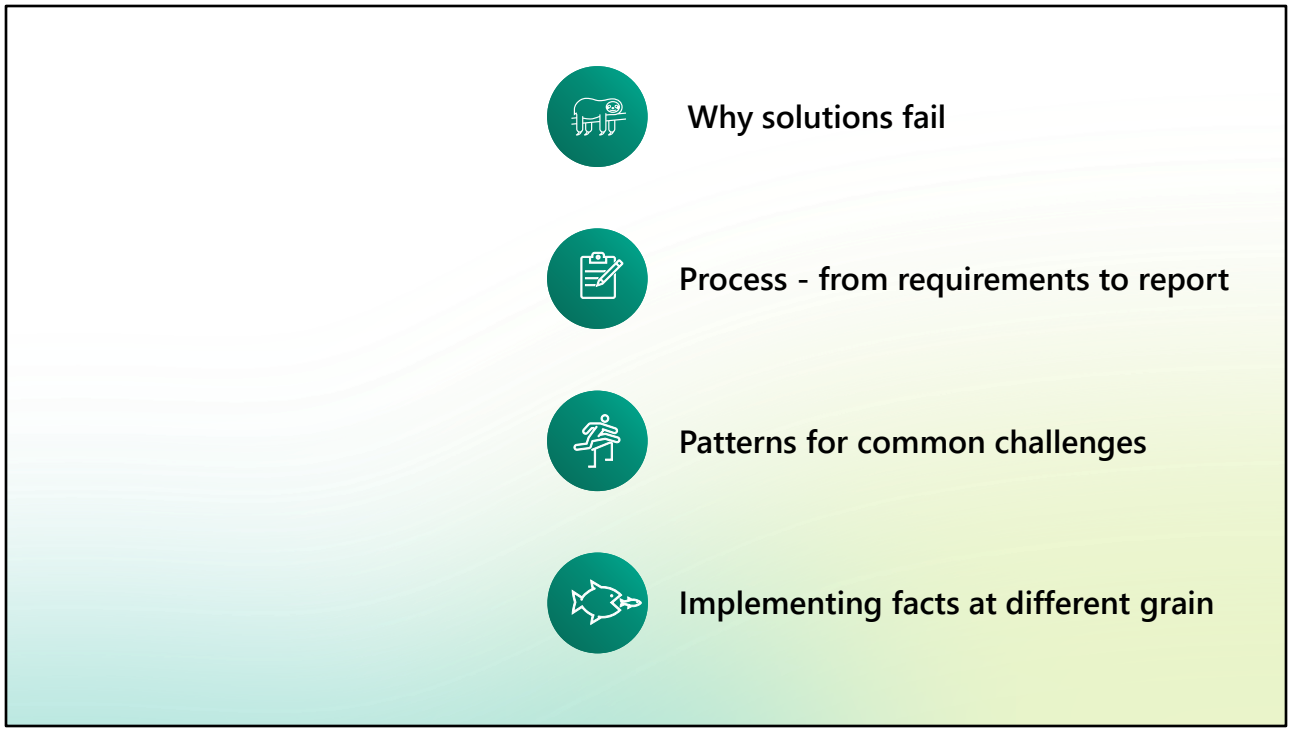

Power BI Development Process
A Roadmap to Excellent Solutions



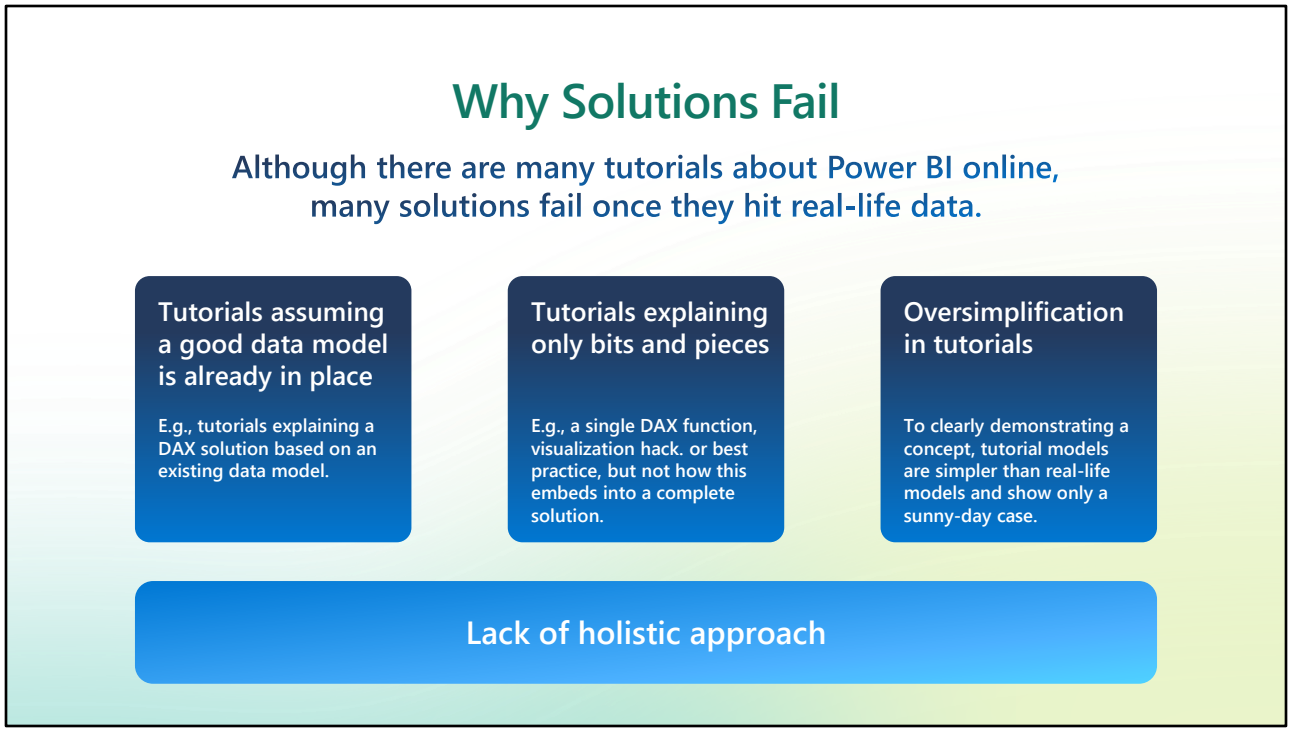
Agenda
Why this presentation?

Although there is so much training material and so many YouTube channels out there, I still frequently see report creators struggle with applying the concepts to real life solutions.

Solutions fail despite people having basic knowledge of star schema.

We will explain what grain is and how to implement it in Power BI.

Why solutions fail



Each of these tutorial approaches have a valid purpose and there is nothing wrong about them. But they leave a gap in the learning journey.

Despite having many articles and tutorials online about star schema modeling, DAX programming, etc. I frequently see messy data models, DAX code and report layouts being slow and producing wrong or surprising results in real-life.

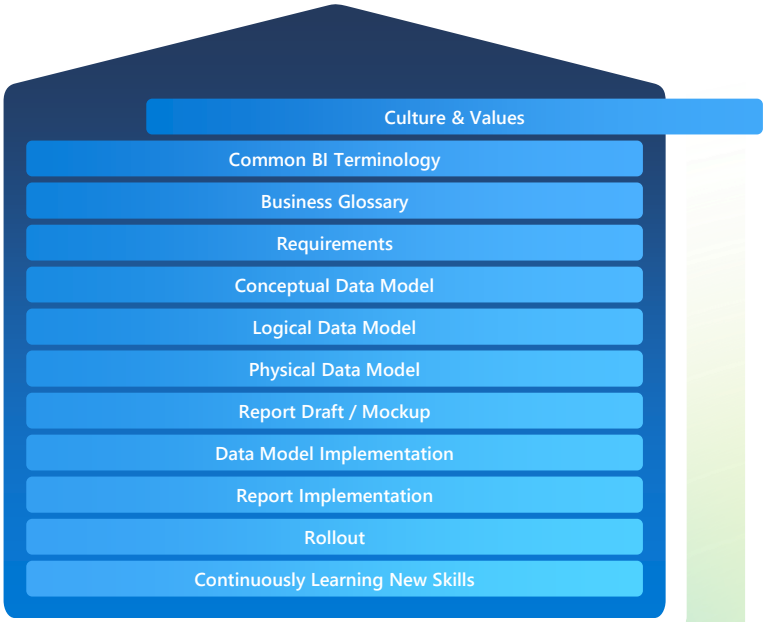
Just consuming an unstructured set of tutorial content, sometimes even just looking for a solution after already running into it, tricks beginners into jumping directly from requirements or just a report idea straight into implementation in Power BI, without laying out a concept first. Problems arise and fixing them in the implementation is a lot of effort.

In this session we are learning a structured process to first conceptualize and then implement a solution.

Process House

This holistic approach is the process house.

The driver of what you need in your data model is not the source systems but what you need to meet your analytical requirements. This refers to the fields and their structure in the semantic model.

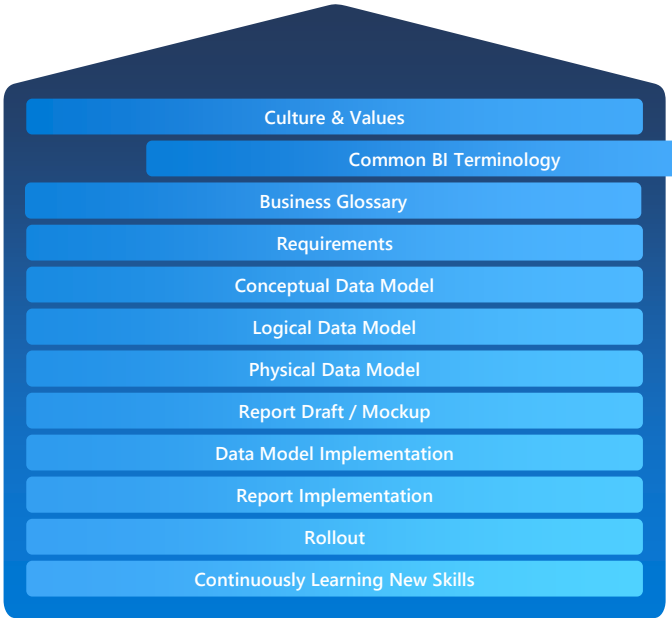


- Commitment
- Efficient communication
- Collaboration
- Support
- Transparency
- Evolution of software
- Continuous learning
- **Love the outcome!**

Business stakeholders' availability and involvement is crucial.

You can see whether a report is created with love, passion and commitment:

- Objects are named in the selection pane.
- Tables and fields in the semantic model have descriptions.
- Tables and fields have clear, natural language names,
- Unnecessary fields are removed or hidden.
- Model view is nicely arranged.
- DAX code is formatted and commented where reasonable
- PowerQuery steps are named specifically.
- PowerQueries and measures are arranged in folders in larger models



What is a ...

- Process
- Event
- Measure
- Dimension
- Attribute
- Category
- Value
- Hierarchy
- Conformed dimension
- Shrunken dimension
- Degenerate dimension
- Junk dimension or business profile
- Dashboard

E.g., in data warehousing "measure" usually refers to a values column in a fact table, whereas in Power BI it refers to a calculation on the data.

Is a dashboard a report, an overview page in a report, or a Power BI Dashboard, technically?

Also define terms like "good", "goal", or "success" for your project.

What is success?

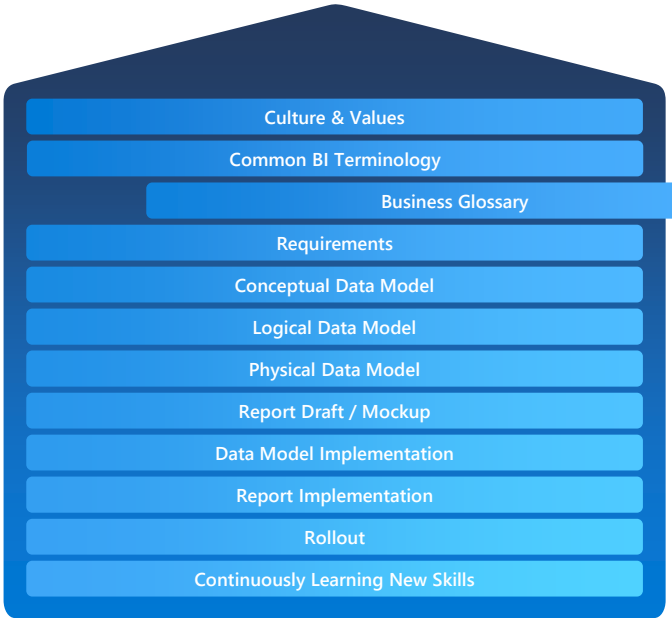
- A nice report?
- Correct numbers?
- Users frequently using the report?
- Users spending a lot of time in the report?

Or is it the impact coming from the insights:

- Increased revenue
- Increased profit
- Increased market share
- Reduced carbon footprint

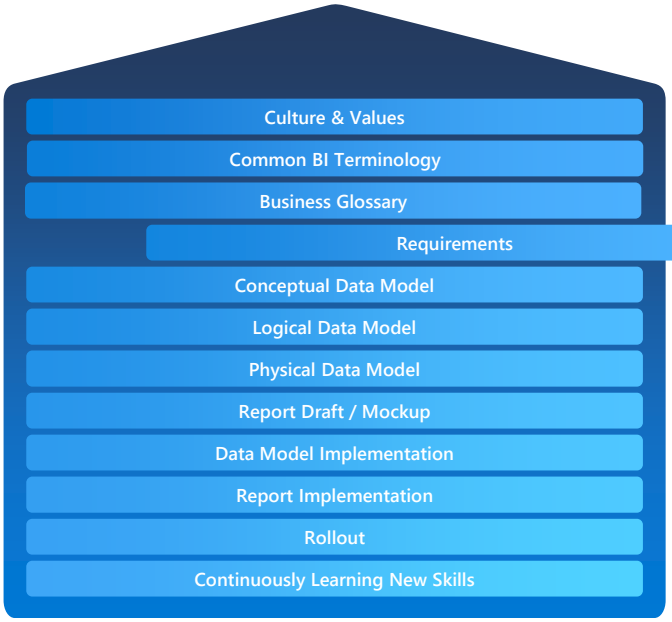
... whatever is the goal of the report.

=> A clear terminology is an enabler for efficient communication.



- Business terms
- Performance indicators
- KPI
- In **natural language**, always
- With formula, if applicable
- Example:

Customer acquisition rate measures the % of new customers acquired in a given period.



- Functional requirements / business requirements
- Non-functional requirements / technical requirements
- Use a template, e.g., KPI canvas
- Document
  - Task board, e.g., Jira
  - Word document
- Complete and clear wording

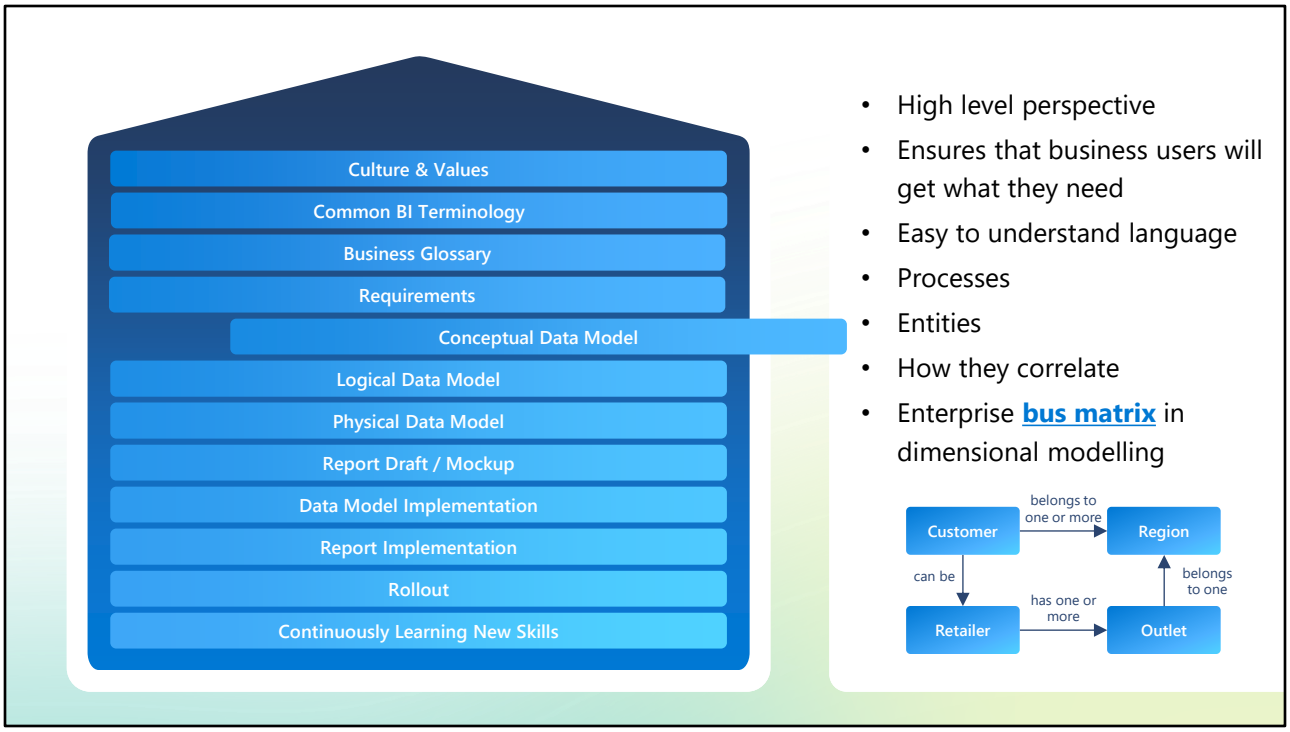
"Requirements" includes all levels of requirements specification:

- Goal: What shall the report be used for? What shall be achieved? How do you measure success of the report?
- Scope: What is part of the report – and what not?
- Personas: Who are the users? What are their expectations? What are their skills?
- Process: How will the report be used? How does report usage embed in daily business? What is the relevance of the report? Describe interaction scenarios.
- KPI definitions: If new KPIs are defined for this report, specify them. Otherwise copy or reference definition of existing KPIs.
- Functional requirements: Features of the report, pages, filters, visuals, etc.
- Non-Functional requirements: Expected amount of data, when to cut off historic data, performance requirements, security requirements, etc.

Document for later use, think about the next developer who should change something in a year or two.

Bernie Smith, KPI canvas: [How to define KPIs with our free KPI format sample template (madetomeasurekpis.com)](https://madetomeasurekpis.com/free-kpi-definition-template/) *https://madetomeasurekpis.com/free-kpi-definition-template/*

Requirements templates resources: [Power Bi dashboard requirement gathering template - Microsoft Fabric Community](https://community.fabric.microsoft.com/t5/Desktop/Power-Bi-dashboard-requirement-gathering-template/td-p/1092633) *https://community.fabric.microsoft.com/t5/Desktop/Power-Bi-dashboard-requirement-gathering-template/td-p/1092633*



The conceptual data model helps in communication with business.

- Helps identifying requirements
- Helps clarifying taxonomy

Don't use database object names, camel case wording, abbreviations, technical terms, etc. in conceptual data model.

The conceptual data model is technology agnostic.

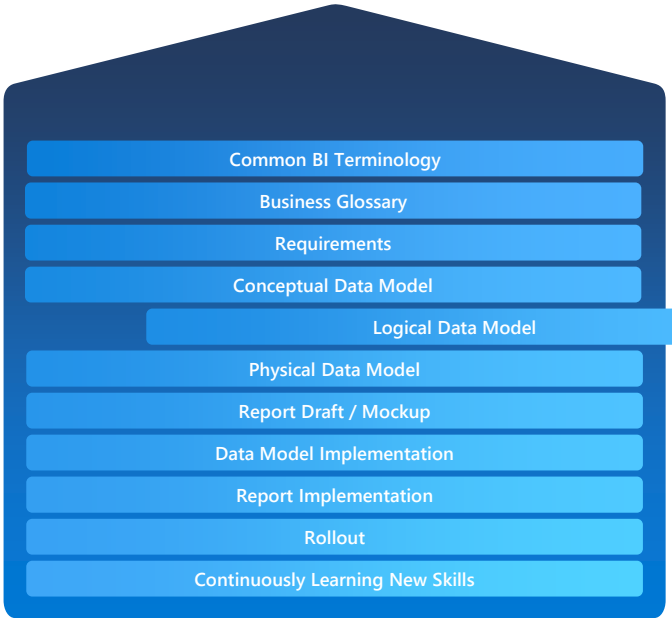
If you want to reduce conceptual effort, the conceptual model would be my preferred step. Choose focus topics for a conceptual data model and make sure you are completing the bus matrix.

Data modeling articles:

Data Modeling for Mere Mortals - Part 1: What is Data Modeling?! - Data Mozart (data-mozart.com)
https://data-mozart.com/data-modeling-for-mere-mortals-part-1-what-is-data-modeling/

What Are Conceptual, Logical, and Physical Data Models? | Vertabelo Database Modeler
https://vertabelo.com/blog/conceptual-logical-physical-data-model/

Enterprise Data Warehouse Bus Architecture - Kimball Group *https://www.kimballgroup.com/data-warehouse-business-intelligence-resources/kimball-techniques/kimball-data-warehouse-bus-architecture/*



- Defines what data structures are needed, based on the requirements
- Blueprint for implementation
- **Star schema** for Power BI semantic models
- Relationships
- Cardinalities
- Hierarchies
- Candidate keys

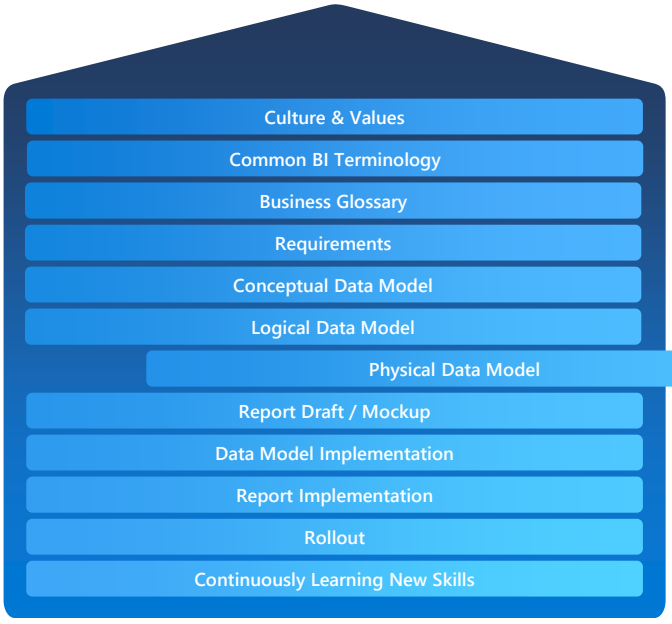
Defines what data is needed for the requirements, not what data is there in the source systems.
Prevents you from getting lost in the complexity of data sources.
The logical data model is technology agnostic.

Entity-relationship diagrams tool recommendation: dbdiagram.io - Database Relationship Diagrams Design Tool *https://dbdiagram.io/*

Data modeling recommended reading:

Power BI star schema documentation: Understand star schema and the importance for Power BI - Power BI | Microsoft Learn *https://learn.microsoft.com/en-us/power-bi/guidance/star-schema*

Ralph Kimball, The Data Warehouse Toolkit: The Data Warehouse Toolkit, 3rd Edition - Kimball Group *https://www.kimballgroup.com/data-warehouse-business-intelligence-resources/books/data-warehouse-dw-toolkit/*



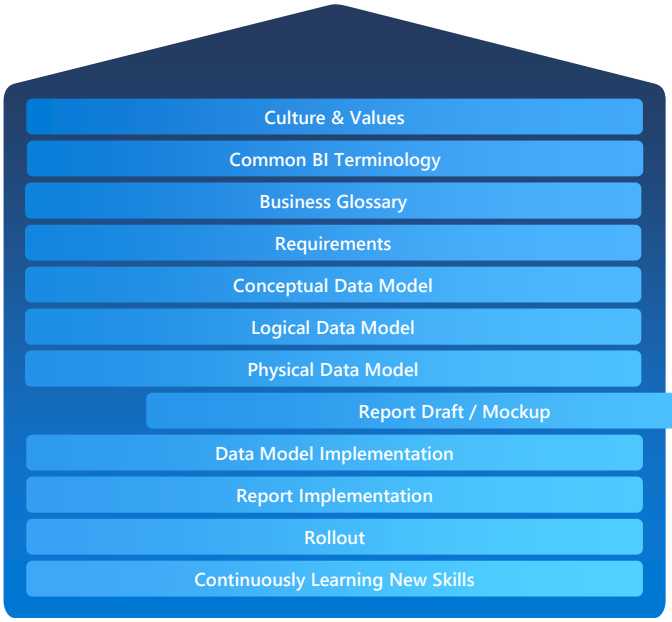
- Implementation specific adaptions to the data model
- Enrich data model with technical requirements
- Surrogate keys
- Historization / slowly changing dimensions
- Time intelligence support columns
- Aggregations & Allocations
- **If it can be solved in the data model, you don't need to solved it in DAX measures!**

Don't hesitate to introduce transformations and additional fields on the source data if this makes the data model more efficient.

E.g., Power BI cannot deal with composite (multi-column) relationships and needs surrogate keys to implement them.

The physical data model is technology specific.

Slowly changing dimensions article: Slowly Changing Dimensions - Kimball Group
https://www.kimballgroup.com/2008/08/slowly-changing-dimensions/



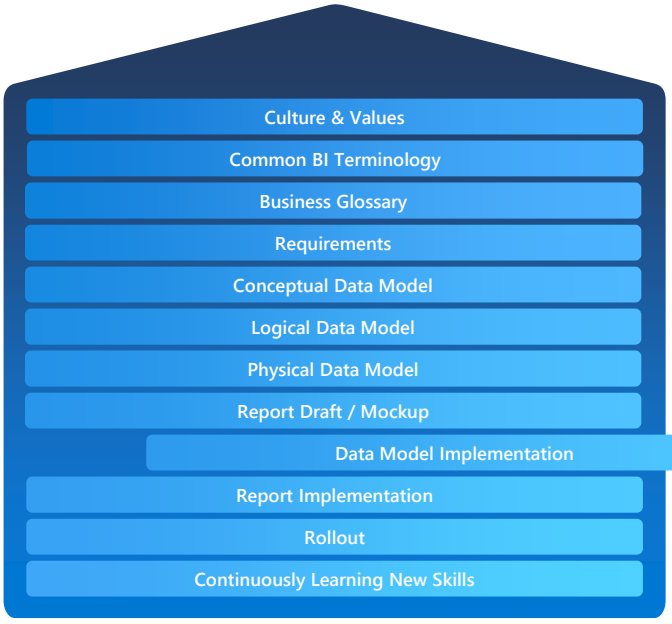
- Clarify usage workflows and interactions.
- Apply a report structure, e.g., from big picture overview to detailed analytics.
- Use design tool, e.g., Figma
- Use a template
- Use a common design language, e.g., IBCS
- Use visuals that match with the analytics goal
- Use consistent colors
- Pay attention to accessibility

Use consistent colors in all reports, e.g., for good, bad, same product group shown in same color everywhere. In Power BI, colors can be loaded as attributes in dimension tables.

Figma design tool: Figma: The Collaborative Interface Design Tool *https://www.figma.com/*
IBCS visual language: IBCS - International Business Communication Standards *https://www.ibcs.com/*

Report design process on YouTube:

1. Design Thought Process for Power BI Reporting (with Miguel Myers) (youtube.com)
   https://www.youtube.com/watch?v=dhHL0Uo3Wgs
2. Report Design Framework - Season 6 Ep.1 (youtube.com)
   https://www.youtube.com/watch?v=Jx6uvvT2QMI
3. PowerBI Visual Evolution - Lunch and Learn - Everyone (youtube.com)
   https://www.youtube.com/watch?v=hs1irtM3PR8&list=PLMGL7BRUz8sypmPOgTft-lvn1yqvYaRv



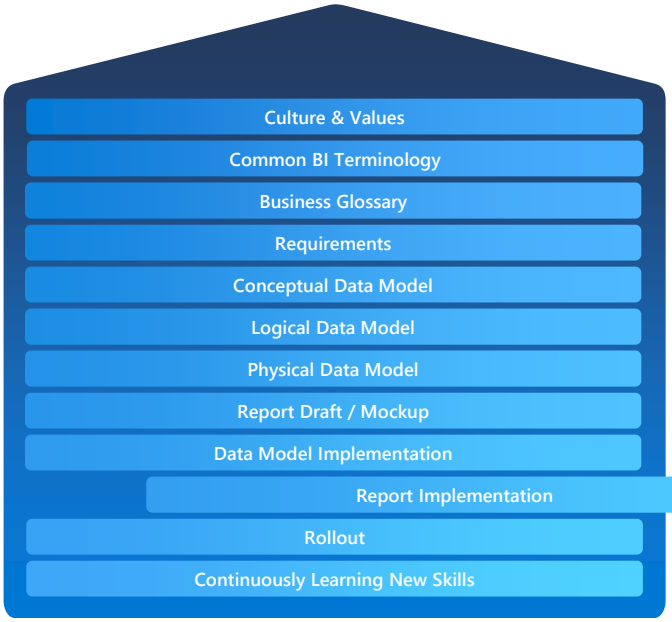
- **Stick to your logical and physical data models**
- **Transform data** as needed
- Follow Roche's Maxim: Implement things as upstream as reasonable
- Avoid redundant data
- Avoid unused or duplicate columns
- Avoid Snowflake data structure
- Avoid direct relationships between fact tables
- Avoid many-to-many relationships
- Avoid DAX tables and columns

This is the first step that must be done in Power BI.

Data model implementation includes DAX measures implementation.

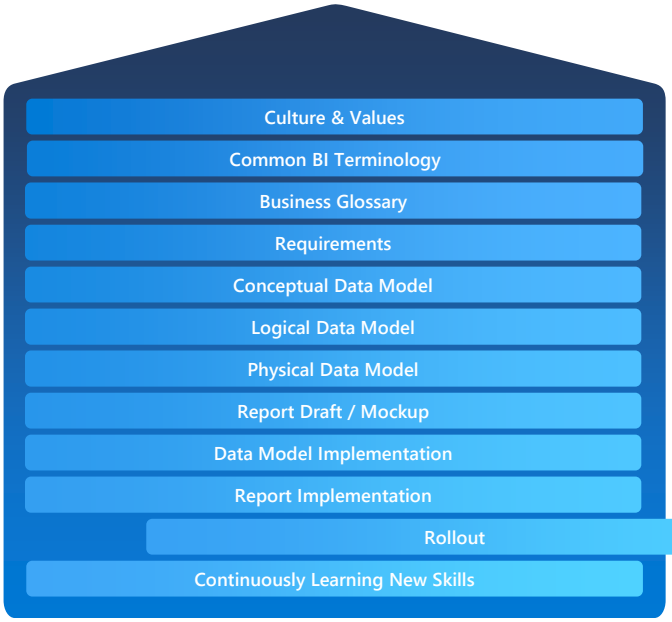
Follow Roche's Maxim: Implement things as upstream as reasonable, e.g., don't fix things on measures or with weird relationships that you can fix in Power Query

Roche's Maxim on YouTube: Roche's Maxim of Data Transformation (youtube.com)
https://www.youtube.com/watch?v=n_8U-9eblXM



- Stick to Power BI standards as often as possible, e.g., use filter pane
- Implement for performance, e.g., reduce number of visuals or avoid scrollable one-page reports
- Implement for maintainability, e.g., chose calculation groups over bookmark navigation
- Avoid unwanted interactions, e.g., don't offer filter options that shouldn't be used, turn off cross-filtering that doesn't work anyway, etc.

- Do reviews.
- Use guideline, e.g.,
  - Power BI Checklists — DATA GOBLINS (data-goblins.com) https://data-goblins.com/checklists
  - Power BI Dataset Checklist — DATA GOBLINS (data-goblins.com) https://data-goblins.com/power-bi/dataset-checklist



- User acceptance test
- User training
- User documentation
- Usage Monitoring
- Monitoring for failure
- Hand over to support
- Hypercare
- Provide contact for errors and changes

Hypercare: Increased user support resources and awareness during the first weeks after rollout.

- Power BI is continuously improving
- Keep your knowledge up to date
- Community of practice
- Encourage user adoption of core Power BI functionality
- Apply new features, if useful

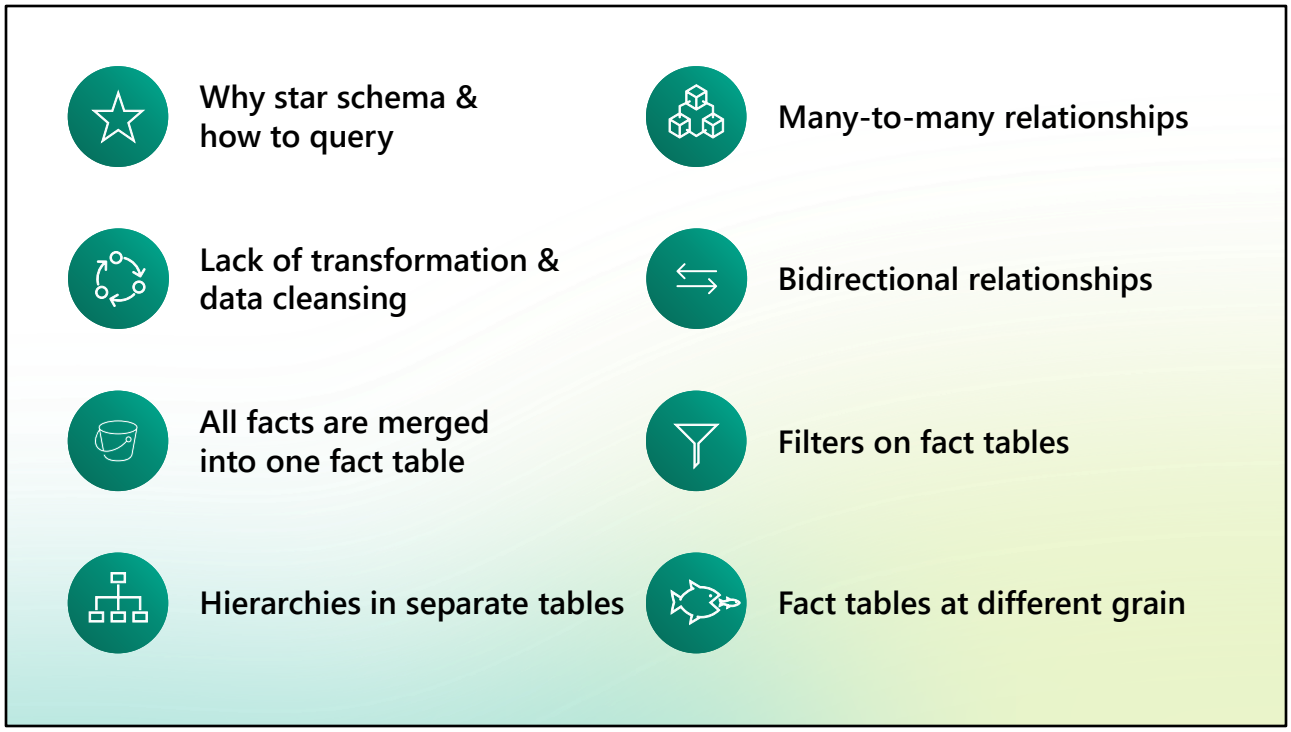
Learning = Consuming content (listen, read, watch) + Practicing + Feedback / Reviews

Core Power BI functionality = Filter pane, reset filters, user bookmarks, apps, standard visuals, focus mode, dashboards, drill-through including cross report drill-through, create reports from shared semantic models, analyze in Excel, featured tables, embed in PowerPoint, etc.

Before starting with the implementation in Power BI, some conceptual work needs to done to get a good, simple and working Power BI solution.

How can you enable the transfer of good conceptual work to a successful Power BI solution that implements the requirements and concepts as designed?

Patterns for common challenges



Typical, frequently seen anti-patterns:

Why star schema & how to query

Artifacts:

- Unnecessary, non-compliant relationships
- Inefficient DAX code

Solution:

- Learn: Read books, attend classes

How to query star schema effectively and efficiently is not part of this session. See Kimball book on slide "Conceptual Data Model".

Lack of transformation

Some people seem to be afraid of transforming the source tables or adding useful additional columns, like surrogate keys or time intelligence support columns in the date dimension.

Artifacts:

- Unnecessarily complex data models, e.g., snowflake pattern
- Inappropriate fact table structure, e.g., not transforming balances into transactional data
- Inefficient DAX code from missing support columns
- Many-to-many relationships

Solution:

- Create logical data model based on requirements before implementing anything in Power BI.
- Transform data to match with star schema pattern and reporting needs.
- Cleanse data before loading.

All facts are merged into one fact table

Artifacts:

- Any facts at any grain, in any structure – e.g., transactional, snapshot, cumulating – is combined in one fact table.

Solution:

- A fact table represents a process and data describes events in this process.
- It's totally fine to have multiple fact tables in one data model.
- Create separate fact tables for different processes.
- Create separate fact tables at different grain.
- Create separate fact tables with different structure, e.g., transactional and snapshot.

In simple reports it's OK to have only one fact table, of course, if data is appropriately simple.

Hierarchies in separate tables

Artifacts:

- Hierarchy tables attached to dimension tables in snowflake pattern

Solution:

- Merge tables: Hierarchies are part of a dimension table

Many-to-many relationships

Many-to-many relationships are not wrong in general, but they are used way more often than appropriate.

Artifacts:

- Users select more data than expected, e.g., by selecting a product in a filter, data of the whole product group is included in the calculation, resulting in unexpected numbers.
- Many-to-many relationships between dimension tables and fact tables (1)
- Many-to-many relationships between different fact tables (2)
- Many-to-many relationships between dimension tables (3)

Solution:

- (1) Use proper grain (shrunken) dimension tables and use one-to-many relationships.
- (1) With low cardinality dimension fields, add all existing combinations to dimension table and use one-to-many.
- (1) With high cardinality dimension fields, split dimension table and relate each to the fact table using one-to-many – do not create direct relationships between the dimension tables.
- (2) Use conformed dimension tables instead, i.e., dimension tables that relate to both of the fact tables.
- (3) Consider relating each of the dimension tables directly to the fact tables, instead of directly relating dimension tables..

Keep in mind that the fact table is a relation between the related dimension tables.

Bidirectional relationships

Artifacts:

- Slow calculations
- Unexpected results

Solution:

- Activate bidirectional filtering in measures using CROSSFILTER only if needed.
- Put a visual-level measure-based filter on slicers if you want to reduce the options by other filter's selection.

In some row-level security scenarios, bi-directional relationships are necessary.

Filters on fact tables

Artifacts:

- Ambiguous field selection: Report creators don't know which field to use, e.g., which date.

- Filters and cross-filtering not working on all visuals.

Solution:

- Hide fields in fact tables.
- Turn degenerate dimensions into dimension tables (technology-specific recommendation for Power BI).
- In DAX code, apply filter expressions to dimension tables.
- In visuals only use fields from dimension tables and measures.

Some measures are more efficient using filters on fact table columns, especially date related or using flag columns. In any case, consider filters on dimension tables first. In visuals, aim to avoid filters on fact table columns.

Exception: Your data model has only one table.

Fact tables at different grain

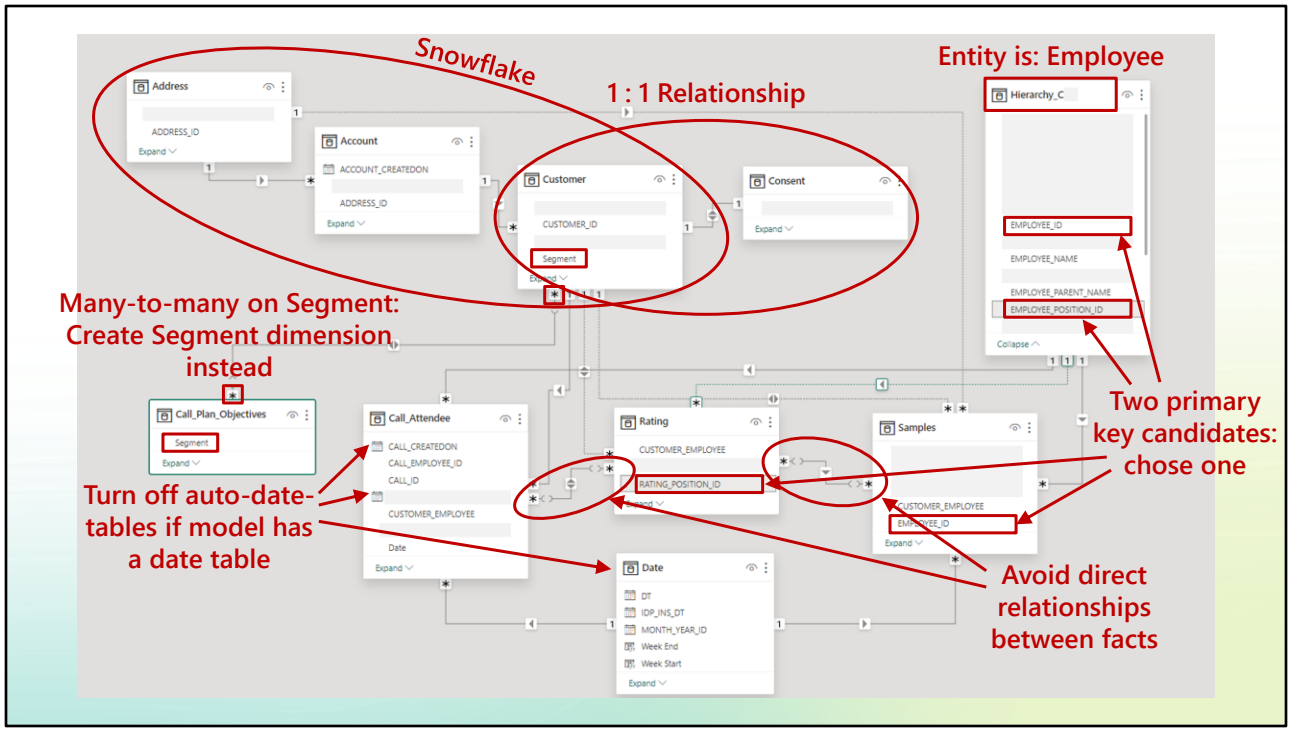
Fact tables at different grain exist and are OK in a data model. Nevertheless, consider avoiding them in the first place by transforming source tables into the same level of grain for each common dimension. This makes the data model and the measures much simpler.

Artifacts:

- Many-to-many relationships
- Incorrect numbers
- Filters not working as expected

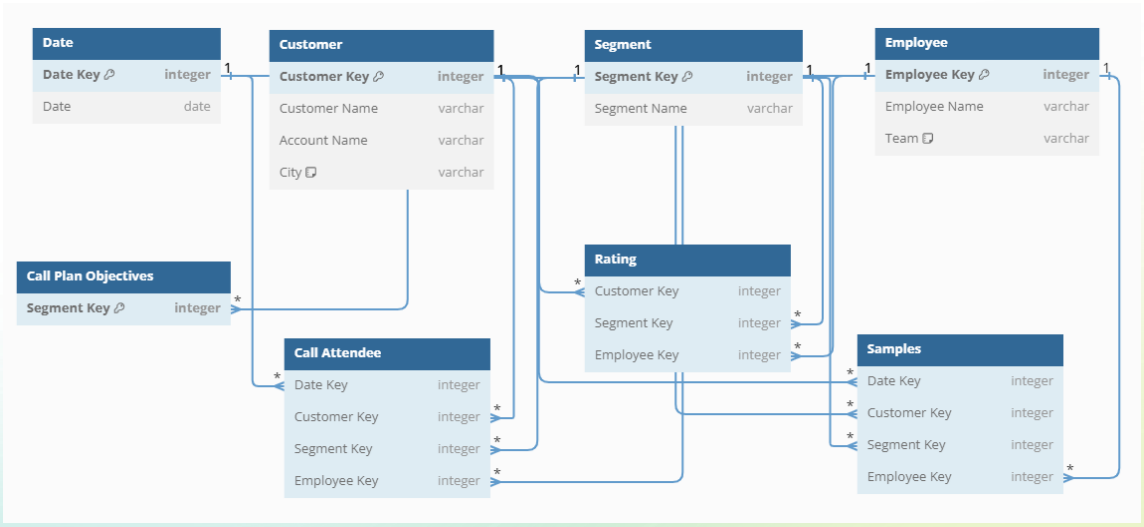
Solution:

- Use conformed, shrunken dimensions.
- Add surrogate keys to fact tables on common grain level.
- Aggregate or allocated fact data before loading.



Findings

- 1 : 1 Relationship in dimensions
- Snowflake structure in dimensions
- Many-to-many relationship between dimension and facts
- Entity named "Hierarchy"
- Fact-to-fact relationship
- Auto-data-table
- More than one primary key for the same record



Simplified Model

This model allows to answer all the same business questions and calculate the same measures as the previous one.

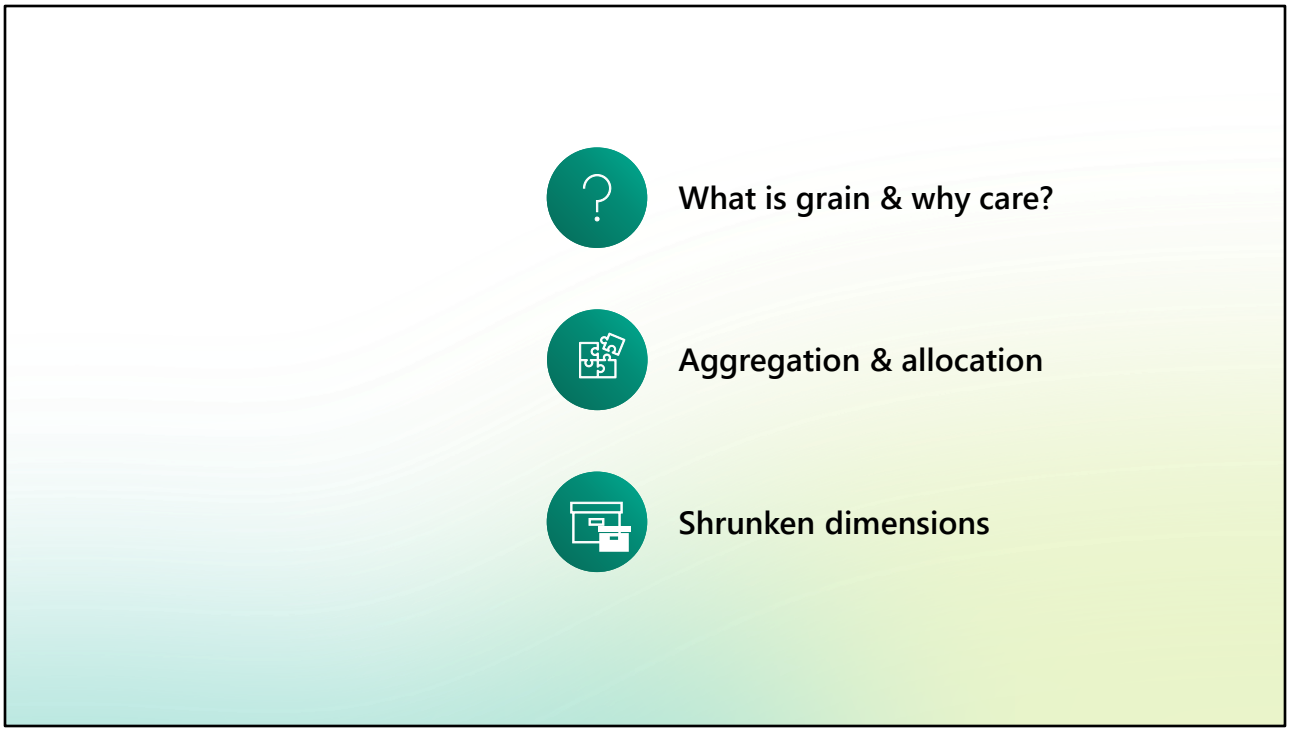
Segment is a shrunken dimension of Customer.

City represents address attributes.

Team represents employee hierarchy levels.

Fact tables at different grain

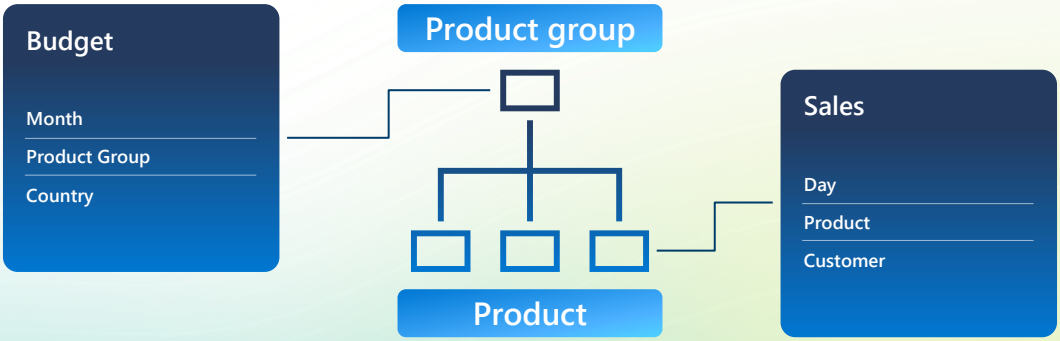
Facts at different grain



Facts at different grain

What is Grain & Why Care?

"Grain" defines the level of detail of each fact table,
e.g., product vs. product group, date vs. month.



Grain becomes relevant when comparing data that is available at different level.

In literature you will also find the term „granularity" instead of „grain".

Aggregation & Allocation

Aggregation = Summarize values by lower grain
Allocation = Distribute values to lower grain

Low grain = few large aggregates

High grain = many small details

Allocation requires defined allocation rules.

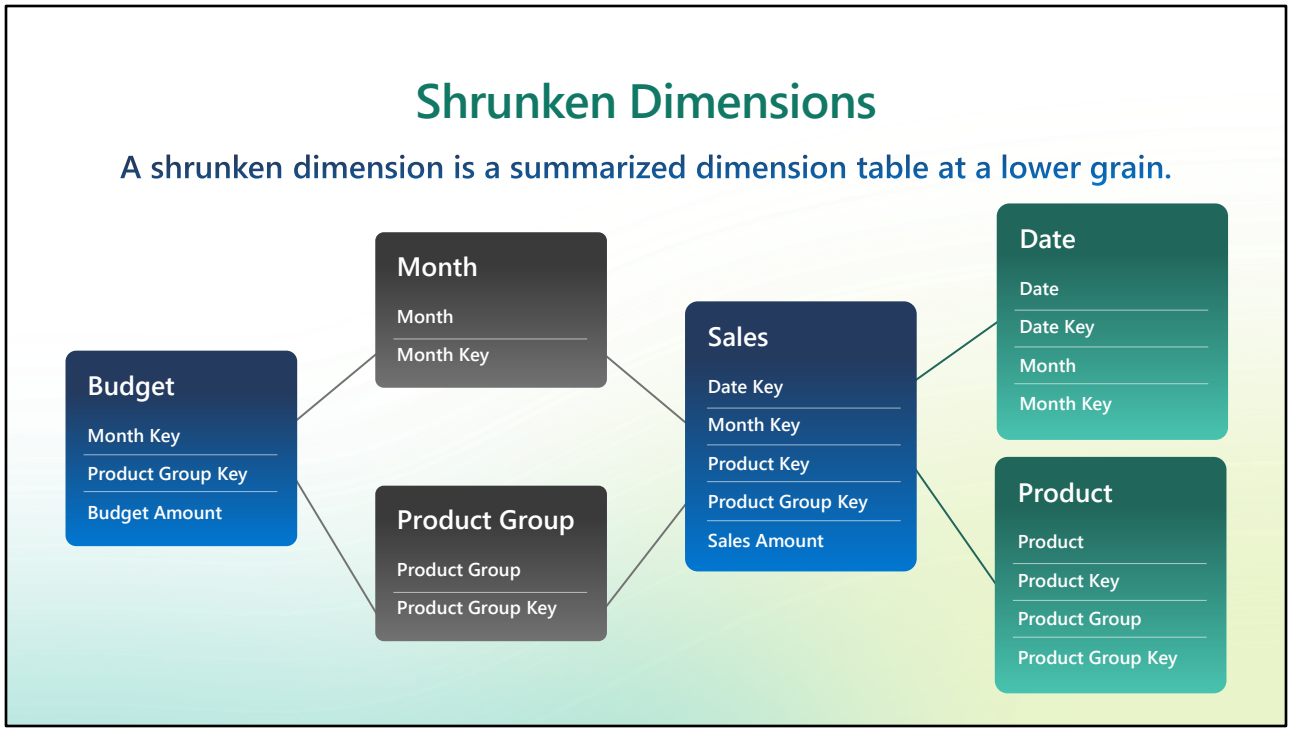
Allocation artificially splits values into small values for a higher grain category.

Allocation requires allocation rules – they don't come naturally. Align with business on allocation rules, e.g., even distribution, proportional distribution, seasonal distribution, no allocation to weekends, etc.

An **example of an allocation rule** could be: Distribute the monthly budget evenly across the working days of the month.

Make sure you are aligned on the wording within your organization or project:

- Low grain = Low level of detail = High level overview= Bird's eye view
- High grain = High level of detail = Low level from manager's perspective = operational view



Data model example: Star schema with fact tables at different grain

Month: Shrunken dimension of Date

Product Group: Shrunken dimension of Product

Often the lower grain keys need to be added to the higher grain fact tables. If the lower grain key is part of the higher grain dimension table and not only part of the lower grain dimension table, then this makes merging the lower grain keys on the higher grain fact tables easy.

The grain of each fact table is documented in the bus matrix. Enterprise Data Warehouse Bus Architecture - Kimball Group *https://www.kimballgroup.com/data-warehouse-business-intelligence-resources/kimball-techniques/kimball-data-warehouse-bus-architecture/*

Aggregation & Allocation – Implementation Options

Allocate in DAX measure custom code typically requires use of virtual tables.

Martin Bubenheimer
Power BI Architect

martin.bubenheimer@gmx.de

Nuremberg
Germany

Martin_D Microsoft Power BI Community Superuser

Top Data Warehousing Voice

QR code is download link for this slide deck:
https://raw.githubusercontent.com/MartinBubenheimer/powerbi-training/main/Community%20of%20Practice/Process%20House/Power%20BI%20Process%20House.pdf