# Date dimensions and relationships

DATA MODELING IN POWER BI

**Maarten Van den Broeck**
Content Developer

datacamp

# 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

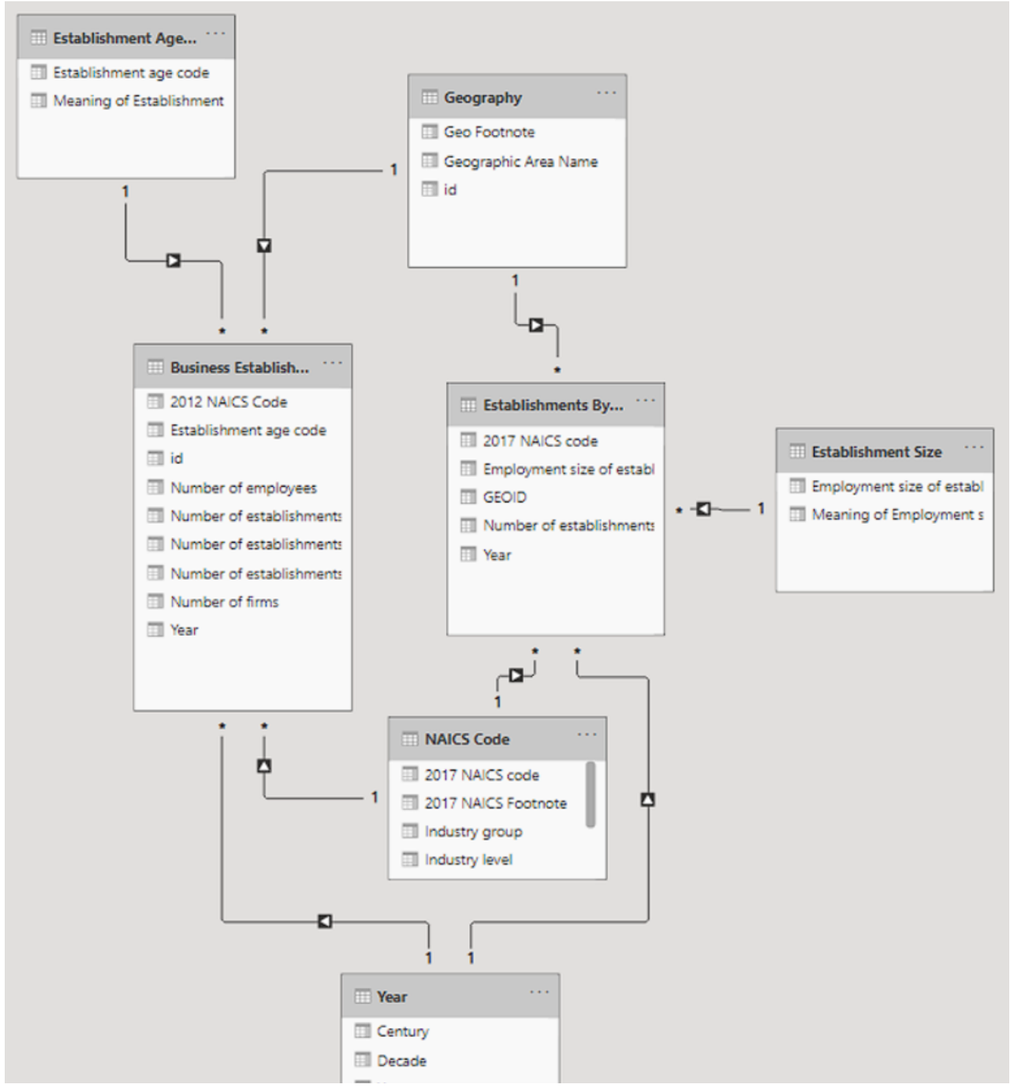
```
Month_Year =
DISTINCT(
    SELECTCOLUMNS(
        CALENDAR(DATE(1950, 1, 1),
                 TODAY()),
        "Month", MONTH([Date]),
        "Year", YEAR([Date])
    )
)
```

- `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
- Only keep unique rows

| 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

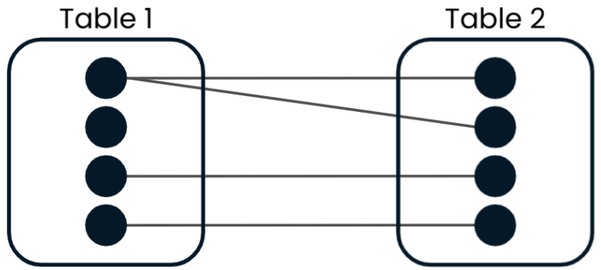
| First Name | Last Name | Birth year | Value |
|---|---|---|---|
| *Chris P* | *Bacon* | *1996* | 599 |
| *Jane* | *Bonds* | *1998* | 523 |
| *Dwayne* | *Pipe* | *1988* | -566 |

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

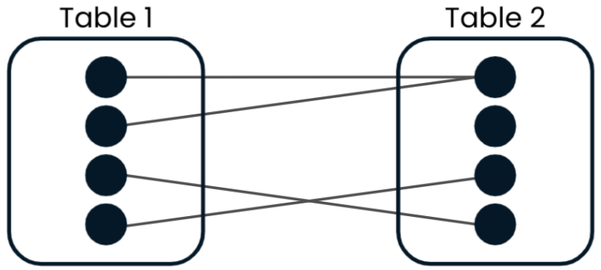
# Cardinality

- 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



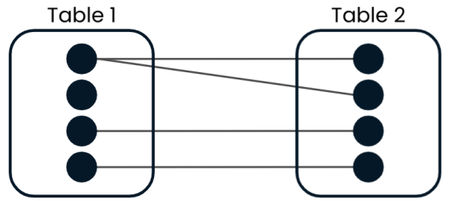


# Cardinality

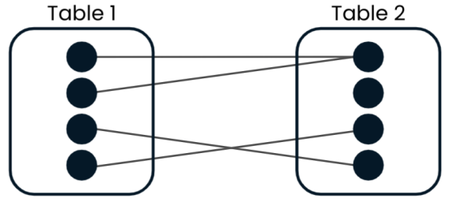
- Less common:
  - **One-to-one**


  - **Many-to-many**

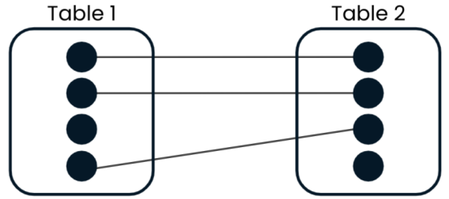

One-to-many



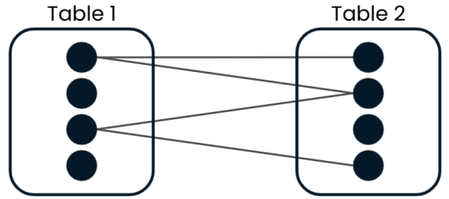

Many-to-one




One-to-one




Many-to-many



# Let's practice!

DATA MODELING IN POWER BI

# Date dimensions and relationships in Power BI

DATA MODELING IN POWER BI

**Maarten Van den Broeck**
Content Developer at DataCamp

datacamp

# Let's practice!

DATA MODELING IN POWER BI

# Granularity, measures, and hierarchies

DATA MODELING IN POWER BI

**Maarten Van den Broeck**
Content Developer at DataCamp

datacamp

# 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

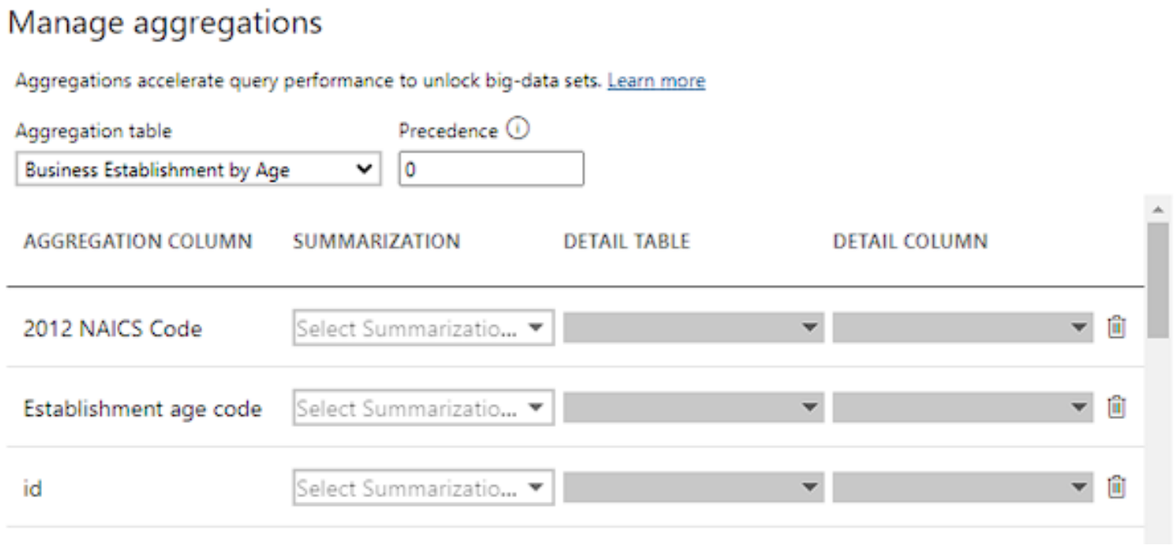
| id | 2012 NAICS Code | Establishment age code | Year | Number of firms | Number of establishments | Number of employees | N |
|---|---|---|---|---|---|---|---|
| 0100000US | 31-33 | 110 | 1978 | 0 | 0 | 0 | |
| 0100000US | 31-33 | 110 | 1979 | 0 | 0 | 0 | |
| 0100000US | 31-33 | 110 | 1980 | 0 | 0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

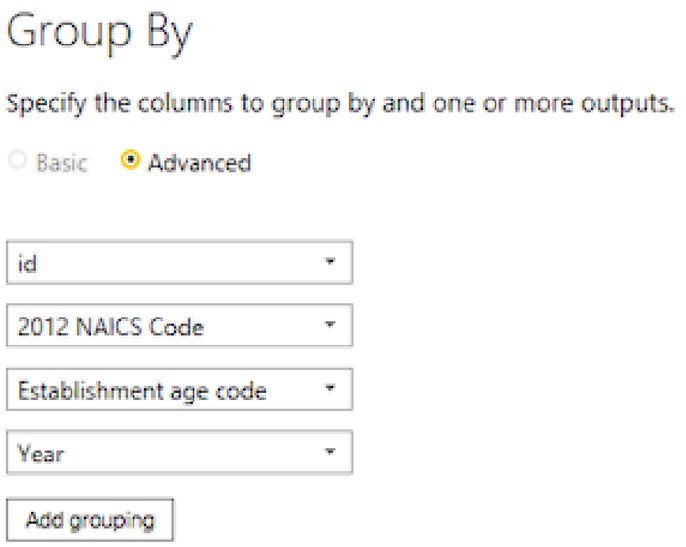
[1] NAICS: North American Industry Classification System

# 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**



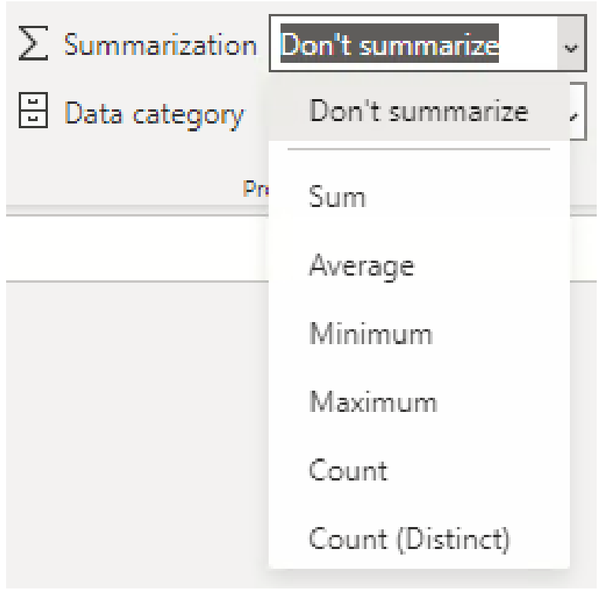
# Measures

- 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 | |

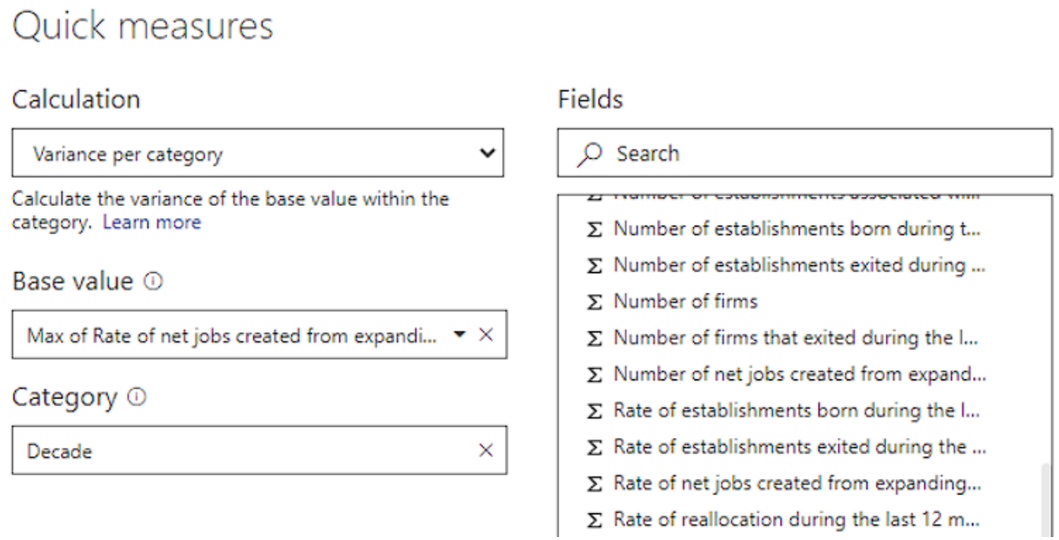
# Creating measures

- 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

# Let's practice!

DATA MODELING IN POWER BI

# Hierarchies and measures in Power BI

DATA MODELING IN POWER BI

**Maarten Van den Broeck**
Content Developer at DataCamp

datacamp

# Let's practice!

DATA MODELING IN POWER BI