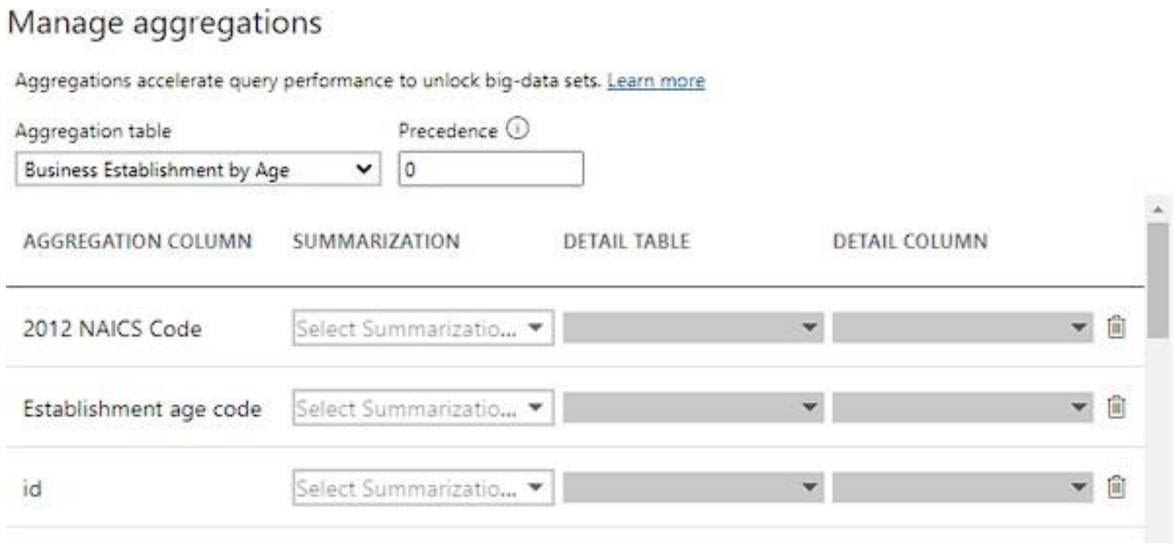
id	2012 NAICS Code	Establishment age code	Year	Number of firms	Number of establishments	Number of employees	N
0100000US	31-33	110	1978	0	0	0	
0100000US	31-33	110	1979	0	0	0	
0100000US	31-33	110	1980	0	0	0	
0100000US	31-33	110	1981	0	0	0	
0100000US	31-33	110	1982	0	0	0	
0100000US	31-33	110	1983	0	0	0	
0100000US	31-33	110	1984	0	0	0	
0100000US	31-33	110	1985	0	0	0	

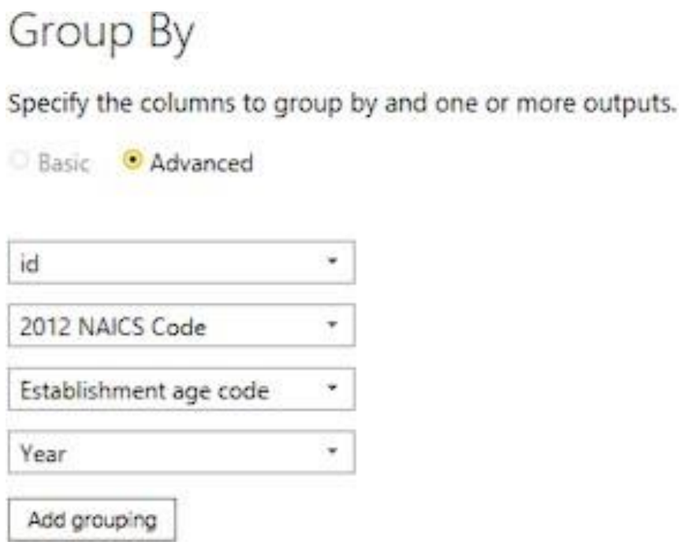
[1] NAICS: North American Industry Classification System

Handling granularity in Power BI

- Getting to a **finer** grain: not advisable!

- Getting to a **coarser** grain: aggregations and grouping
  - **Better query performance** with fewer rows
  - Smaller cache sizes and **faster refresh time**





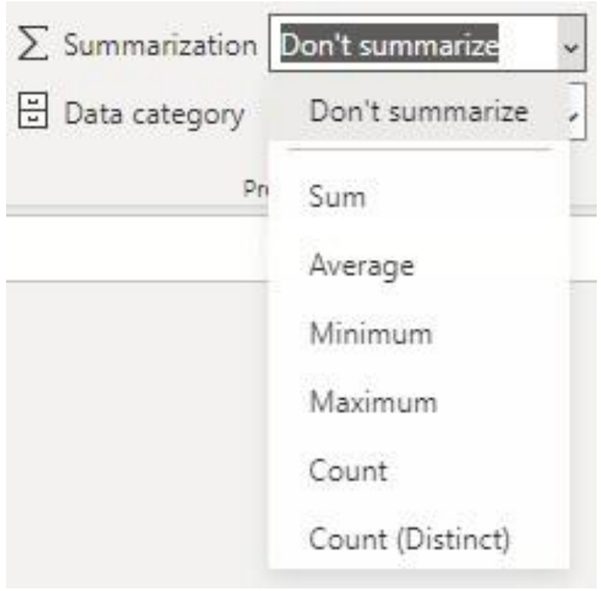
Measures

- Fields or combinations of fields which can be aggregated or calculated
  - Comes directly from fact data
  - New measures can be calculated as well

id	2012 NAICS Code	Establishment age code	Year	Number of firms	Number of establishments	Number of employees	N
0100000US	31-33	110	1978	0	0	0	
0100000US	31-33	110	1979	0	0	0	
0100000US	31-33	110	1980	0	0	0	
0100000US	31-33	110	1981	0	0	0	
0100000US	31-33	110	1982	0	0	0	
0100000US	31-33	110	1983	0	0	0	
0100000US	31-33	110	1984	0	0	0	
0100000US	31-33	110	1985	0	0	0	
0100000US	31-33	110	1986	0	0	0	

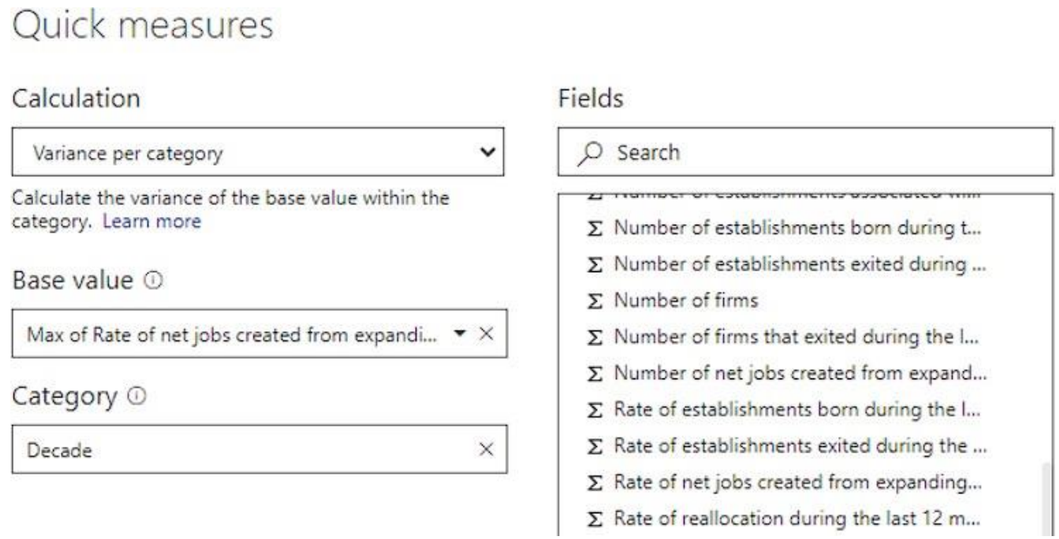
Creating measures

- Numeric values are automatically converted to measures and aggregated by the sum



- Create your own measures in Power BI using DAX

- Create specific types of calculations using a dialog: **Quick measures**



- Great for learning how to create moderately complex measures

Hierarchies

Allow users to drill down into data dimensions

Natural hierarchies

- **Levels** of the hierarchy **exist** "in the real world"
- Year -> Month -> Day

Artificial hierarchies

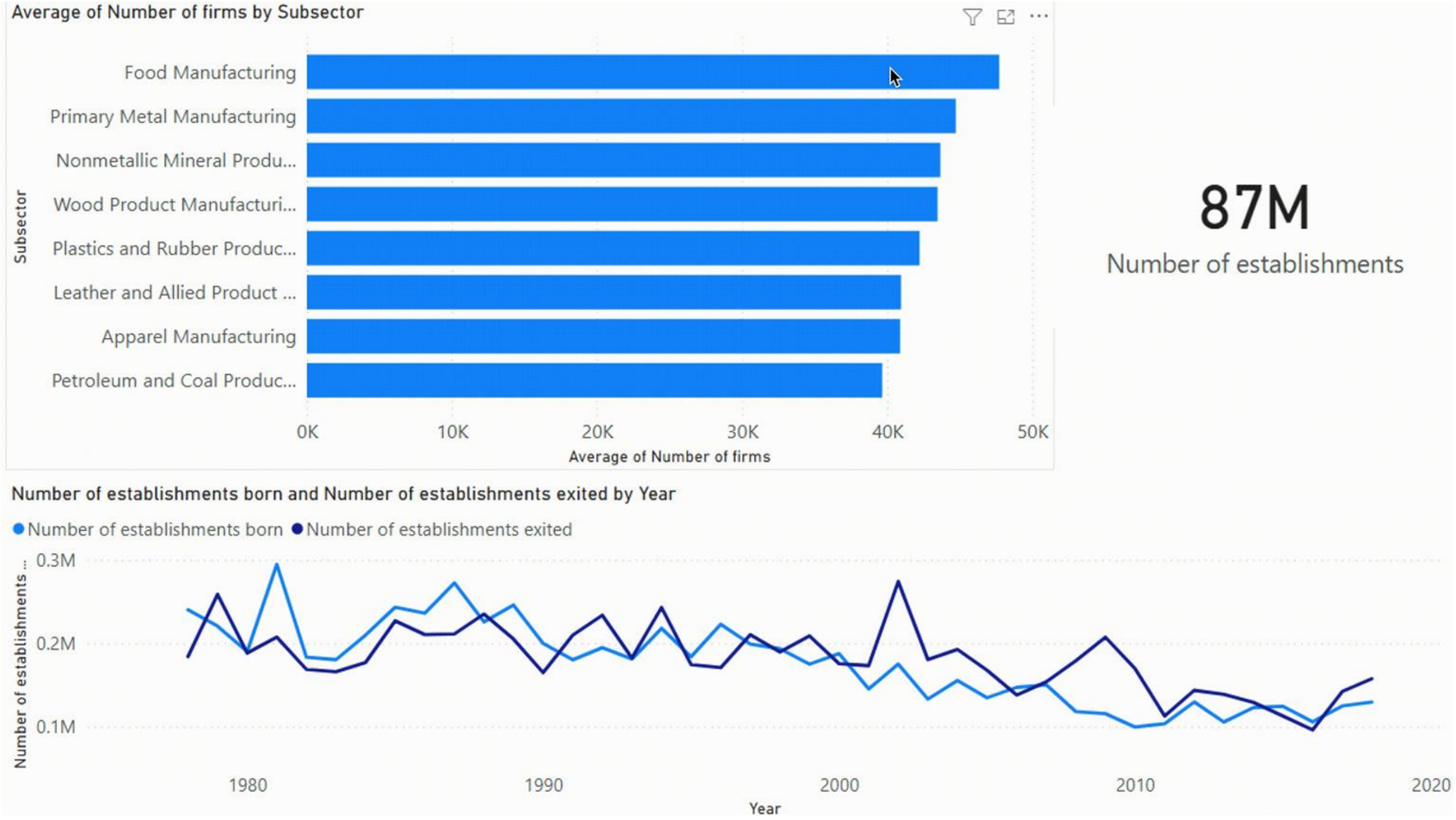
- **Levels** are **created** for querying purposes
- Intake year -> Favorite color -> Favorite sport

Demo

3. Advanced Data Modeling

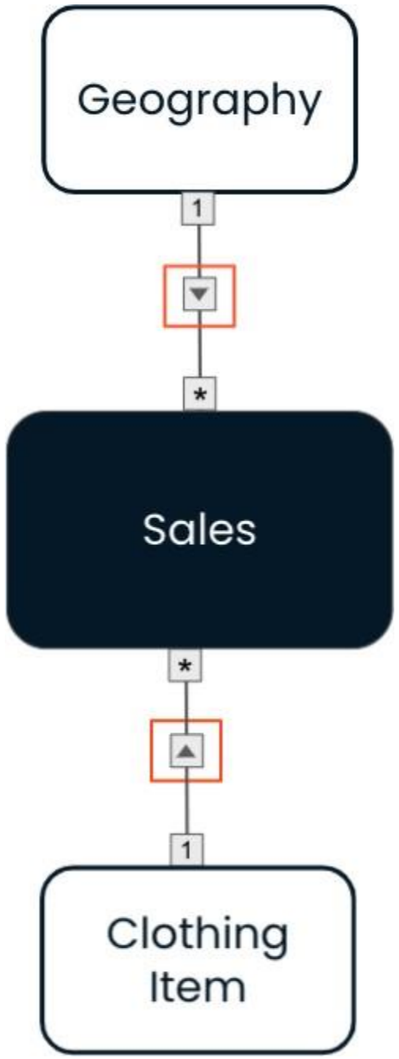
Cross filtering

Cross-filtering: Selecting a value in one visual narrows down visible data in other visuals



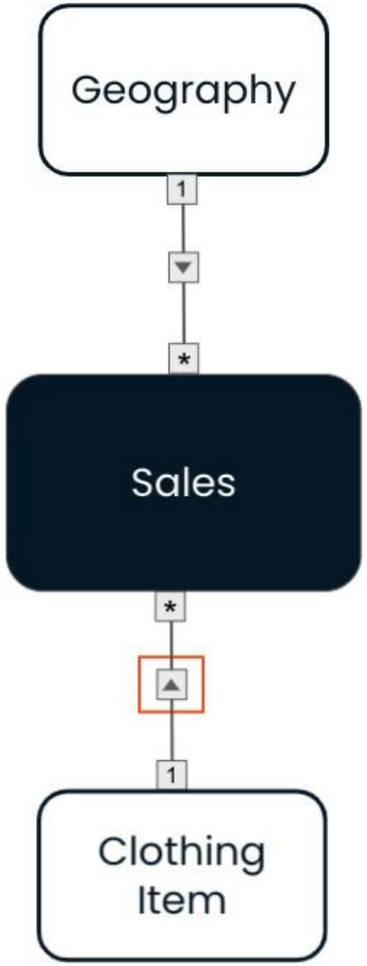
Filter direction

- All relationships have a cross filter direction
- Determines the direction that filters will propagate
- Example:
  - Geography -> Sales
  - Clothing Item -> Sales
- From Dimension to Fact



Filter direction

Data model:



Dimension - Clothing Item

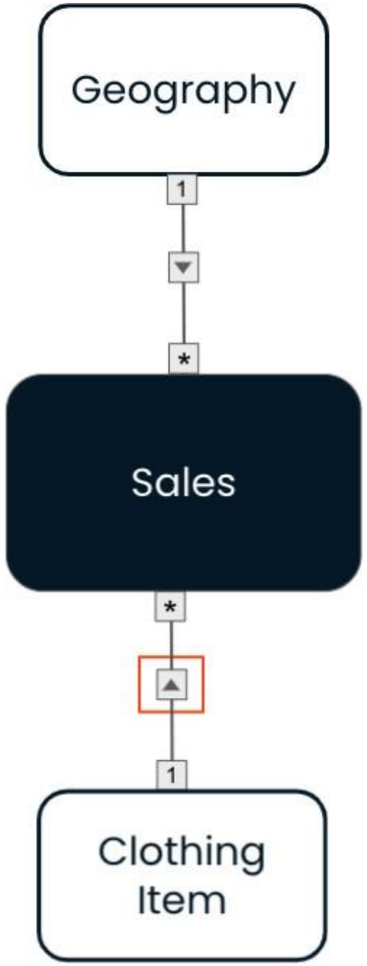
Product Id	Product
C1	T-shirt
C2	Socks
C3	Sweater

Fact - Sales

Id	Units	Amount	Product Id
001	3	60	C2
002	2	10	C1
003	1	70	C3
004	1	50	C3
005	5	50	C3

Filter direction

Data model:



Dimension - Clothing Item

Product Id	Product
C1	T-shirt
C2	Socks
C3	Sweater

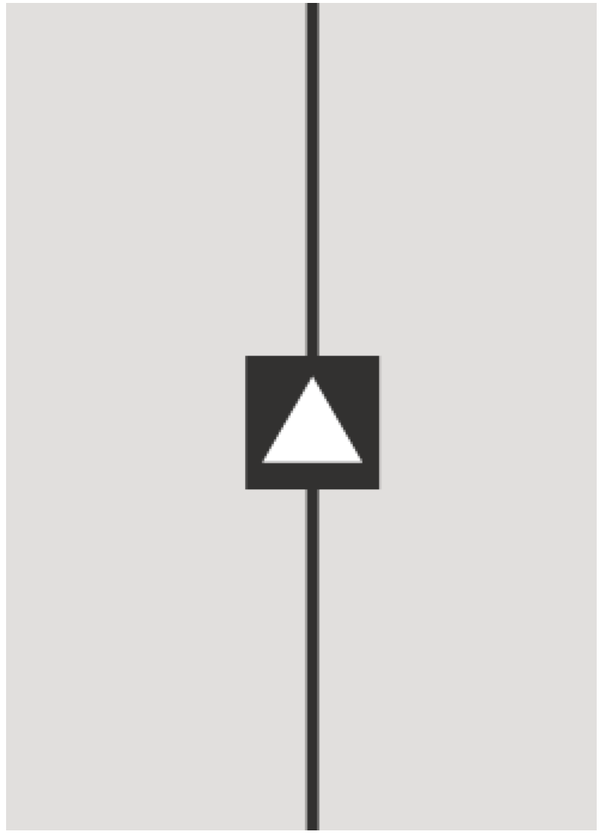
Fact - Sales

Id	Units	Amount	Product Id
001	3	60	C2
002	2	10	C1
003	1	70	C3
004	1	50	C3
005	5	50	C3

Filter direction options

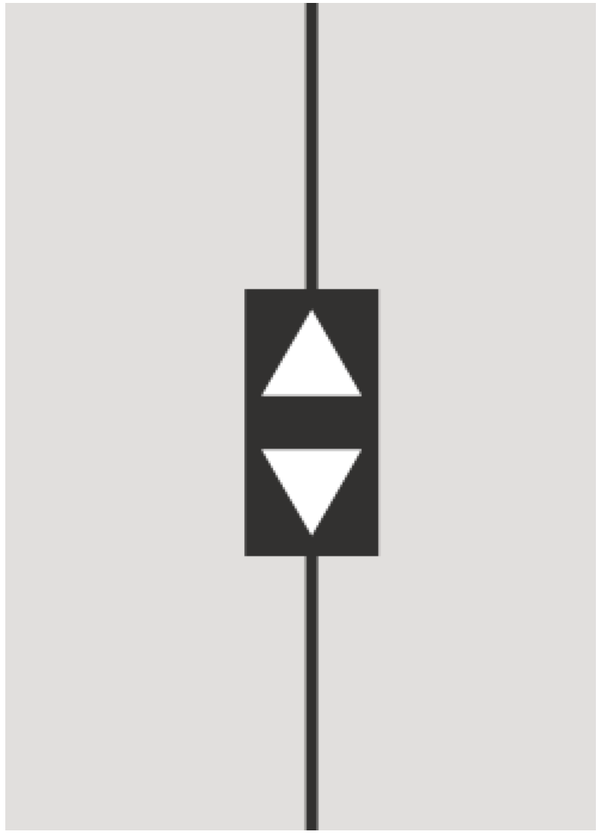
Single direction

Filter in one direction



Bi-directional

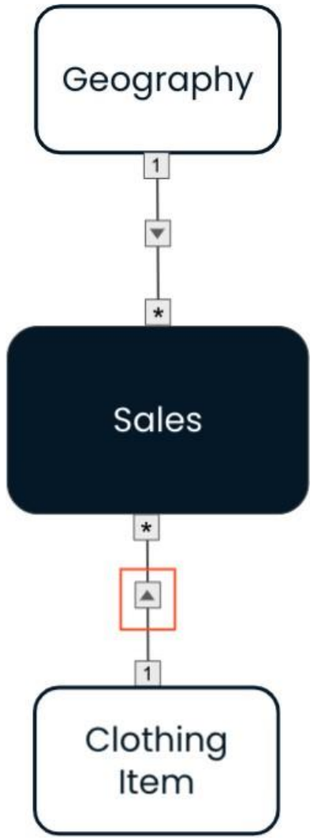
Filter in both directions

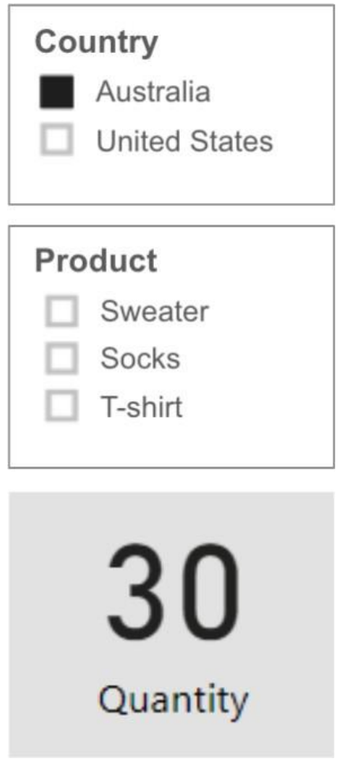


Bi-directional filtering: use case

Show only relevant slicer entries

Data model: Report view:





Bi-directional filtering: use case

Only sweaters were sold in Australia

Fact - Sales

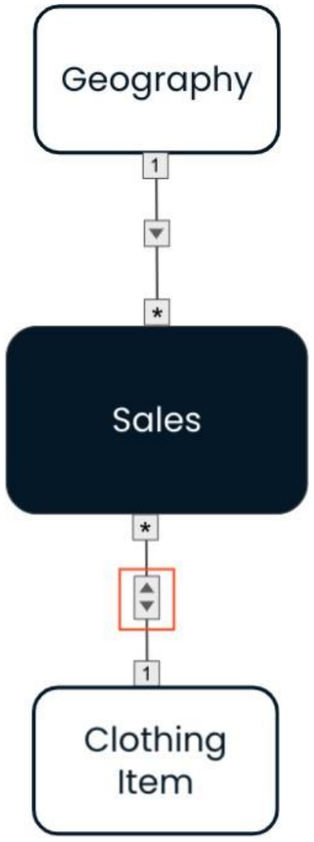
Id	Units	Amount	Product Id	Country Id
001	3	60	C2	US
002	2	10	C1	US
003	1	70	C3	AU
004	1	50	C3	AU
005	5	50	C3	AU

*Product Id: C3 = Sweater, Country Id: AU = Australia

Bi-directional filtering: use case

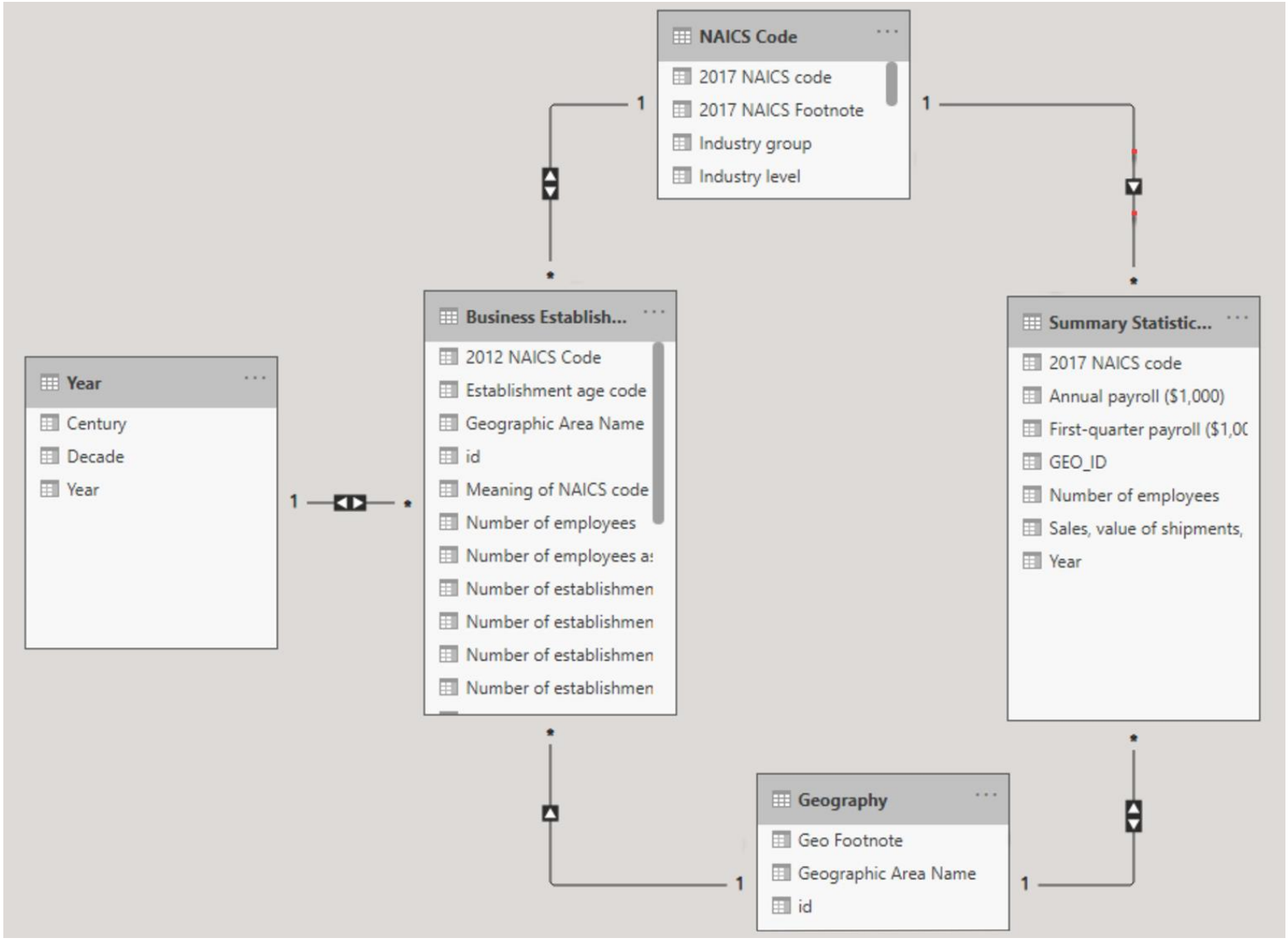
Show only relevant slicer entries

Data model: Report view:



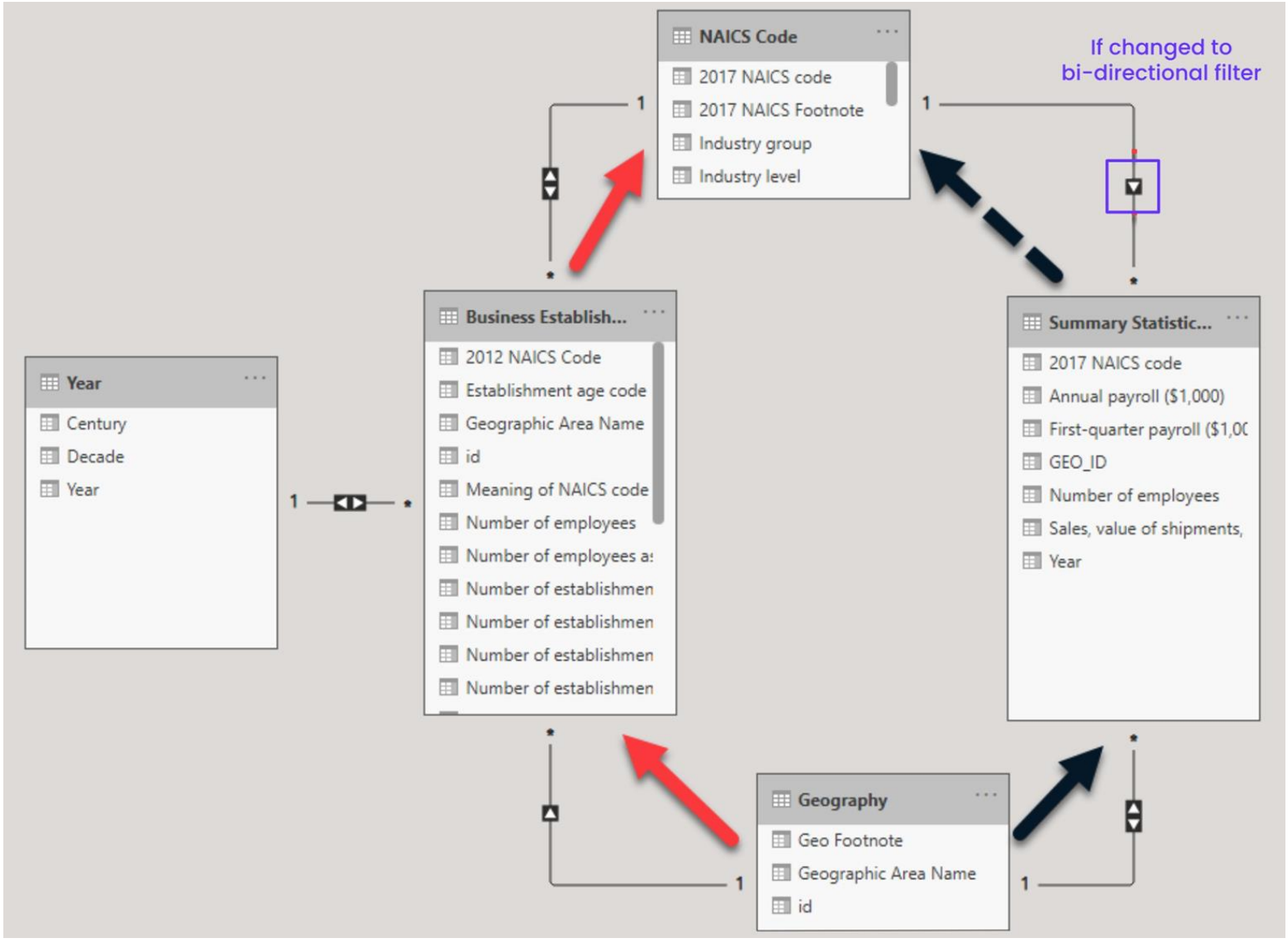
Bi-directional filtering and paths

Bi-directional filters cannot allow for two separate paths between two tables



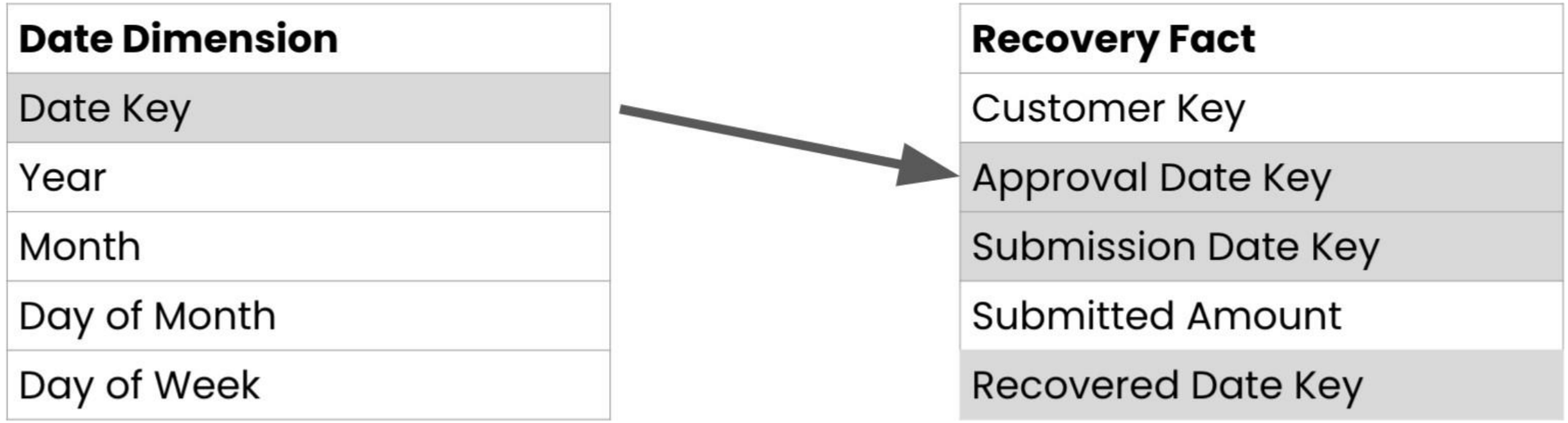
Bi-directional filtering and paths

Bi-directional filters cannot allow for two separate paths between two tables



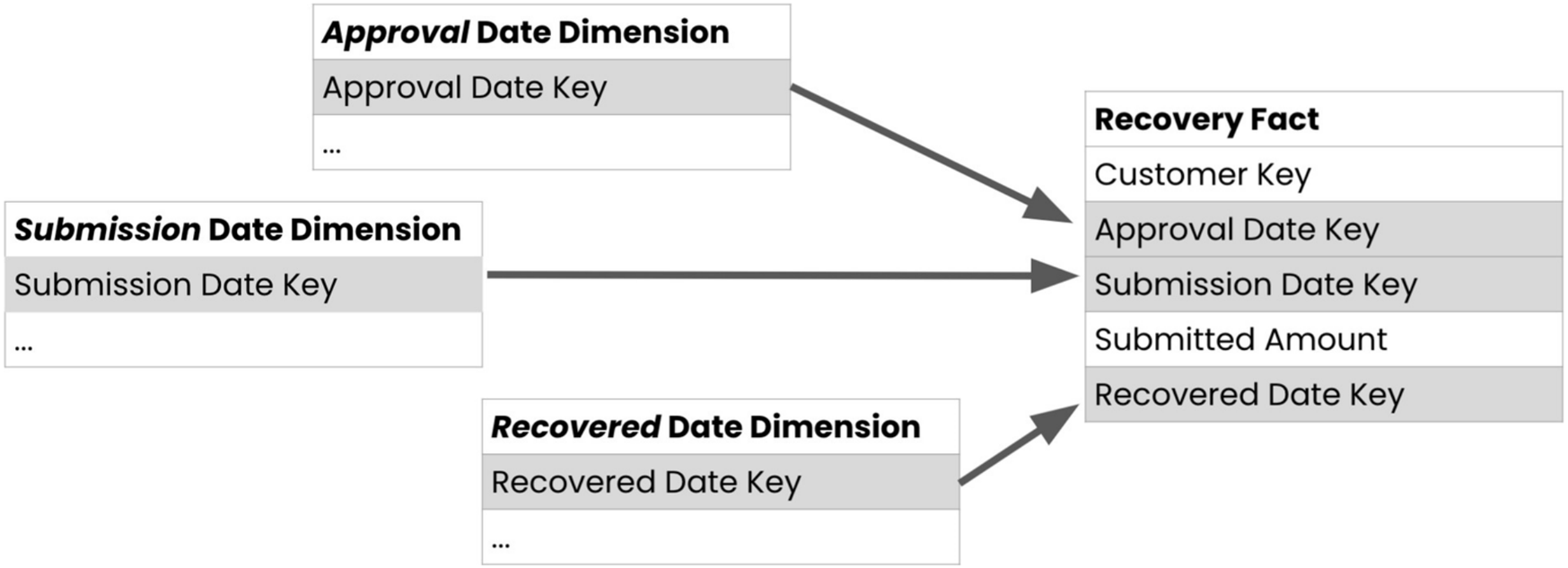
Role-playing dimensions

- Sometimes we need to create multiple relationships between tables



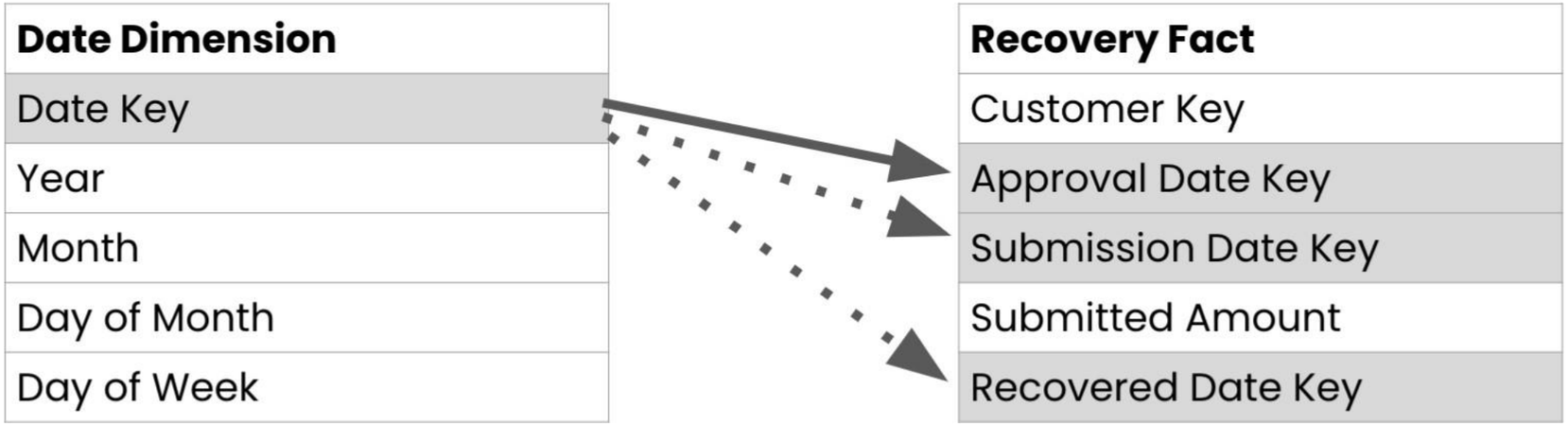
Role-playing dimensions

- Kimball model
- **Role-playing dimension**:
  - Dimension that can filter related facts differently
- Typically implemented as views of the Date dimension



Role-playing dimensions in Power BI

- Create multiple relationships on a dimension, but only one is active



- Use `USERELATIONSHIP()` in DAX to specify which relationship to use:

```
Measure Name = CALCULATE(<Measurement function>,
                         USERELATIONSHIP(<Dimension Key Column>, <Fact Key Column>)
```

Demo

4. Identifying Performance Problems

Resolving performance problems

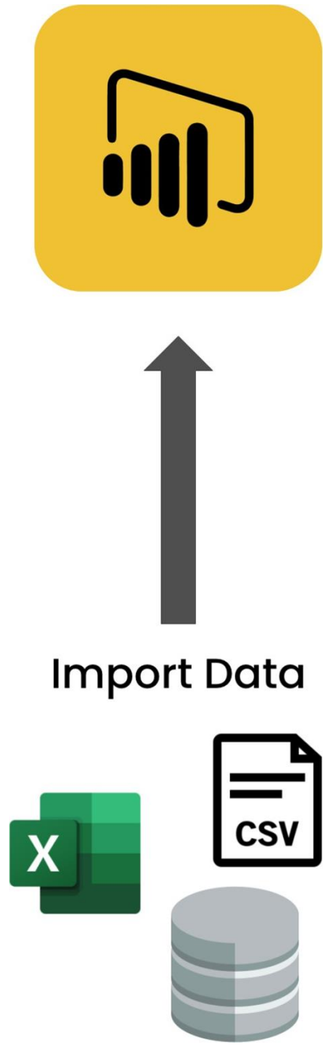
Performance problems

Where things can go wrong:

- Data import
- Querying the database with DirectQuery
- Displaying visuals
- Calculated versus computed columns
- Inefficient relationships
  - Many-to-many relationships
  - Bi-directional cross-filtering

Optimizing data import

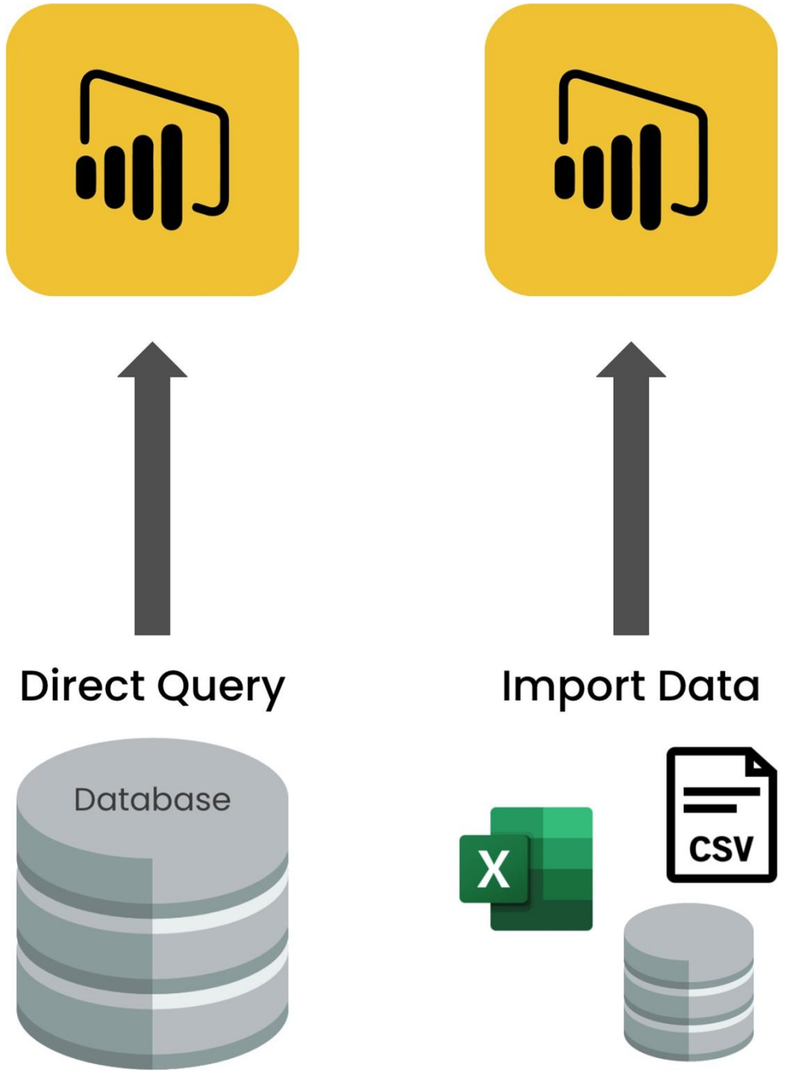
- Remove unnecessary rows and columns
- Choose correct data types
  - Numeric data takes less space
  - Casting and aggregating data is slower
- Group and summarize data
  - Store less data on disk
  - Get to aggregate results faster



Optimizing Direct Query

- Two ways to connect to data:
  - **Import model**: stores data in Power BI
  - **Direct Query**: directly queries the database

- Limit parallel queries
- Relational database advice
  - Write efficient SQL queries
  - Use appropriate indexes
  - Get the right columns and rows



Calculated versus computed columns

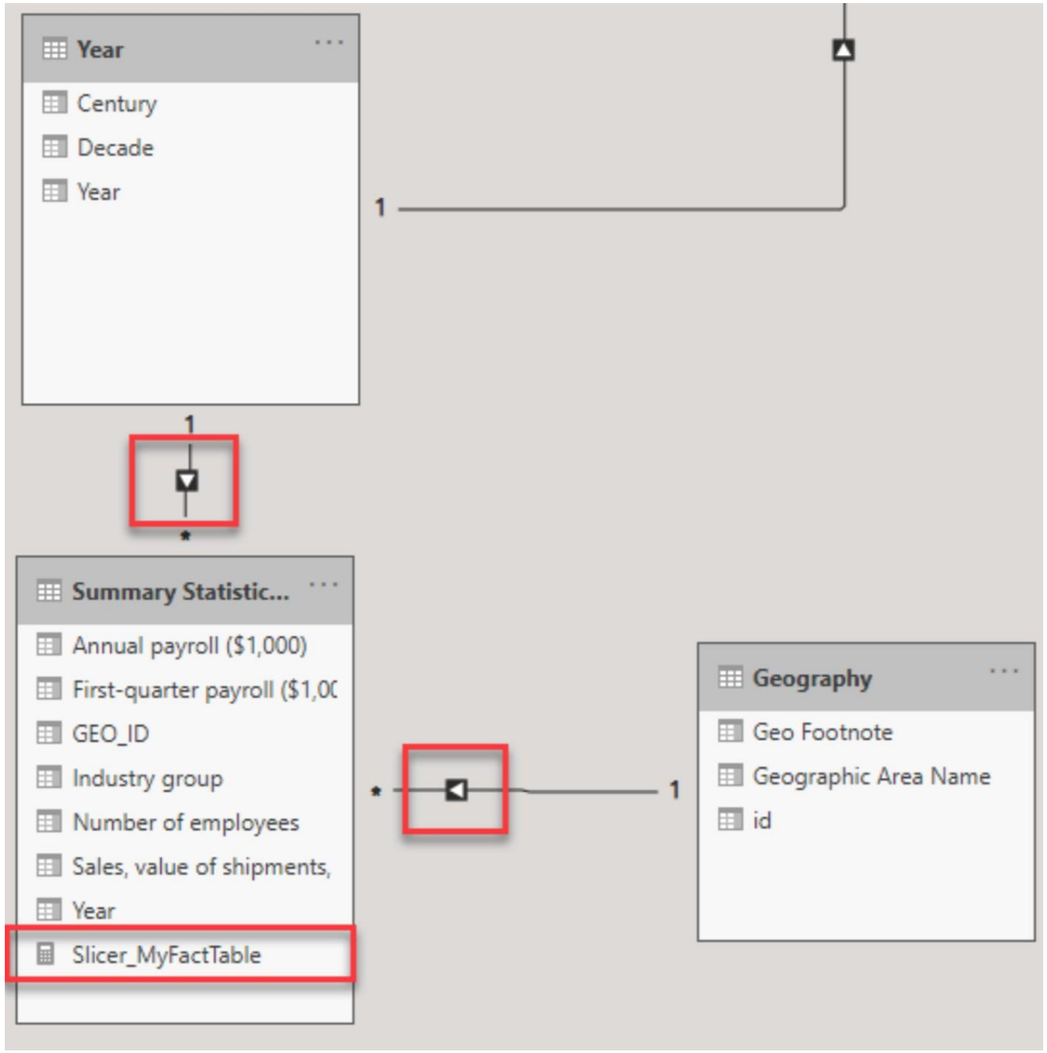
Build custom columns with:

Calculated columns	Computed columns
DAX	Power Query (M)
Fast for *simple* calculations	Fast for *simple* calculations
Slow for *complex* calculations	Fast for *complex* calculations
Generated per visual at runtime	Generated once at import time

Removing bi-directional filtering using filter measures

- Use case for bi-directional filtering
  - Find relevant slicer entries between dimensions
- We can create filter measures to avoid bi-directional relationships for the third use case!

Removing bi-directional filtering using filter measures



Removing bi-directional filtering using filter measures

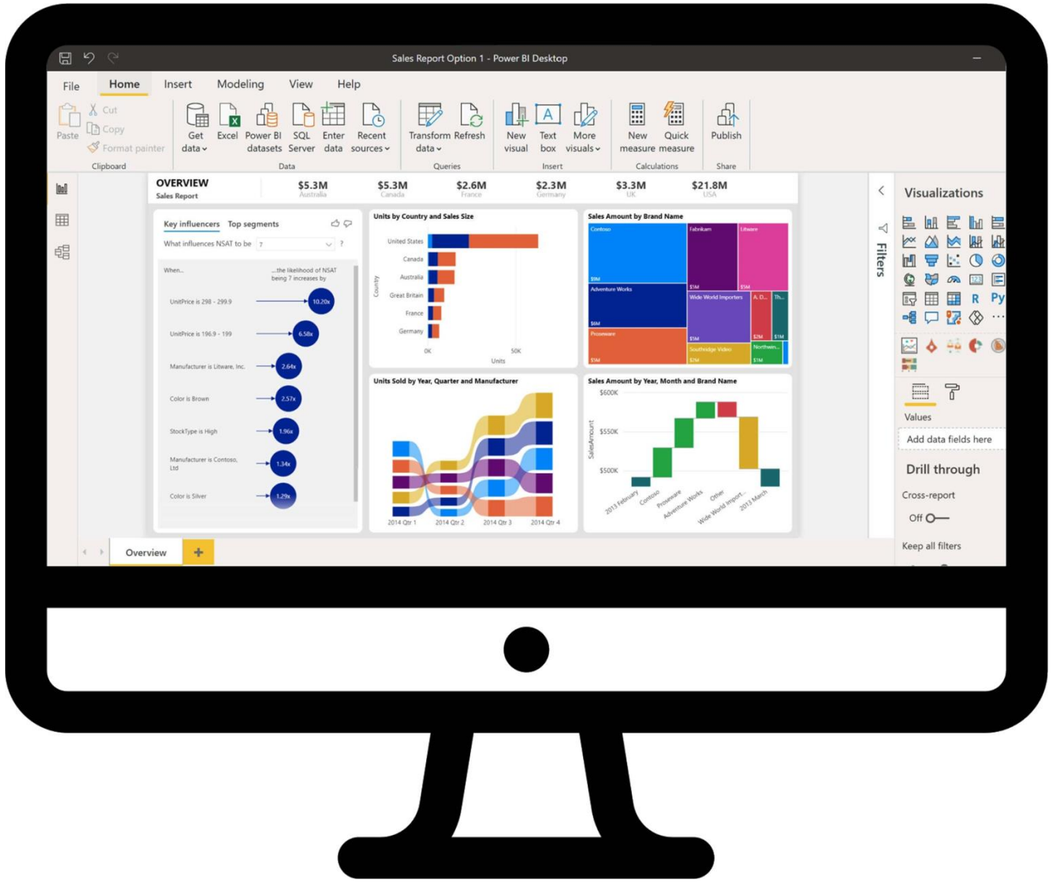
1) Create a filter measure in DAX:

```
Slicer_MyFactTable = INT(NOT ISEMPTY('My Fact Table'))
```

- Returns 1 if at least one value in the fact table
- Returns 0 if no values in the fact table

2) Add a visual filter to the slicer and set where `Slicer_MyFactTable = 1`

Displaying visuals



- Use restrictive filters to minimize data
- Show as little data as possible on visuals
- Limit the number of visuals on report pages
- Use only fast custom visuals

Demo

Congratulations!

Intermediate Data Modeling

- Date dimensions and relationships
- Hierarchies and granularity
- Bi-directional cross filtering
- Role-playing dimensions
- Performance optimization