

# Designing a Data Warehouse on the Microsoft SQL Server Platform

---

UNDERSTANDING THE FUNDAMENTAL CONCEPTS OF  
A DATA WAREHOUSE



**Ana Voicu**

@ana\_voicu



# Overview



## Goals and purpose of a data warehouse

## Introducing dimensional modeling

- What are facts and fact tables?
- What are dimensions and dimension tables?
- Star schemas

## Putting it all together

- The 4-step dimensional design process

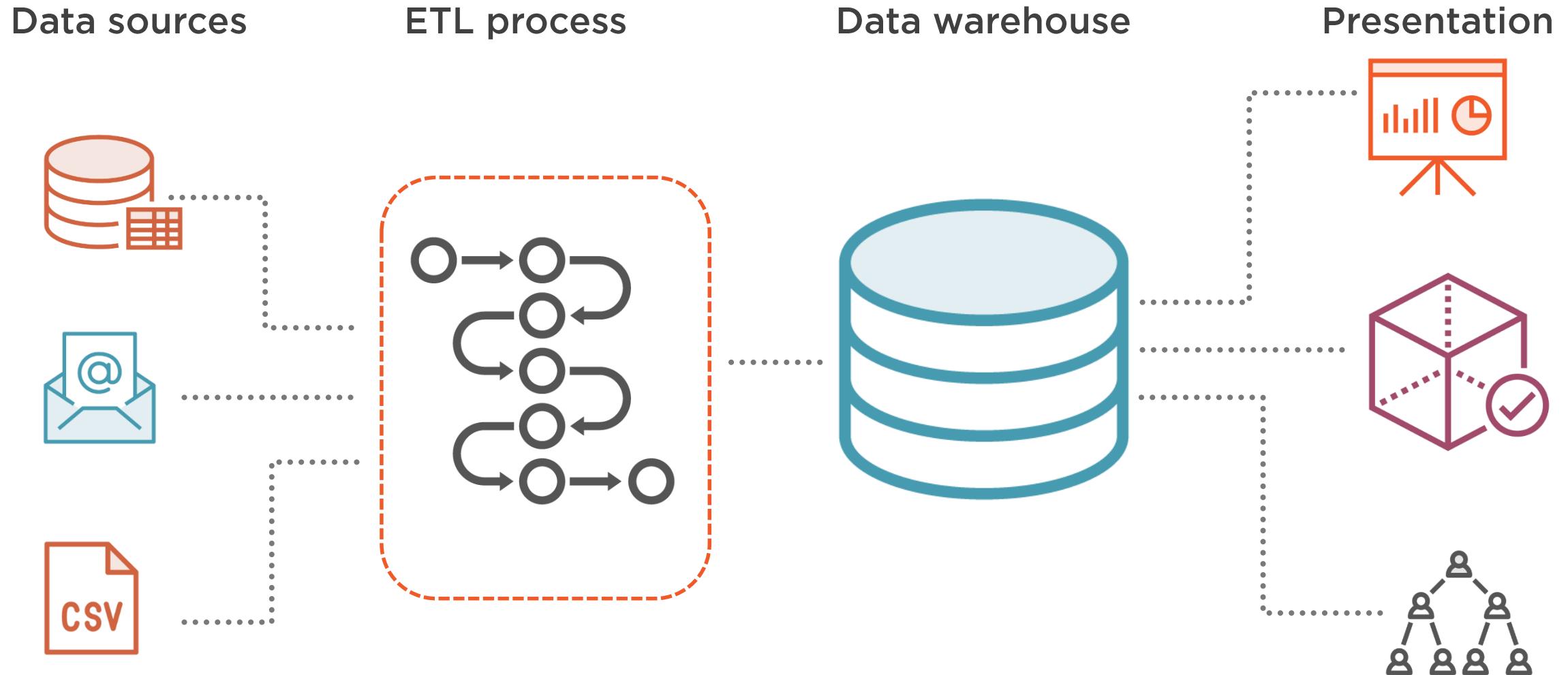


# Goals of a Data Warehouse Solution

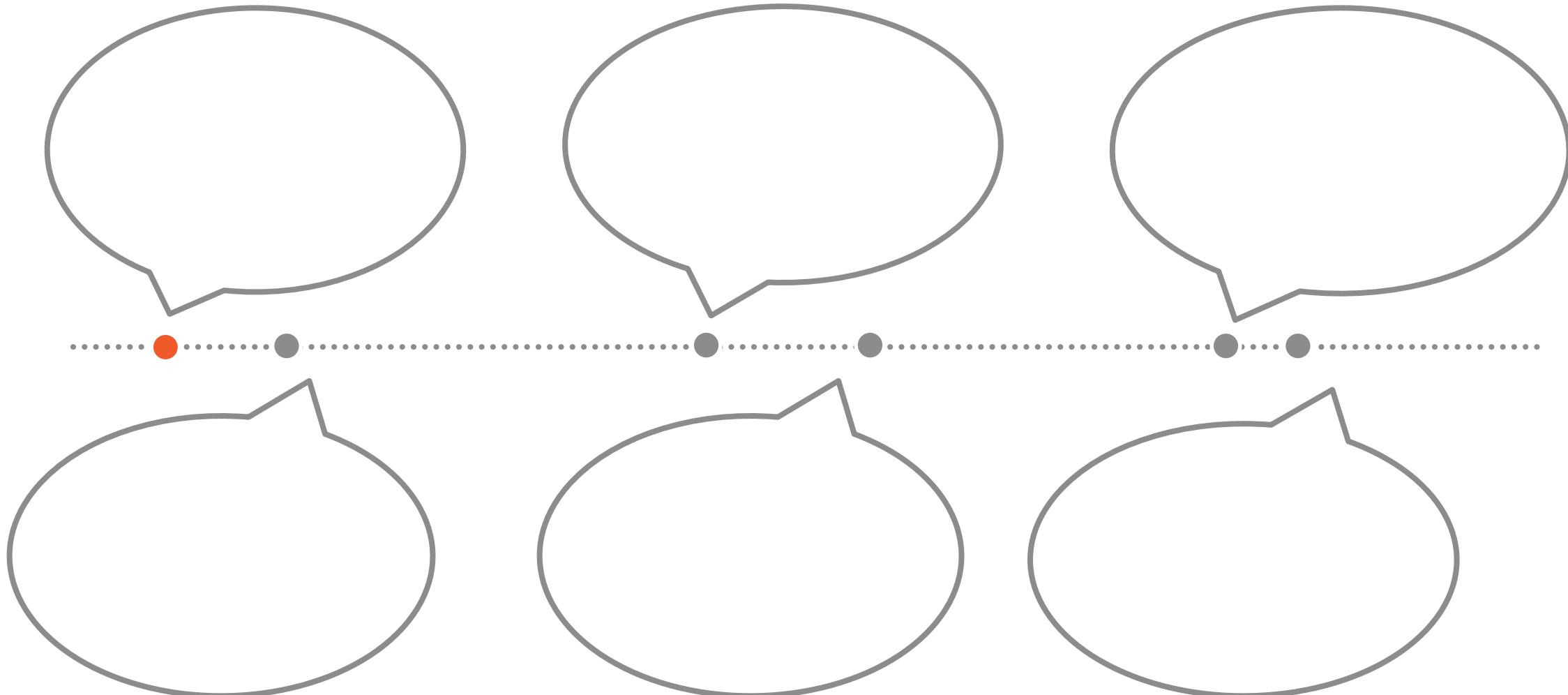
---



# What Is a Data Warehouse?



# Problems a Data Warehouse Can Solve



# Requirements of a Data Warehouse Solution

Easily accessible

Fast

Consistent

Flexible

Secure

Foundation for  
decision-making



# Responsibilities of a Data Warehouse Designer

Understand the  
business users

Deliver high-quality,  
relevant, and  
accessible  
information

Sustain the DW  
environment



# Introduction to Dimensional Modeling

---



# Dimensional Modeling



**Database design method optimized for data warehouse solutions**

**Popular technique because it addresses two important requirements:**

- Deliver data in an understandable format
- Deliver fast query performance

**Key word is “simplicity”**



# Elements of a Dimensional Model



Facts (the measurements/metrics or facts from your business process)



Dimensions (for providing the context of a business process event)



Attributes (the various characteristics of a dimension)



Star schema (and/or OLAP cubes)



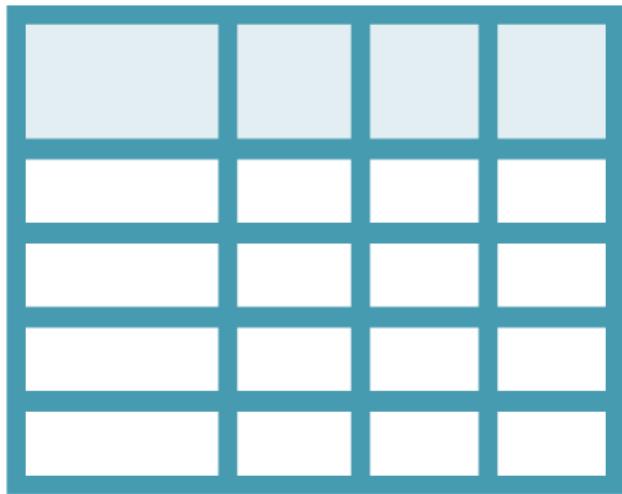
“We sell ice-cream and other products in various locations and measure our achievements over time.”

**Jane Poppins**

*Happy Scoopers CEO*



# Fact Tables and Facts



**Fact table** = table that stores the performance measurements resulting from an organization's business process events



**Fact = a business measure**

- Sales
- Profit
- Volume
- Number of transactions



# Fact Tables and Facts



**Facts answer questions like:**

- “What are we doing?” (sell, buy, count)
- “What do we want to achieve?” (more sales, bigger profit)

**1 row in the fact table is 1 measurement in real life**

**Fact columns in a fact table should be additive**

**Facts make sense in combination with dimensions**

- Linked with foreign keys
- Date/Time dimension is present in most data warehouses



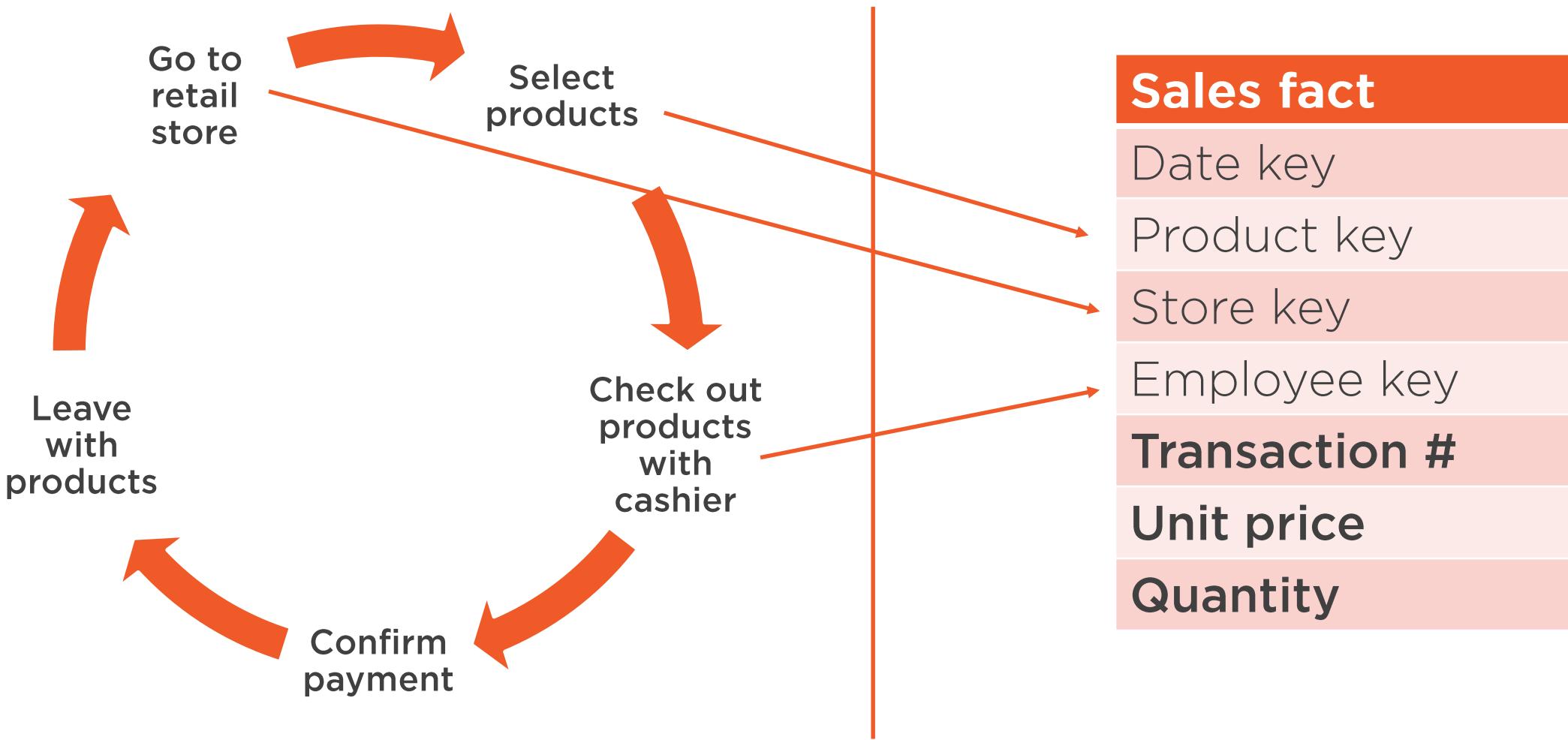
# Example of a Fact Table



Sales fact
Date key
Product key
Store key
Employee key
Transaction #
Unit price
Quantity



# Example of a Fact Table



# Example of a Fact Table



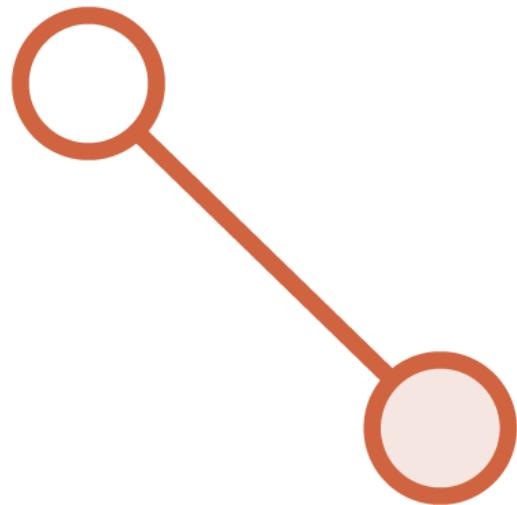
## Sales Fact

Date key
Product key
Store key
Employee key
<b>Transaction #</b>
Unit price
Quantity

C  
O  
M  
P  
O  
S  
I  
T  
E  
K  
E  
Y



# Characteristics of Fact Tables



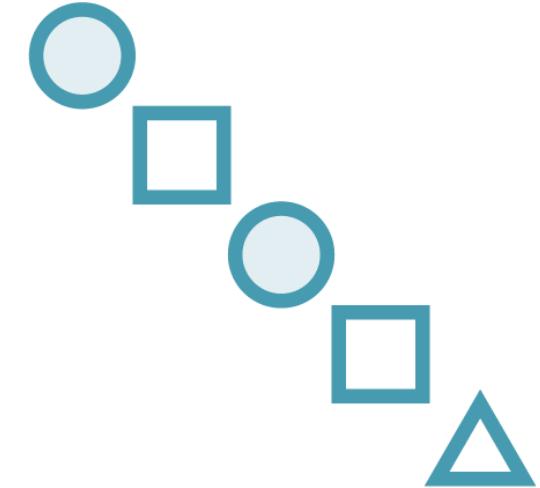
1:1 relationship  
between fact table  
row and real-world  
event



Most facts  
should be  
additive



Foreign keys to  
dimension  
tables



Composite key  
as PK for fact  
table



# What Are Dimensions?



**Companions to a fact table**

**Textual context associated with a business process measurement event**



# Questions Answered by Dimension Tables



Who



What



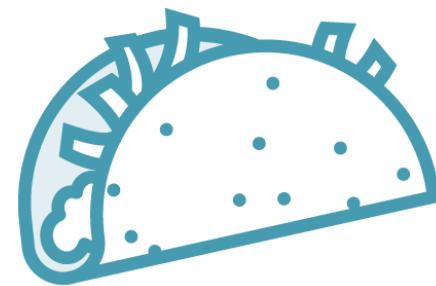
Where



When



How



Why



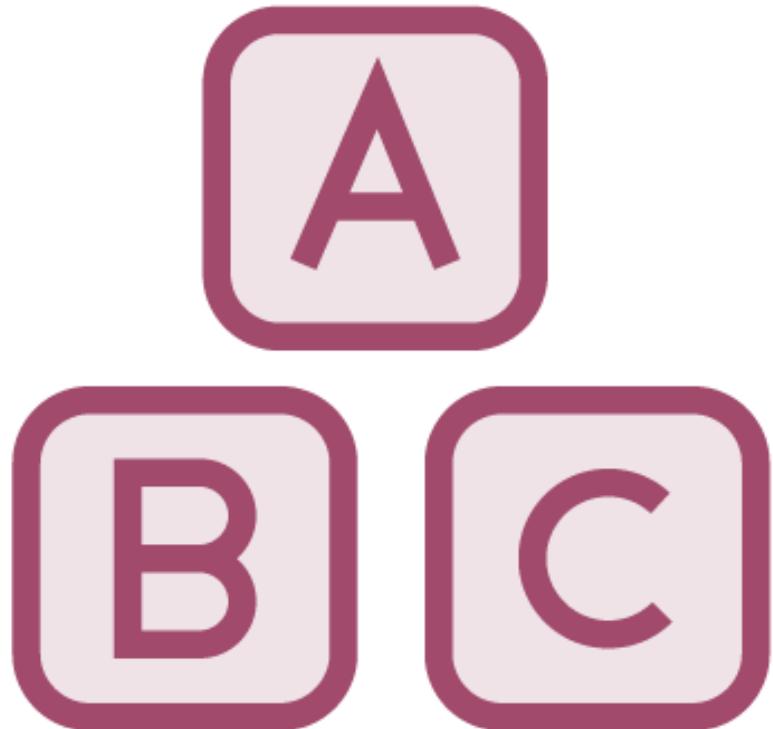
# Example of a Dimension Table



Product Dimension
Product key
Product name
Brand name
Category name
Subcategory name
Package type
Package size
Weight
Weight unit of measure



# Characteristics of Dimension Tables



**No limit for the number of attributes in a dimension table**

- Common to have tables with 50 to 100 attributes
- Some dimension tables have only a handful of attributes

**Have fewer rows than fact tables**

- But can be much wider

**Defined by a single primary key**

- Basis for the referential integrity with the fact table

**Denormalized**

- Flattened many-to-one relationships within a single dimension table



# Dimension Attributes

## The primary source of

- Query constraints
- Groupings
- Report labels

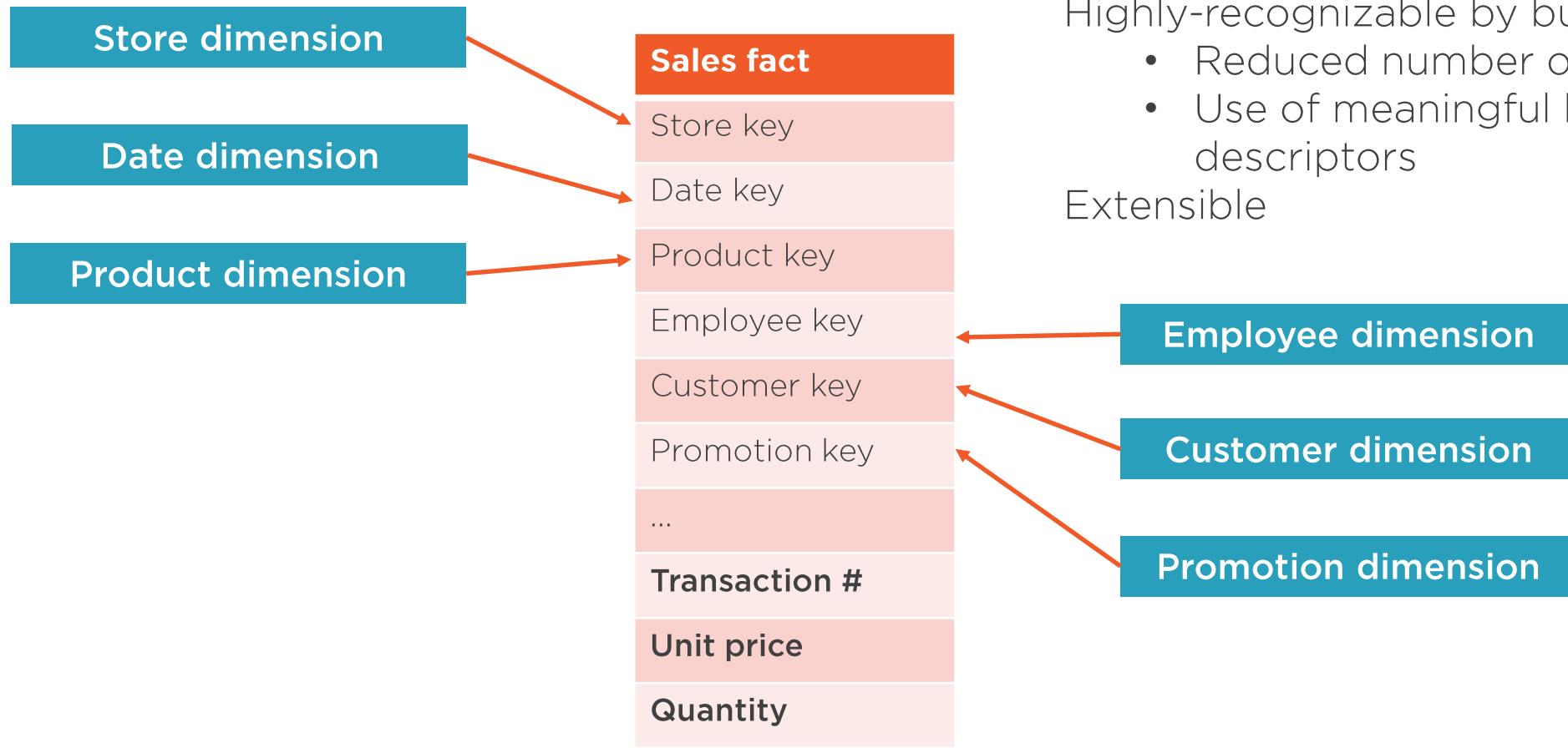
## Quality of attributes $\propto$ quality of the system

- Use real words vs. cryptic abbreviations
- Minimize codes in the dimension tables

Product Key	Name	Category name	Subcategory name
1	Beery cotton candy	Candy	French candy
2	Cotton candy	Candy	French candy
3	Peppermint candy (seasonal)	Candy	Fudge
4	Green tea ice cream	Ice-cream	Reduced fat
5	Chocolate chip cookie dough ice cream	Ice-cream	Fat-free frozen dairy
6	Neapolitan ice cream	Ice-cream	Lactose-free
7	Cantuccini	Cookie	Biscotti
8	Chocolate mint cookie	Cookie	Retro snacks
9	Lemon cookie	Cookie	Fruity cookies



# Dimensional Model



## Characteristics:

Simple

Symmetric

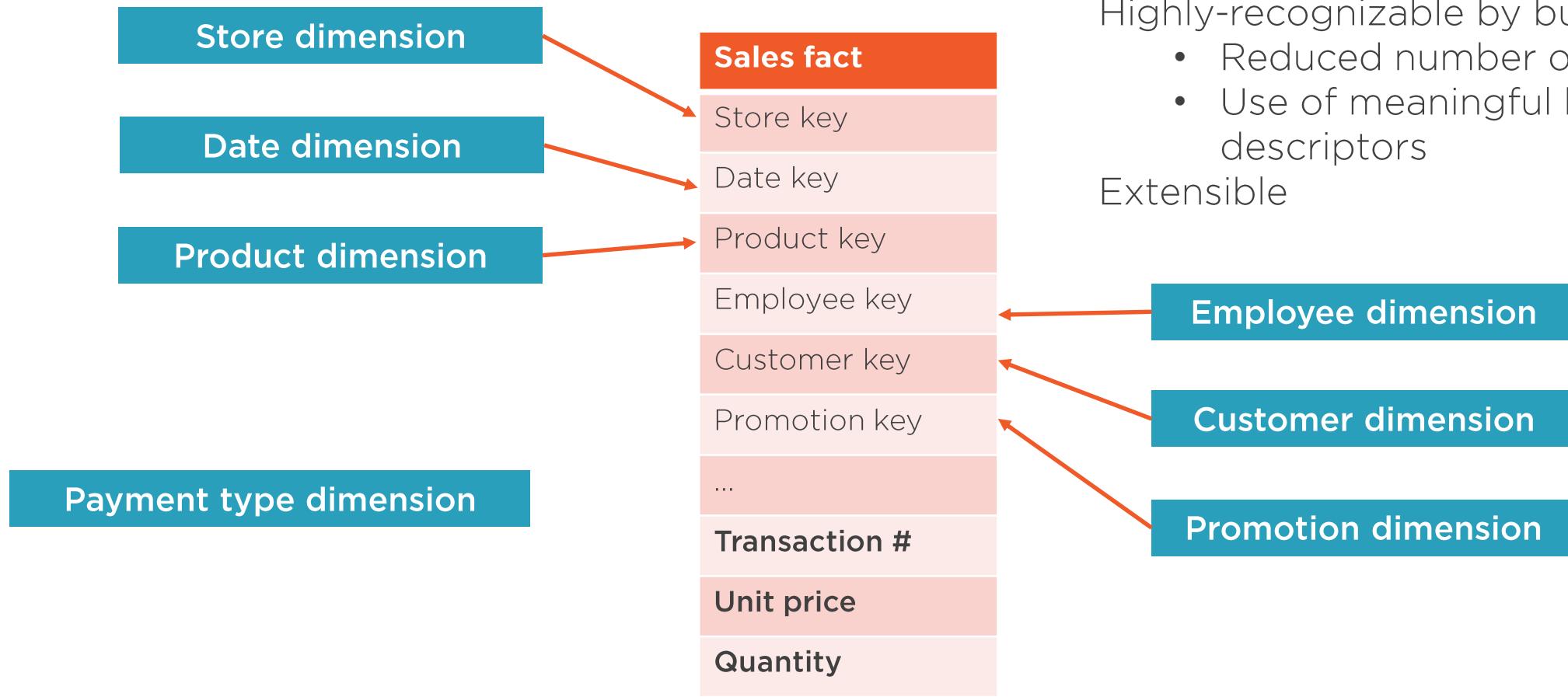
Highly-recognizable by business users

- Reduced number of tables
- Use of meaningful business descriptors

Extensible



# Dimensional Model



## Characteristics:

Simple

Symmetric

Highly-recognizable by business users

- Reduced number of tables
- Use of meaningful business descriptors

Extensible

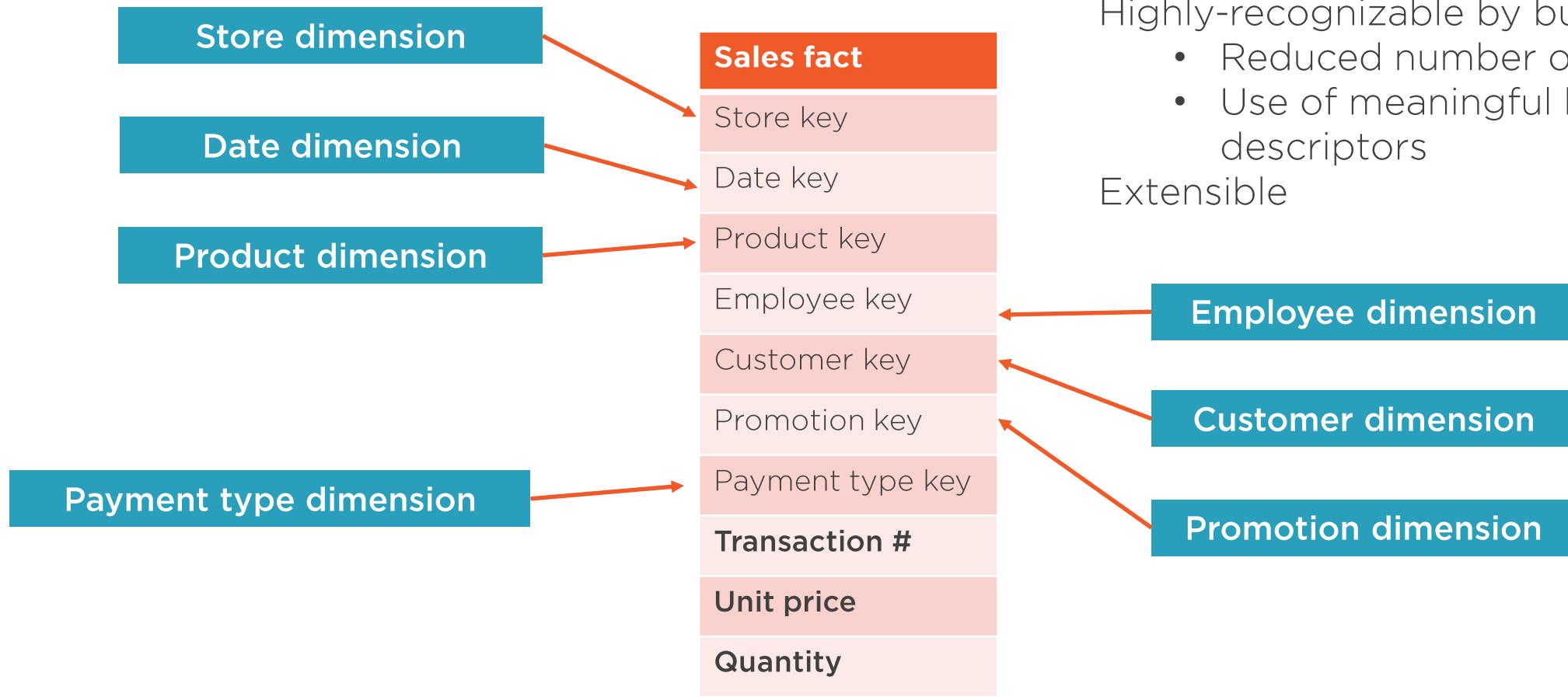
Employee dimension

Customer dimension

Promotion dimension



# Dimensional Model



## Characteristics:

Simple

Symmetric

Highly-recognizable by business users

- Reduced number of tables
- Use of meaningful business descriptors

Extensible

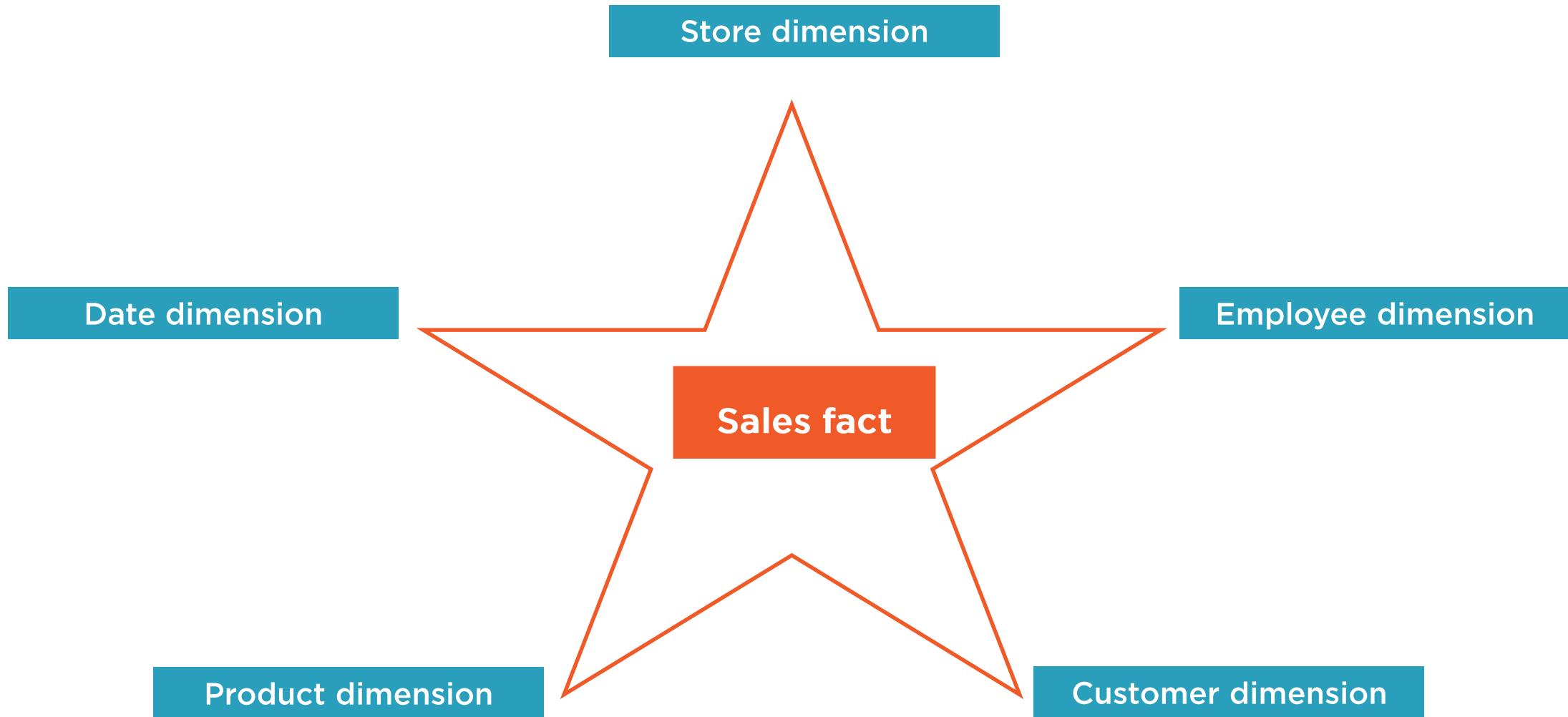
**Employee dimension**

**Customer dimension**

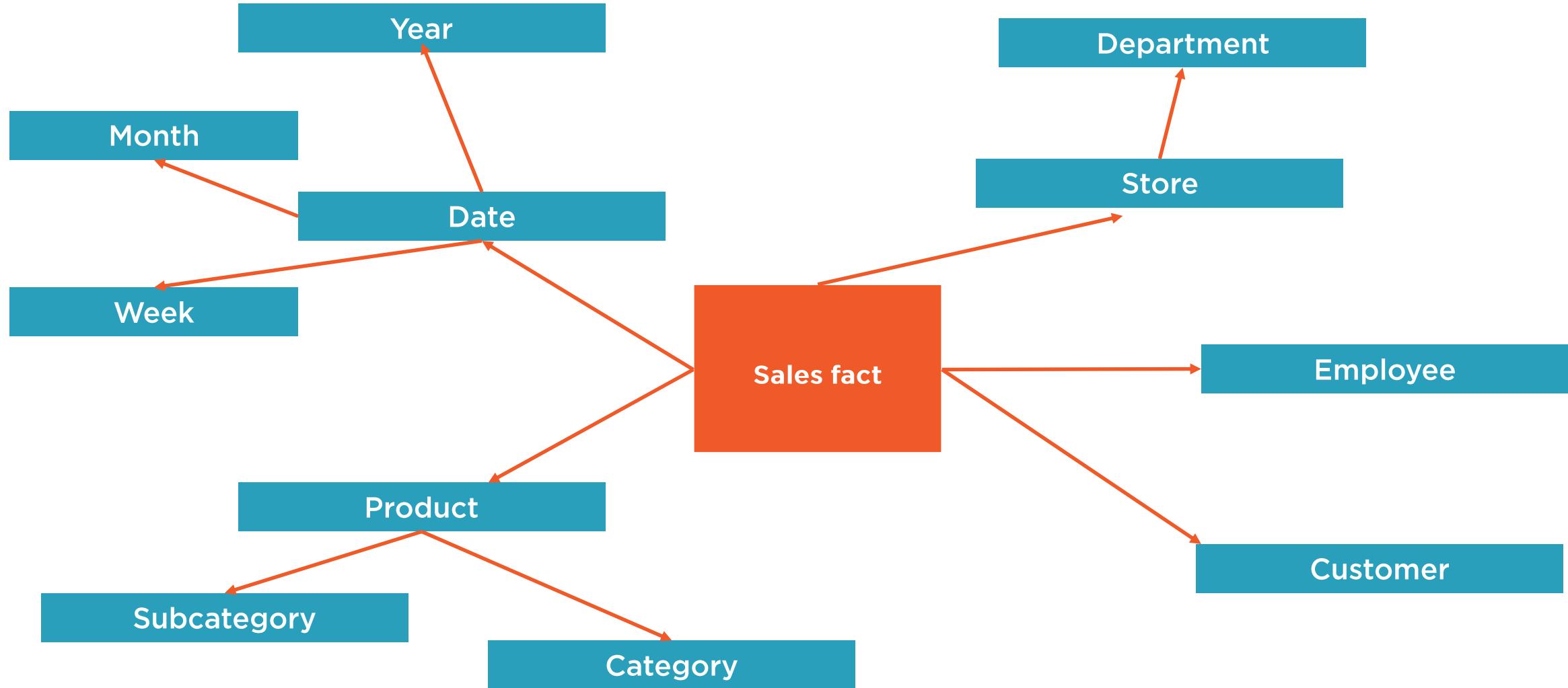
**Promotion dimension**



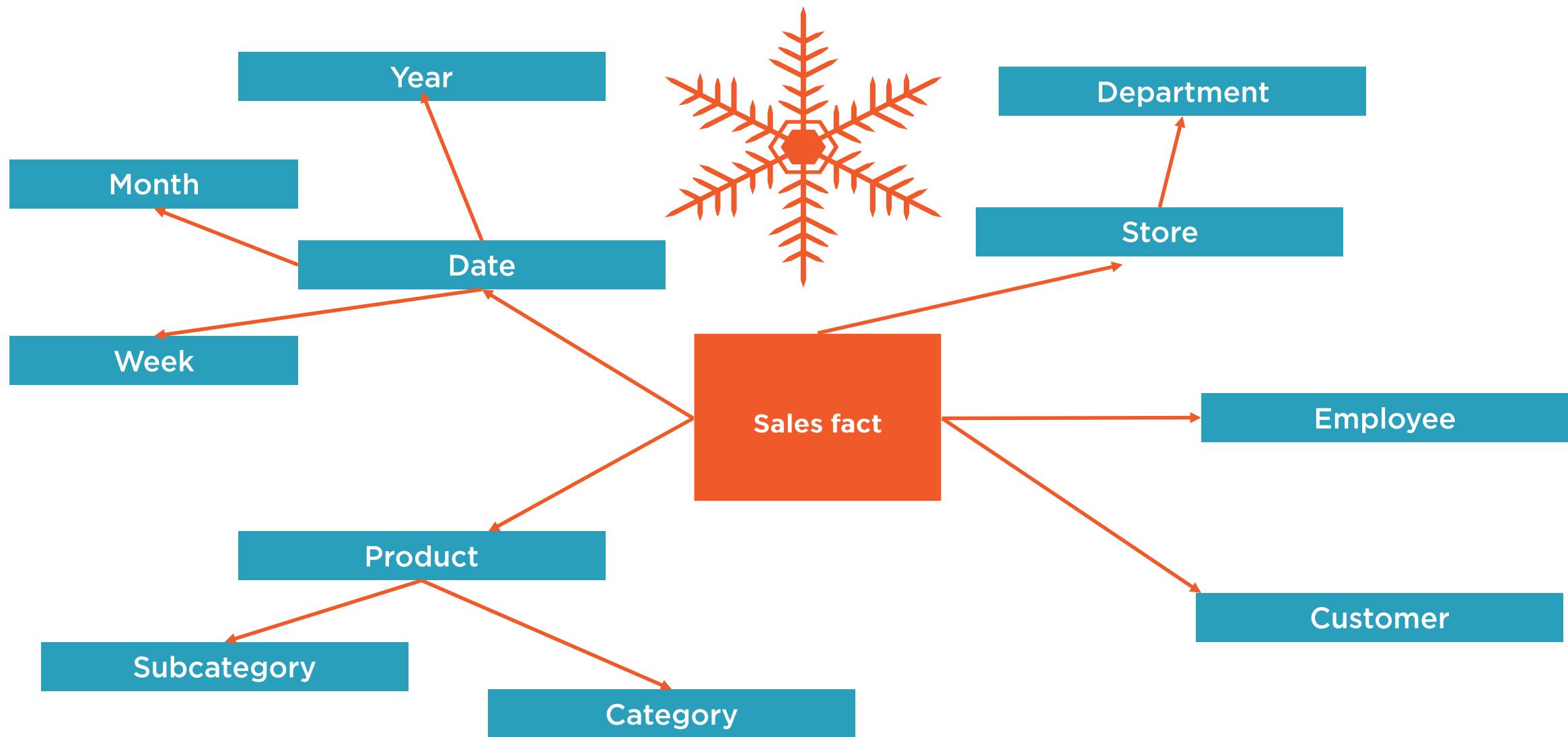
# Dimensional Model as a Star Schema



# Dimensional Model as a Snowflake



# Dimensional Model as a Snowflake



# The Four-step Dimensional Design Process

---

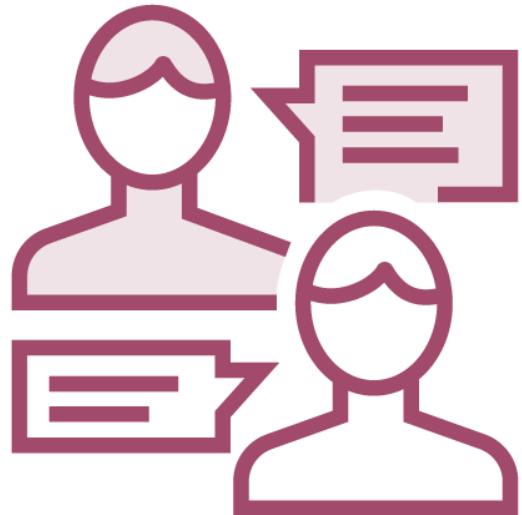


# Dimensional Design

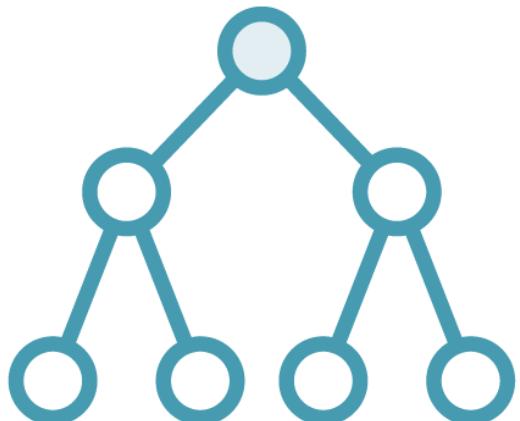


- Is done “with pen and paper”**
- Focuses on understanding the deliverables of the project**
- Consists of four steps**

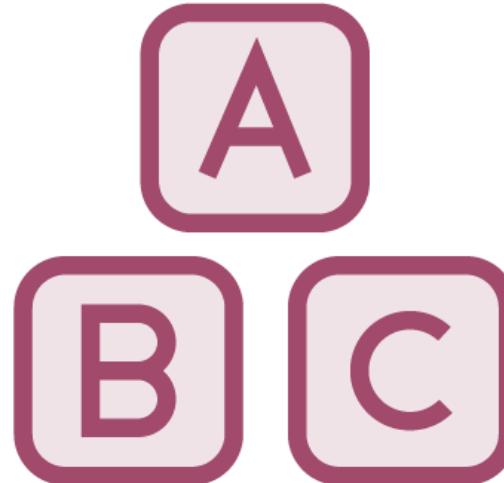
# Steps of Dimensional Design



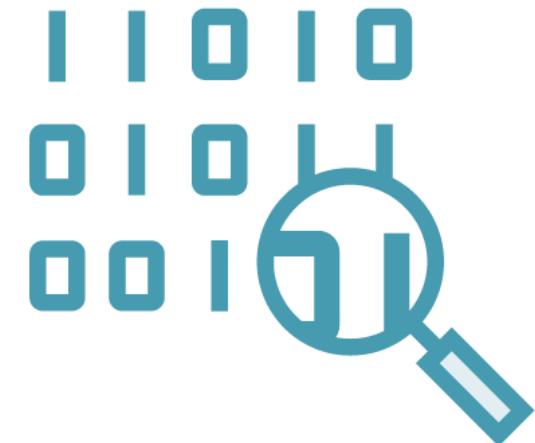
Select the  
business  
process



Declare the  
grain



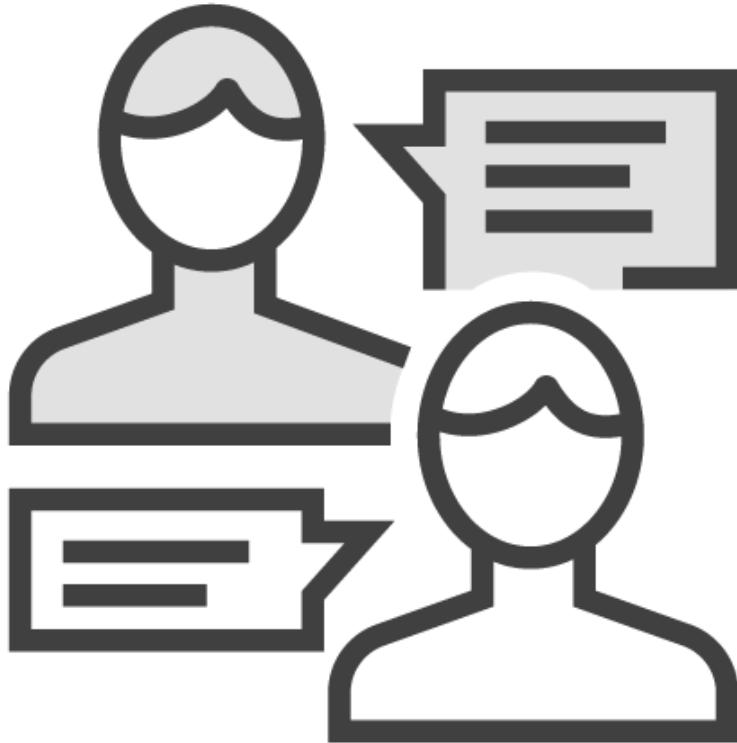
Identify the  
dimensions



Identify the  
facts



# Step 1: Select the Business Process



**Low-level activities performed by an organization**

**Are identified by listening carefully to the business users**

**Characteristics:**

- Expressed as verbs
- Are supported by an operational system
- Generate KPIs



# Step 2: Identify the Grain



**Specify the detail level of a business process we want to measure**

Example	Grain	Questions to ask
Sales/day	One row at the end of the day	How much was payed by each customer? Is this important information?
Sales/day/order	One row for every finished order	What was the most sold product? Is this important information?
Sales/day/order/product	One row for every product sold within an order	Is this enough information?



## Step 2: Identify the Grain



**Specify the detail level of a business process we want to measure**

**“How do you describe a single row in the fact table?”**

**Grain declarations are expressed in business terms**



# Steps 3 and 4

## Identify the dimensions

“How do business users describe the data resulting from the process?”

“who, what, where, when, why, how”

Examples:

- Date
- Product
- Customer
- Employee

## Identify the facts

“What is the process measuring?”

All candidate facts must be true to the grain for that fact table

Facts with different grains are split in separate tables

Examples:

- Sales price
- Sales quantity (or Units sold)



Both business requirements and  
the realities of the source data  
should be considered when  
designing the dimensional model



# Dimension Table Design Techniques

---



**Ana Voicu**  
@ana\_voicu



# Dimension Table Techniques



**What is specific to a dimension table?**

**Technical guidelines for designing a table**

- Working with keys
  - Surrogate key
  - Business key
- Populating the dimension's attributes with data

**Examples of common dimensions**

- Date dimension
- Product dimension



# What Is Specific to a Dimension Table?

---



# Example: Sources for the Product Dimension

Administration system

Inventory changes

- Normalized model
- Products, departments, categories, subcategories

Software product for storing recipes

Used by the cook for daily recipes

POS used by staff

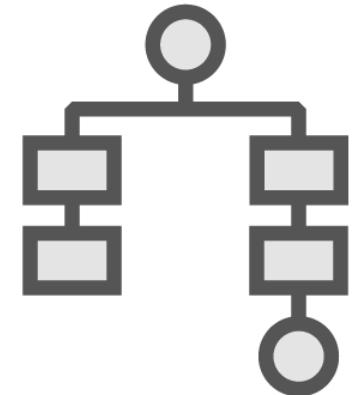
Take and fulfill orders

Flat file with information from an old accounting system

The system is not in use anymore

- But kept for historic information

Product dimension



# Requirements of a Dimension Table



**How will you build a table meeting these requirements?**

- Dimension table can be linked to fact tables
- Keep track of the source system for data
- Data is consistent and readable
- Keep history of data changes from the operational system



# What Is Specific to a Dimension Table?



**A dimension table is different than a relational table**

- The normalization rules don't apply
- Not optimized for data modification aspects
- Integrates in a consistent way multiple source data
- Data is easy to read by the user



# Surrogate Keys

---



# Relationship between Fact and Dimension

## One-to-many relationship

Dim Product	Fact Sales
PK	PK
Attribute1	FK product
Attribute2	FK date
Attribute3	...
Attribute4	Fact1
Attribute5	Fact2
...	...

## Relationship components

- 1:M example
  - 1 product (chocolate chip cookie)
  - 2 sales (transactions)
- Keys involved
  - Primary key of dimension table
  - Foreign key from fact table
- It's important to create it correctly
  - Plays an important role in the BI solution
  - Should be unique across all data sources for the dimension



# Creating the Primary Key

## Bad practice

Use the PK from the source systems

Can lead to overlapping rows

Source system 1

Id	Name	...
3	Chocolate chip cookie	...

Source system 2

Id	Name	...
3	Potato	...

## Good practice

Generate an artificial key

- Simple integer column
- Automatically incremented
- Doesn't require careful maintenance

In DW terminology, this is the surrogate key



# Synonyms for Surrogate Key



**Meaningless key**

**Integer key**

**Non-natural key**

**Artificial key**

**Synthetic key**



# Advantages of Using a Surrogate Key

---



# Advantages of Using a Surrogate Key



**Integrate multiple source systems**

# Advantages of Using a Surrogate Key



## **Integrate multiple source systems**

- Example: gathering data for the Product dimension in a restaurant or store



# Integrate Data from Multiple Sources



# Integrate Data from Multiple Sources

Source system 1

ID	Name	Description	UM	Color
123	Milk	Non-fat	L	White
776	Sugar	Brown sugar	KG	Brown



# Integrate Data from Multiple Sources

Source system 1

ID	Name	Description	UM	Color
123	Milk	Non-fat	L	White
776	Sugar	Brown sugar	KG	Brown

Source system 2

ID	Name	Description	UM	Color
776	Butter	Non-fat	G	Yellow
778	Sugar	Normal sugar	KG	White



# Integrate Data from Multiple Sources

Source system 1

ID	Name	Description	UM	Color
123	Milk	Non-fat	L	White
776	Sugar	Brown sugar	KG	Brown

Source system 2

ID	Name	Description	UM	Color
776	Butter	Non-fat	G	Yellow
778	Sugar	Normal sugar	KG	White



# Integrate Data from Multiple Sources

Source system 1

ID	Name	Description	UM	Color
123	Milk	Non-fat	L	White
776	Sugar	Brown sugar	KG	Brown

Source system 2

ID	Name	Description	UM	Color
776	Butter	Non-fat	G	Yellow
778	Sugar	Normal sugar	KG	White

Data warehouse – Product dimension

Dim Key	Original Key	Name	Description	UM	Color
1	123	Milk	Non-fat	L	White
2	776	Sugar	Brown sugar	KG	Brown
3	776	Butter	Non-fat	G	Yellow
4	778	Sugar	Normal sugar	KG	White



# Advantages of Using a Surrogate Key



**Integrate multiple source systems**

**Keep track of attribute changes over time**



# Advantages of Using a Surrogate Key



**Integrate multiple source systems**  
**Keep track of attribute changes over time**

- Use surrogate keys to handle multiple versions of a row



# Keep Track of Attribute Changes over Time



# Keep Track of Attribute Changes over Time

Before 09-16-2018

Key	Name	Description	UM	Color
387	Cherry toffee	Great cherry taste, sticky, chewy consistency	Piece	Pink



# Keep Track of Attribute Changes over Time

Before 09-16-2018

Key	Name	Description	UM	Color
387	Cherry toffee	Great cherry taste, sticky, chewy consistency	Piece	Pink

After 09-16-2018

Key	Name	Description	UM	Color
387	Super cherry toffee	Great cherry taste, sticky, chewy consistency	Piece	Pink



# Keep Track of Attribute Changes over Time

Before 09-16-2018

Key	Name	Description	UM	Color
387	Cherry toffee	Great cherry taste, sticky, chewy consistency	Piece	Pink



After 09-16-2018

Key	Name	Description	UM	Color
387	Super cherry toffee	Great cherry taste, sticky, chewy consistency	Piece	Pink



# Keep Track of Attribute Changes over Time

Before 09-16-2018

Key	Name	Description	UM	Color
387	Cherry toffee	Great cherry taste, sticky, chewy consistency	Piece	Pink

After 09-16-2018

Key	Name	Description	UM	Color
387	Super cherry toffee	Great cherry taste, sticky, chewy consistency	Piece	Pink

Data warehouse - Product dimension

Dim Key	Orig. Key	Name	Description	UM	Color
112	387	Cherry toffee	Great cherry taste, sticky, chewy consistency	Piece	Pink
244	387	Super cherry toffee	Great cherry taste, sticky, chewy consistency	Piece	Pink



# Keep Track of Attribute Changes over Time

Before 09-16-2018

Key	Name	Description	UM	Color
387	Cherry toffee	Great cherry taste, sticky, chewy consistency	Piece	Pink

After 09-16-2018

Key	Name	Description	UM	Color
387	Super cherry toffee	Great cherry taste, sticky, chewy consistency	Piece	Pink

Data warehouse - Product dimension

Dim Key	Orig. Key	Name	Description	UM	Color	Valid From	Valid To
112	387	Cherry toffee	Great cherry taste, sticky, chewy consistency	Piece	Pink	01-01-2017	09-16-2018
244	387	Super cherry toffee	Great cherry taste, sticky, chewy consistency	Piece	Pink	09-17-2018	12-31-9999



# Advantages of Using a Surrogate Key



- Integrate multiple source systems
- Keep track of attribute changes over time
- Protect the data warehouse from operational changes



# Protect the Data Warehouse from Operational Changes



## Characteristics of an operational system:

- Logs and processes daily activities/current actions
- Some data history is preserved
- Obsolete data is deleted after a period of time
- Codes/IDs can be reassigned to new data



# Protect the Data Warehouse from Operational Changes



# Protect the Data Warehouse from Operational Changes

Before 01-01-2018

Key	Name	Description	UM	Color
432	Chocolate chip cookie	Grandma's cookie recipe	Piece	Brown
776	Sugar	Brown sugar	KG	Brown



# Protect the Data Warehouse from Operational Changes

Before 01-01-2018

Key	Name	Description	UM	Color
432	Chocolate chip cookie	Grandma's cookie recipe	Piece	Brown
776	Sugar	Brown sugar	KG	Brown

After 01-01-2018

Key	Name	Description	UM	Color
776	Sugar	Brown sugar	KG	Brown



# Protect the Data Warehouse from Operational Changes

Before 01-01-2018

Key	Name	Description	UM	Color
432	Chocolate chip cookie	Grandma's cookie recipe	Piece	Brown
776	Sugar	Brown sugar	KG	Brown



After 01-01-2018

Key	Name	Description	UM	Color
776	Sugar	Brown sugar	KG	Brown



# Protect the Data Warehouse from Operational Changes

Before 01-01-2018

Key	Name	Description	UM	Color
432	Chocolate chip cookie	Grandma's cookie recipe	Piece	Brown
776	Sugar	Brown sugar	KG	Brown

After 01-01-2018

Key	Name	Description	UM	Color
776	Sugar	Brown sugar	KG	Brown

Data warehouse – Product dimension

Dim Key	Orig. Key	Name	Description	UM	Color	Valid From	Valid To
83	432	Chocolate chip cookie	Grandma's cookie recipe	Piece	Brown	01-01-1753	01-01-2018
2	776	Sugar	Normal sugar	KG	Brown	01-01-1753	12-31-9999



# Advantages of Using a Surrogate Key



- Integrate multiple source systems**
- Keep track of attribute changes over time**
- Protect the data warehouse from operational changes**
- Handle null or unknown conditions**



# Advantages of Using a Surrogate Key



**Integrate multiple source systems**

**Keep track of attribute changes over time**

**Protect the data warehouse from operational changes**

**Handle null or unknown conditions**

- Empty row technique



# Advantages of Using a Surrogate Key



**Integrate multiple source systems**

**Keep track of attribute changes over time**

**Protect the data warehouse from operational changes**

**Handle null or unknown conditions**

- Empty row technique
- Handle situations when a link does not exist between fact and dimensions



# Advantages of Using a Surrogate Key



**Integrate multiple source systems**

**Keep track of attribute changes over time**

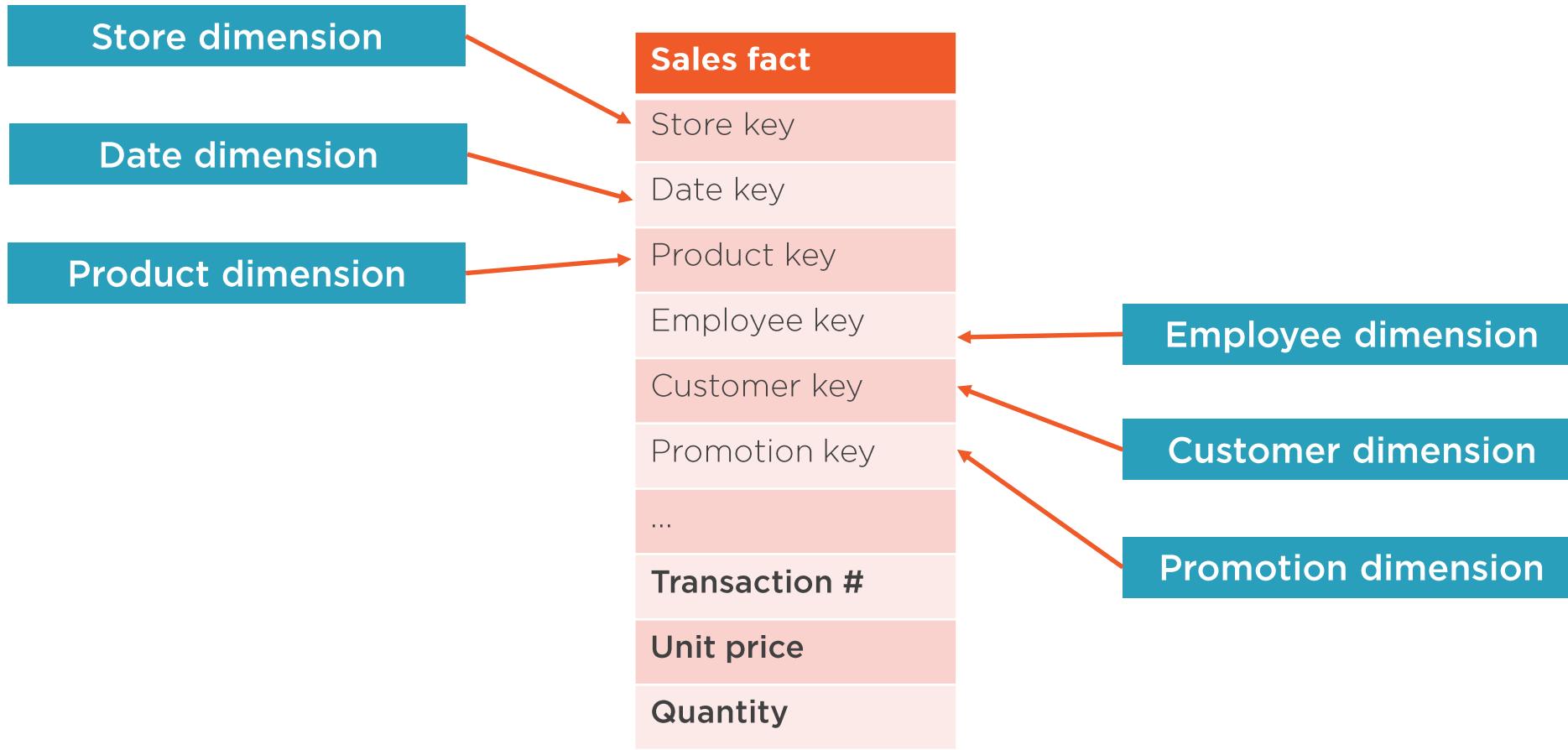
**Protect the data warehouse from operational changes**

**Handle null or unknown conditions**

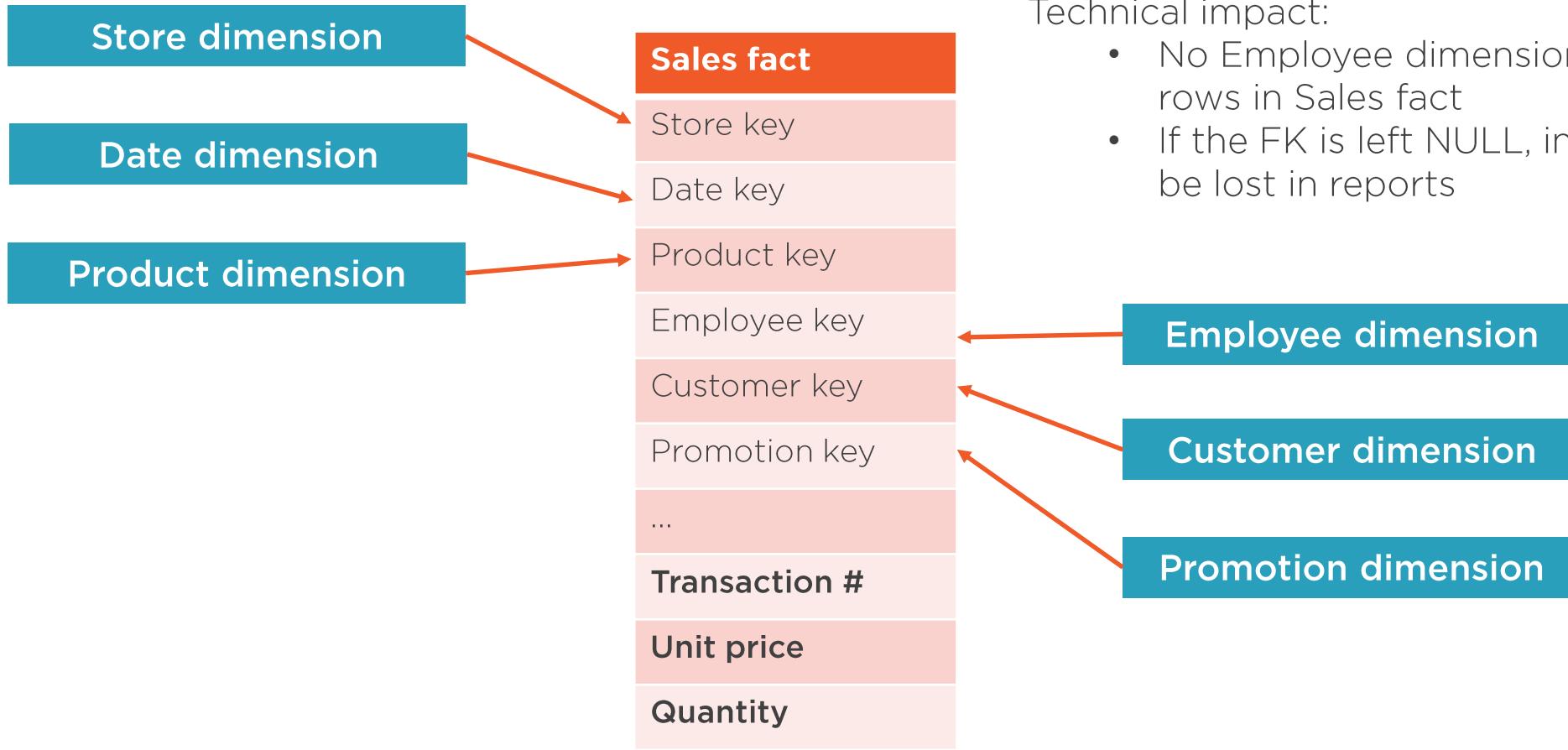
- Empty row technique
- Handle situations when a link does not exist between fact and dimensions



# Scenario for Null or Unknown Conditions



# Scenario for Null or Unknown Conditions



Self-checkout kiosks were introduced  
Customers pay by themselves  
Employees are not involved for this transaction  
Technical impact:

- No Employee dimension key for these rows in Sales fact
- If the FK is left NULL, information may be lost in reports



# Handle Null or Unknown Conditions

Adding the “empty row” to each dimension table

Key	Name	Birthday	Gender	PhoneNr.
1	Unknown	01-01-1753	Unknown	Unknown
32	Alice Cohen	03-05-1989	Female	Unknown



# Handle Null or Unknown Conditions

Adding the “empty row” to each dimension table

Key	Name	Birthday	Gender	PhoneNr.
1	Unknown	01-01-1753	Unknown	Unknown
32	Alice Cohen	03-05-1989	Female	Unknown

Data warehouse – Sales fact table

Fact Key	Product Key	Customer Key	Employee Key	Transaction Nr	Amt.
...	432	10	32	#1050	\$5
...	776	12	1	#2367	\$3



# Advantages of Using a Surrogate Key



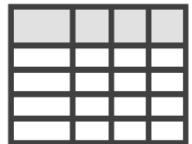
- Integrate multiple source systems**
- Keep track of attribute changes over time**
- Protect the data warehouse from operational changes**
- Handle null or unknown conditions**
- Improve performance**



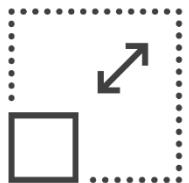
# Improve Performance



The surrogate key should be a small number, preferably an integer



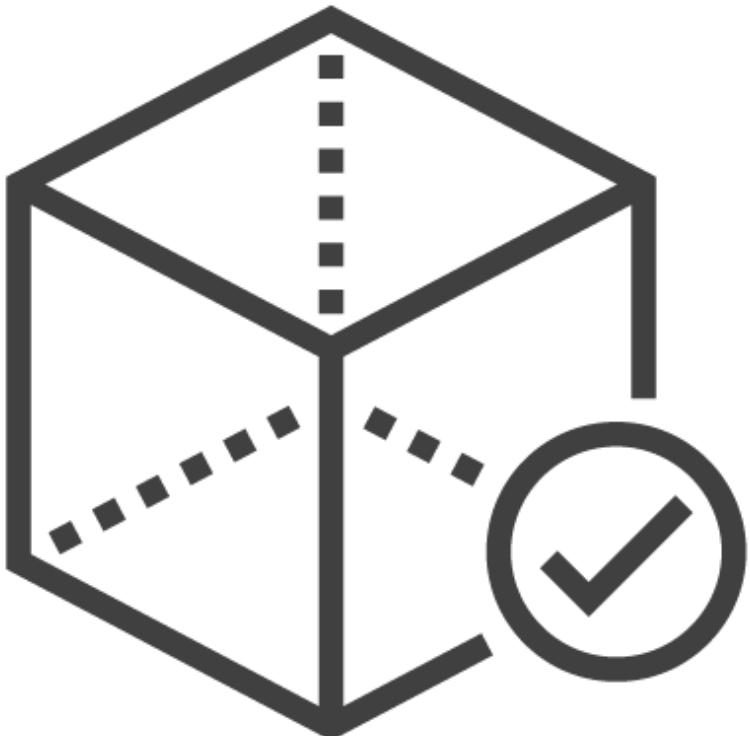
You need to anticipate the growth of your dimension table (a 4-byte int can handle ~2 billion positive values and ~4 billion values in total)



The smaller the surrogate key, the smaller the fact table



# Dimensions of a Fact Table



## Fact tables

- Have FKs to a lot of dimension tables
- Grow very fast
- Every byte counts!

FK length	Fact table growth	Fact table storage
X bytes	1 billion rows	Y GB
X + 1 bytes	1 billion rows	Y + 1 GB



# Advantages of Using a Surrogate Key



- Integrate multiple source systems**
- Keep track of attribute changes over time**
- Protect the data warehouse from operational changes**
- Handle null or unknown conditions**
- Improve performance**



# Business Keys

---



# Business Keys



**The primary key of the source system**

**Also known as**

- Natural keys
- Production keys
- Operational keys

# Business Keys Properties

**Attribute in the dimension  
table**



# Business Keys Properties

**Attribute in the dimension  
table**

**Can be prefixed with a source  
system's code  
(POS|432 or CSV|635)**



# Business Keys Properties

**Attribute in the dimension table**

**Can be prefixed with a source system's code  
(POS|432 or CSV|635)**

**Can cause duplicated items in the dimensions**



# Business Keys Properties

**Attribute in the dimension table**

**Can be prefixed with a source system's code  
(POS|432 or CSV|635)**

**Can cause duplicated items in the dimensions**

**If the BK is composed of meaningful codes, it should be split and each code stored in separate columns**



# Keys in a Dimension Table



## Surrogate key

- Primary key, no business value

## Business key

- Original primary from the source system



# Other Dimensional Design Considerations

---



# Characteristics of a Dimension Table



**Companion of a fact table**

**Used to describe the business in clear terms**

**Design implications**

- Store descriptive words instead of codes
- Replace flags or indicators with descriptions
- Replace null values with meaningful words



# Use Descriptions Instead of Codes



**Each code stored should be accompanied by a descriptive decode**

**Each attribute should be easy to interpret by people**



# Example of Using Descriptions Instead of Codes

## Country of import for the most sold products

# Sold products	Name	Imported from
5000	Zywiec Beer	PL
4856	Pan Beer	HR
3991	Ursus Beer	RO
2674	Mort Subite Beer	BE

Codes are not easy to interpret by everyone

Id	Name	Imported from
5000	Zywiec Beer	Poland
4856	Pan Beer	Croatia
3991	Ursus Beer	Romania
2674	Mort Subite Beer	Belgium

Decodes can be retrieved from the operational system as well



# Use Textual Attributes Instead of Flags and Indicators



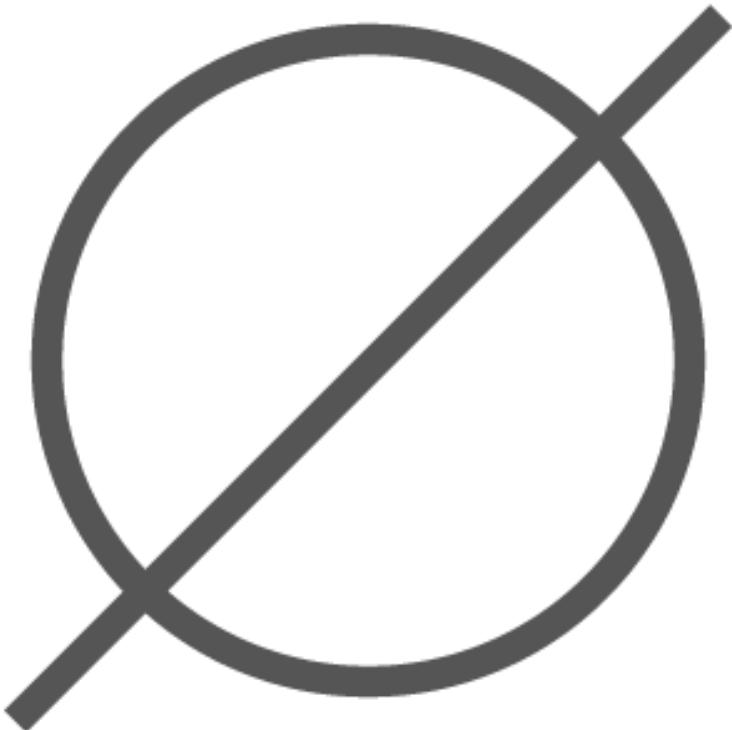
Flags and operational indicators should be supplemented with meaningful text words

Examples from the Date dimension:

Column	Meaningful name	Indicator
Weekday indicator	Weekday/Weekend	1/0
Holiday indicator	Holiday/Non-holiday	true/false



# Replace Null Values in Dimensions



**Replace null values with meaningful words, like**

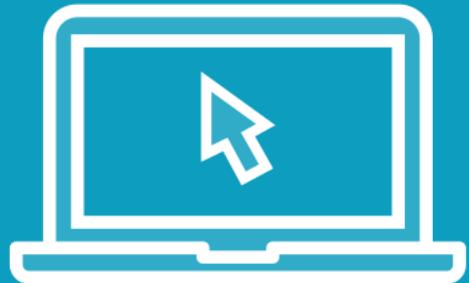
- Unknown
- Not applicable

**Null values in data warehouse tables have unexpected behaviors**

**Nulls are not present in drop-down menus, so data may be missing from reports**



# Demo



## Creating the Date dimension

- What is a Date dimension?
- Why is it important?



# What Is a Date Dimension?



A table that stores dates

Is populated with a large number of dates  
(10 years, 100 years)

As a dimension table, it is not large

It is loaded once, not periodically



# Why Is It Important?



**Is present in almost all data warehouses**

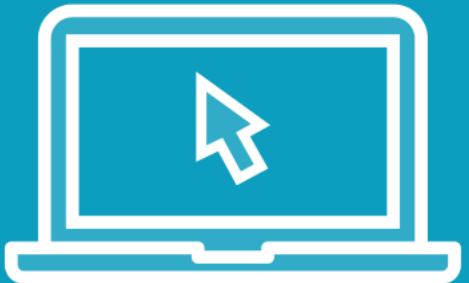
**Helps performing time analysis**

- Products sold this year vs. last year
- Products sold per month compared to last year
- Per week, per weekend, per holiday..

**Contains pre-calculated information about dates**



# Demo

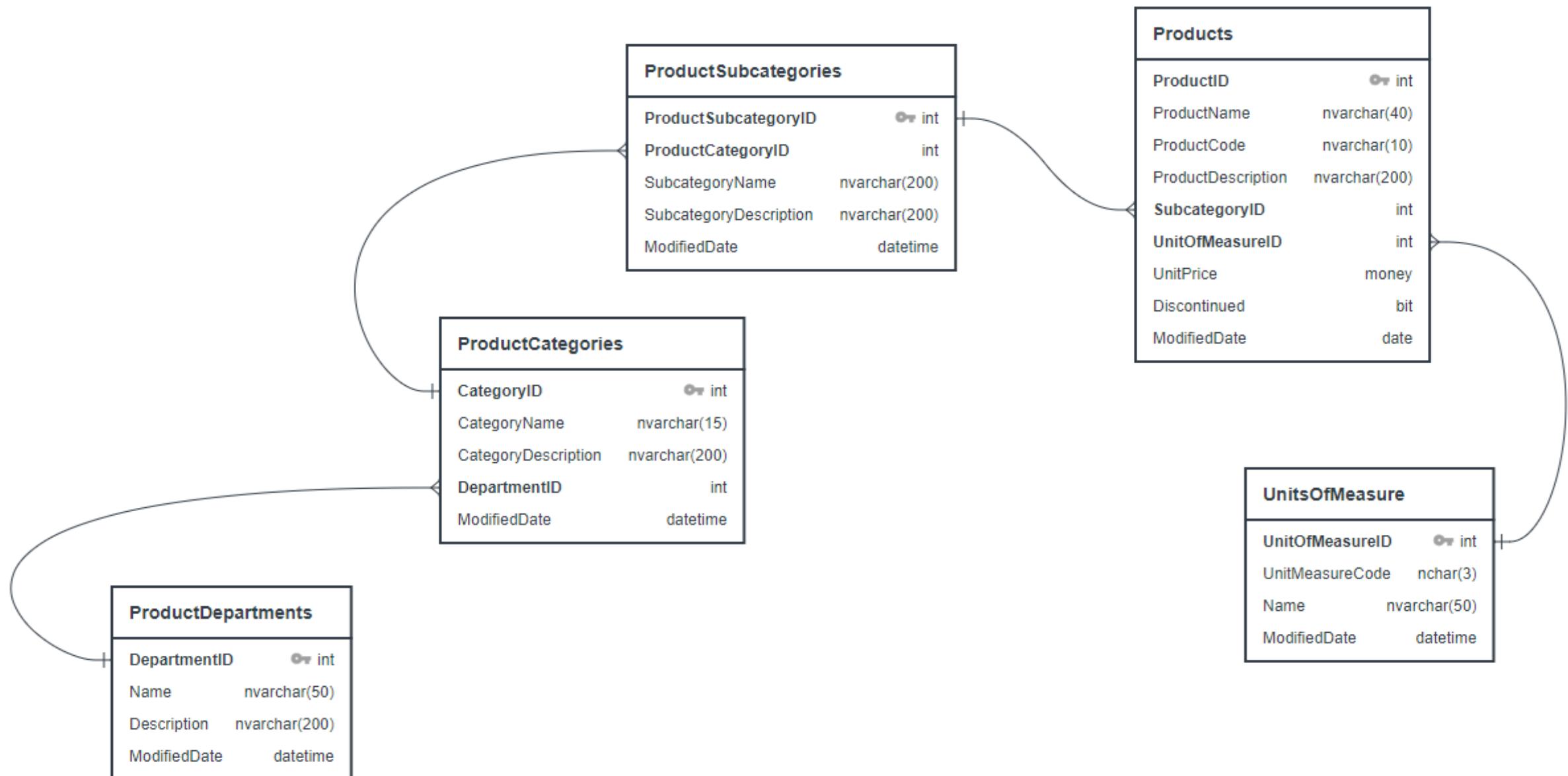


## Creating the Product dimension

- Product dimension is present in many data warehouse projects
- It doesn't have a standard set of attributes



# Product-related Tables



# Summary



## Characteristics of a dimension table

- Is different than a relational table
- Integrates data from multiple sources
- Data is easy to read by the user

## Keys in a dimension table

- Surrogate key
- Business key

## Guidelines for implementing a dimension table

- Store descriptive words instead of codes
- Replace flags or indicators with descriptions
- Replace null values with meaningful words



# Creating and Working with Hierarchies

---



**Ana Voicu**  
@ana\_voicu



# Overview



**In short, a hierarchy is:**

- A data structure
- Created with attributes from a dimension
- Used for data aggregation

**Topics elaborated in this chapter:**

- What is a hierarchy?
- Why is it useful?
- What does drilling-down mean?
- Implementing a hierarchy in a data warehouse



# What Is a Hierarchy?

---

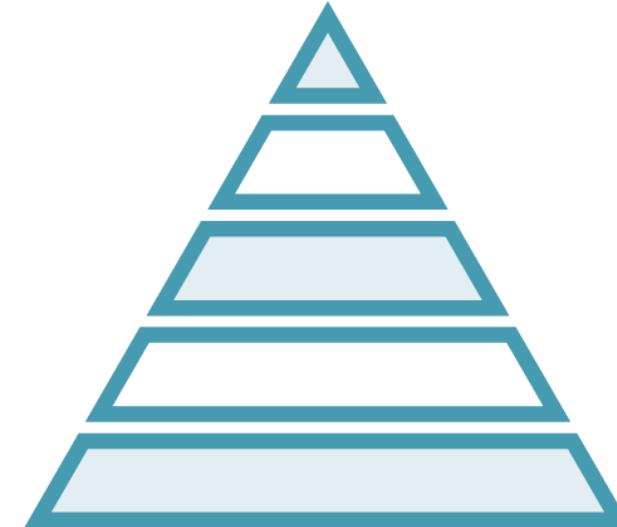


# What Is a Hierarchy?



**Data structure with multiple levels**

- The levels are dimension attributes
- The elements on each level are called nodes

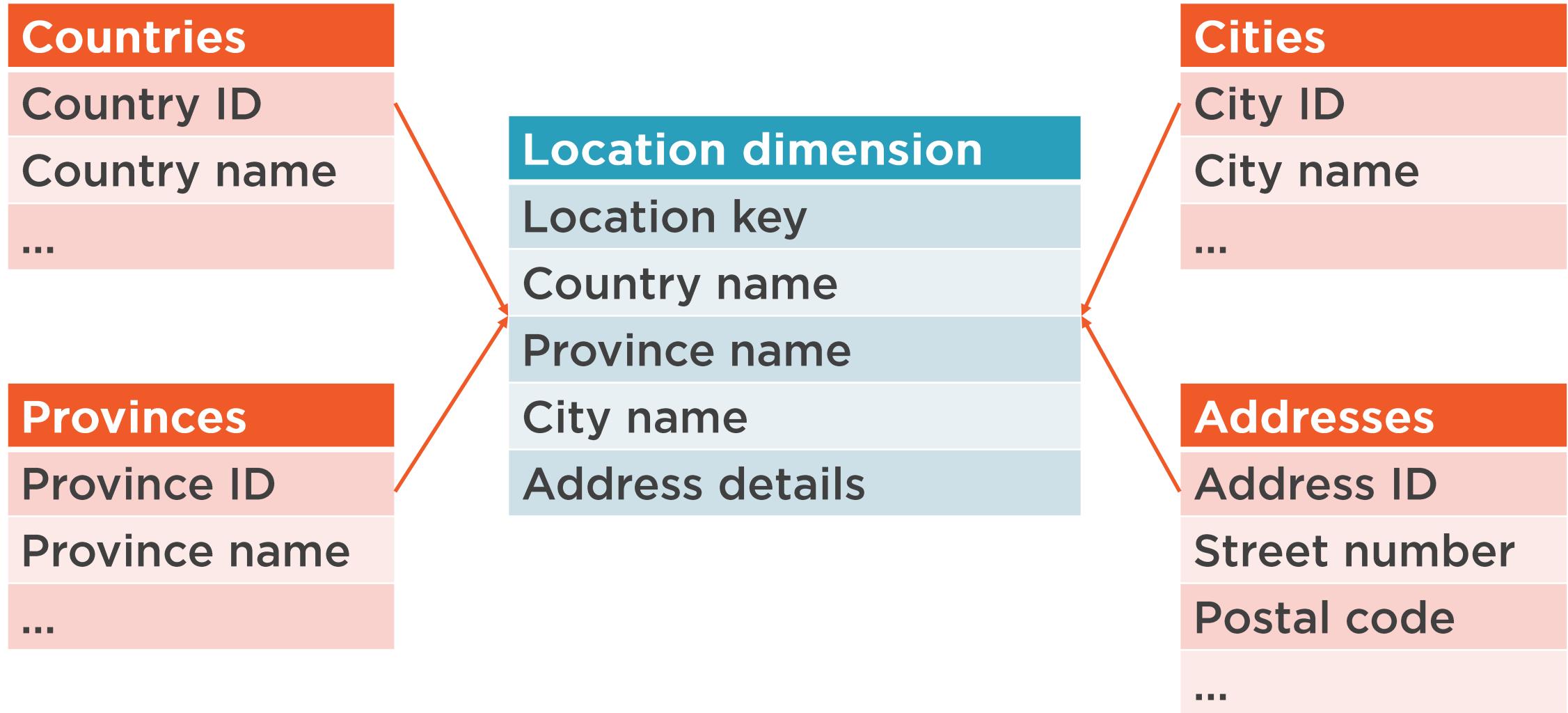


**Similar to a pyramid**

- Bottom level is the weakest
- Highest level is at the top



# A Hierarchy in the Location Dimension



# A Hierarchy in the Location Dimension

Bucharest

Vienna

New York

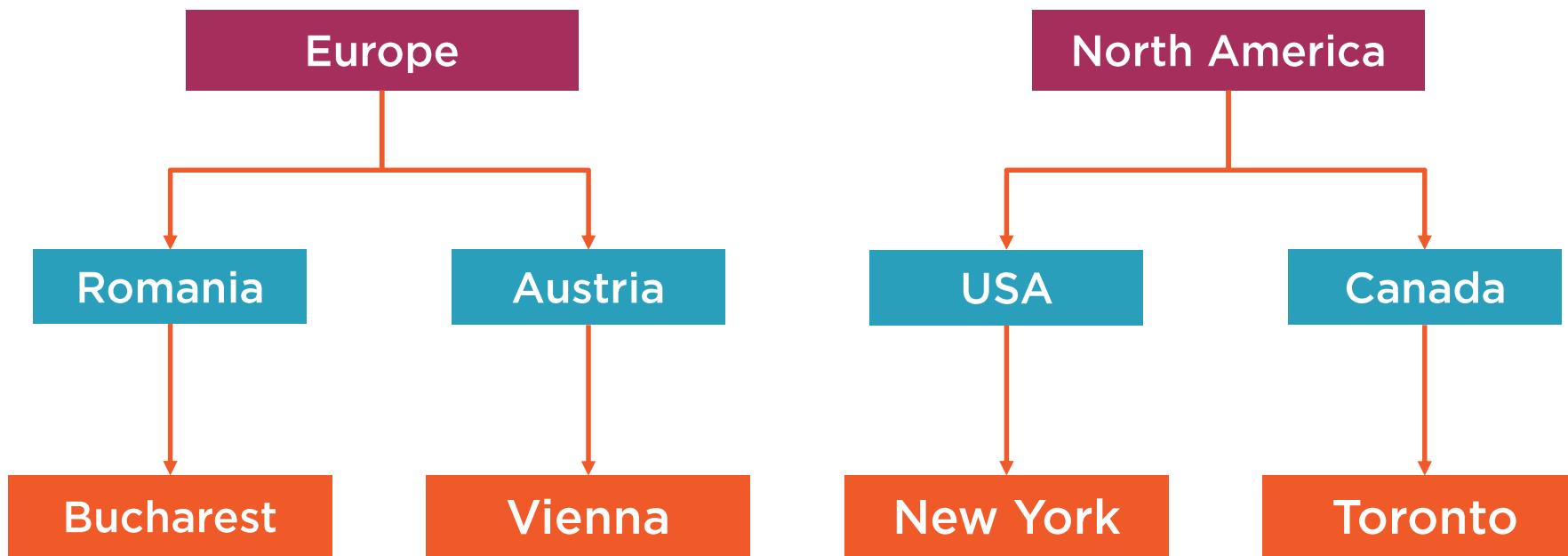
Toronto



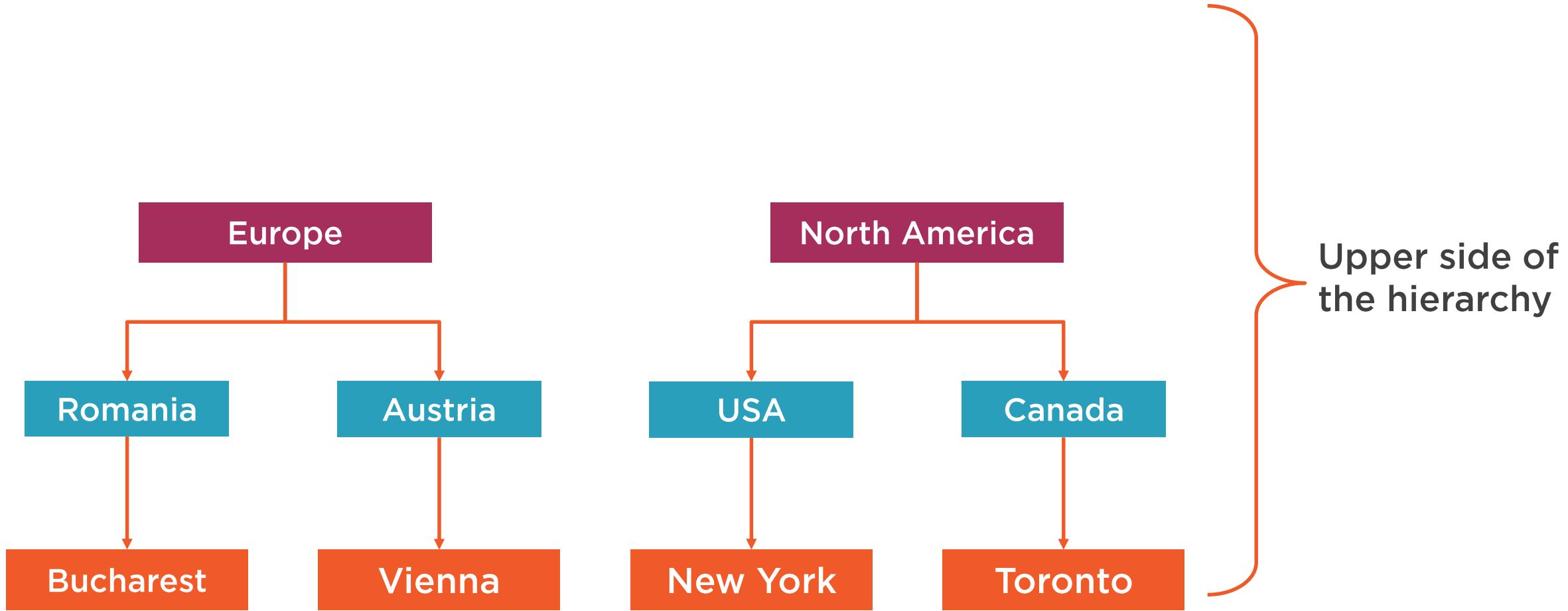
# A Hierarchy in the Location Dimension



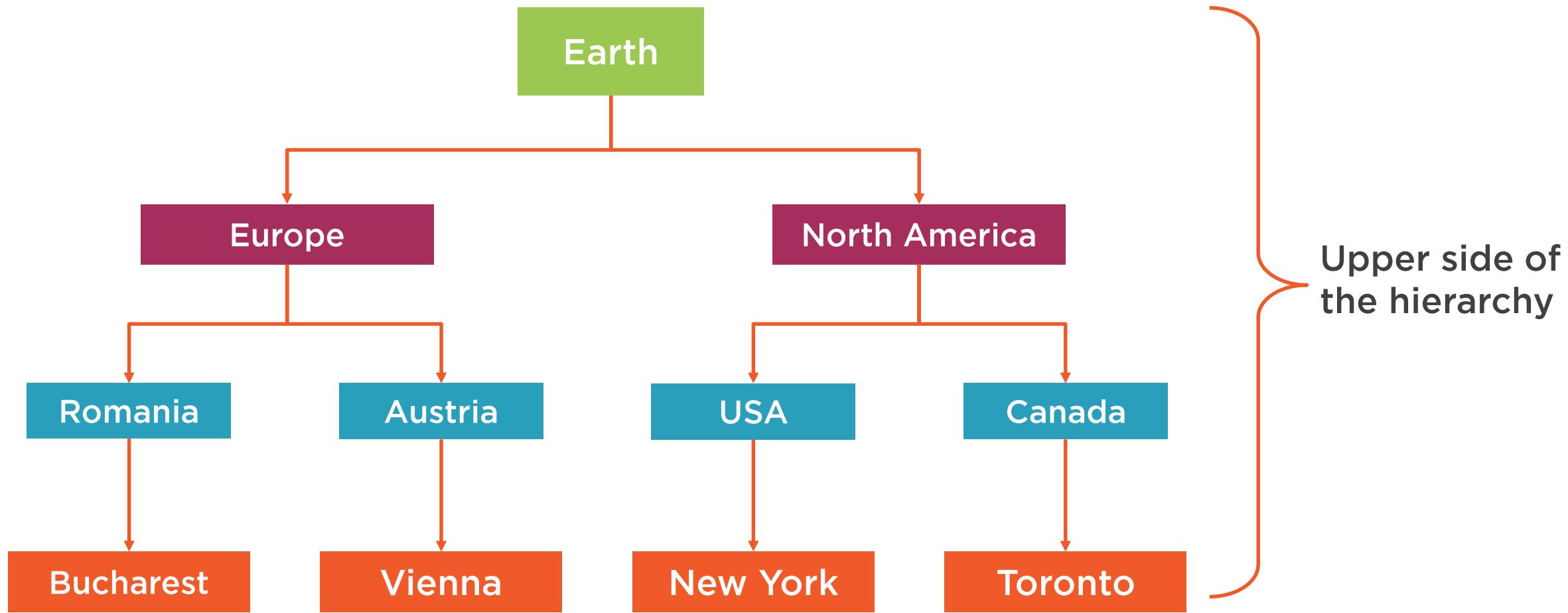
# A Hierarchy in the Location Dimension



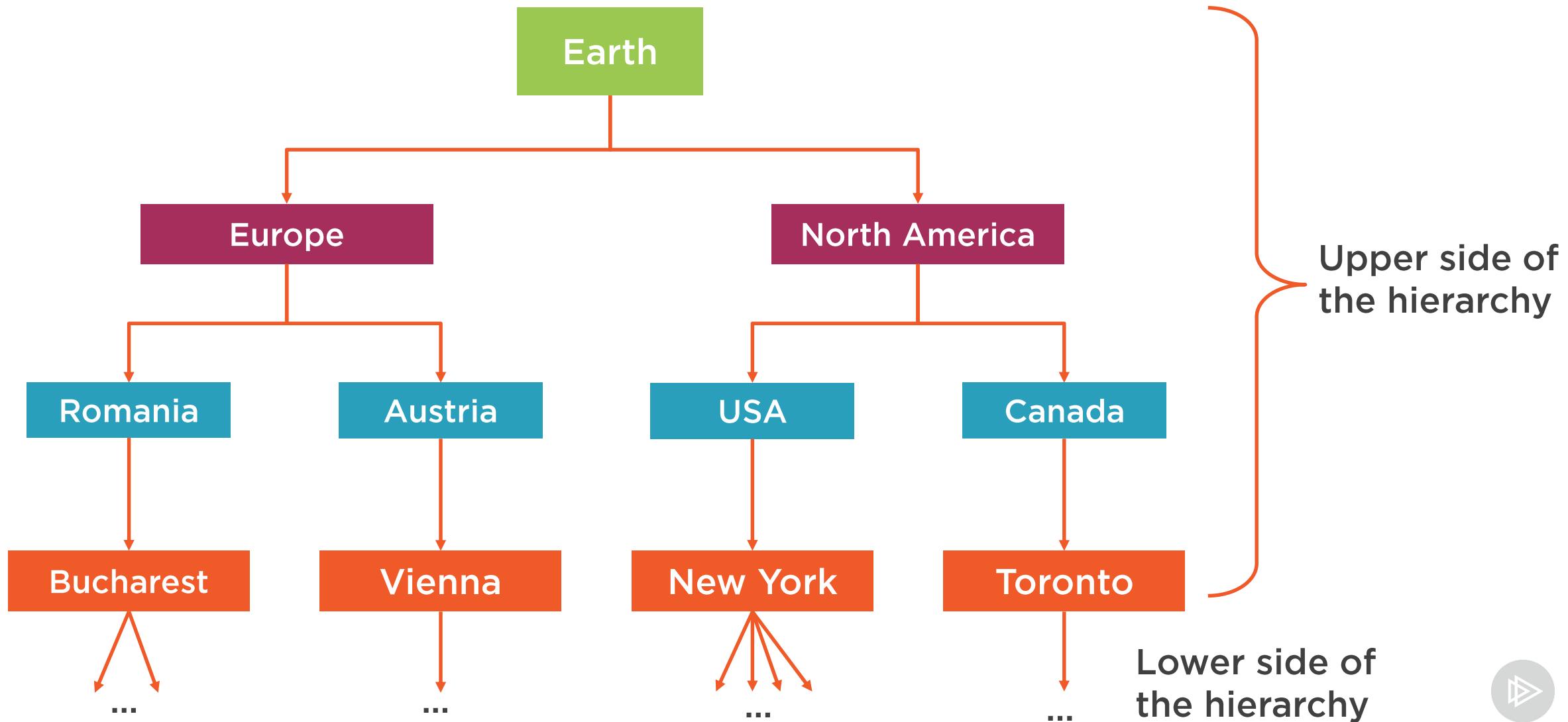
# A Hierarchy in the Location Dimension



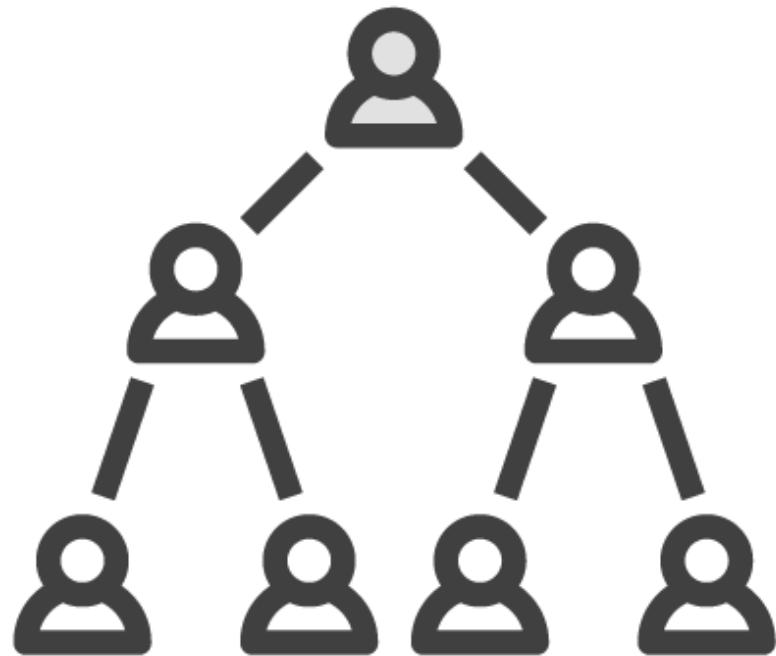
# A Hierarchy in the Location Dimension



# A Hierarchy in the Location Dimension



# What Is a Hierarchy?



## Data structure

- Attributes of a dimension are organized together
- Going down the hierarchy nodes -> more details
- Going up the hierarchy -> summarized data



# Advantages of Using Hierarchies

## Structure

- Important dimensions have plenty attributes
- Analyzing important data can become overwhelming
- Hierarchies provide order in the data

## Multiple perspectives

Merchandise hierarchy



# Advantages of Using Hierarchies

## Structure

- Important dimensions have plenty attributes
- Analyzing important data can become overwhelming
- Hierarchies provide order in the data

## Multiple perspectives

### Merchandise hierarchy

- Department
  - Category
  - Subcategory
  - Product name



# Advantages of Using Hierarchies

## Structure

- Important dimensions have plenty attributes
- Analyzing important data can become overwhelming
- Hierarchies provide order in the data

## Multiple perspectives

### Merchandise hierarchy

- Department
  - Category
  - Subcategory
  - Product name

### Packaging hierarchy



# Advantages of Using Hierarchies

## Structure

- Important dimensions have plenty attributes
- Analyzing important data can become overwhelming
- Hierarchies provide order in the data

## Multiple perspectives

### Merchandise hierarchy

- Department
- Category
- Subcategory
- Product name

### Packaging hierarchy

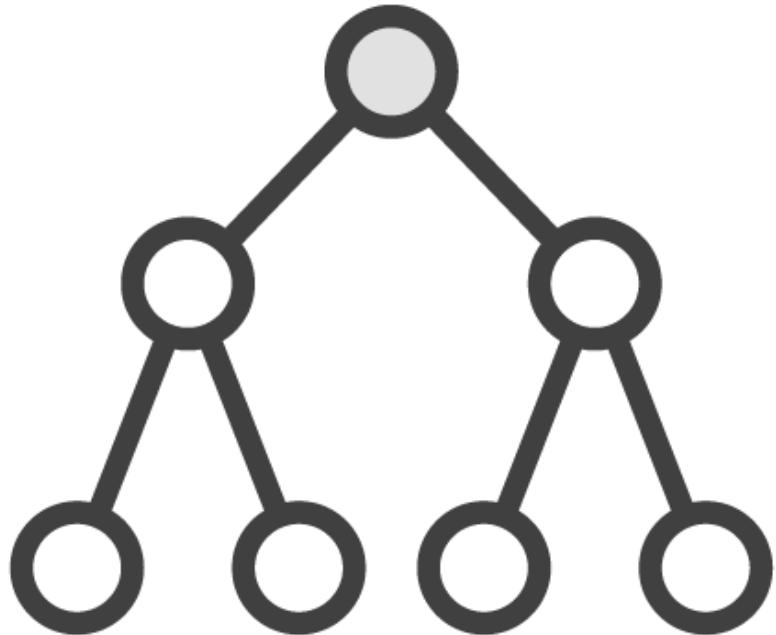
- Package type
- Package size
- Product name

## Zoom in/zoom out

- Visualize summarized data and detailed data
- Go as deep as the business requires it



# Types of Hierarchies



## Fixed-depth (balanced)

- Fixed number of levels
- Easy to create and work with

## Variable-depth (unbalanced or ragged)

- Uneven number of levels
- Creating them is more complex task

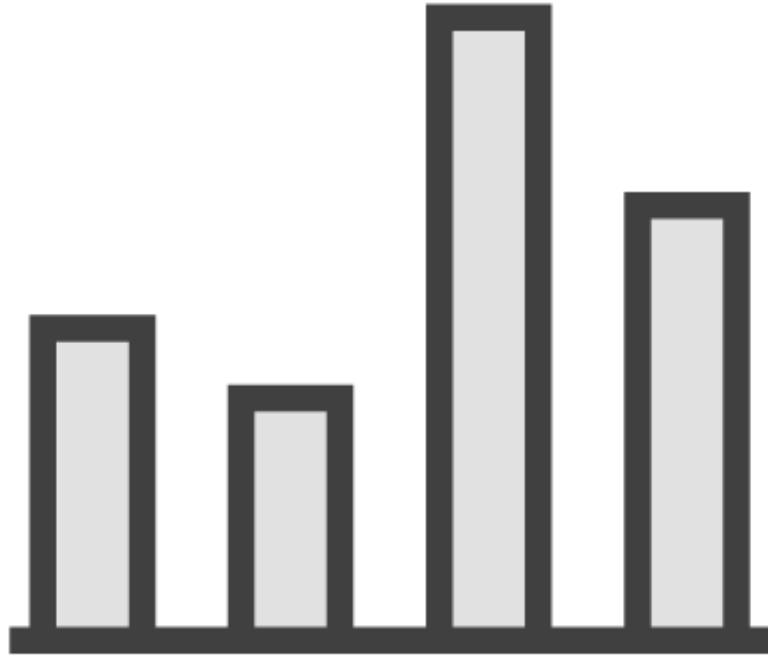


# Drilling Down a Hierarchy

---



# Data Warehouse Analysis



**Minimum requirements for doing data warehouse analysis**

- One fact table
- One dimension table

**Example: sales per product report**

- Sales fact
- Product dimension



# Example of Drilling Down

## Sales fact

Store key

Date key

Product key

Employee key

Customer key

...

## Transaction #

Unit price

Quantity

Amount



# Example of Drilling Down

Sales fact	Product dimension
Store key	Product key
Date key	Product name
Product key	Department
Employee key	Category
Customer key	Subcategory
...	Package size
<b>Transaction #</b>	Package type
<b>Unit price</b>	Description
<b>Quantity</b>	Unit of measure
<b>Amount</b>	...



# Example of Drilling Down

Sales fact
Store key
Date key
Product key
Employee key
Customer key
...
Transaction #
Unit price
Quantity
Amount

Product dimension
Product key
Product name
Department
Category
Subcategory
Package size
Package type
Description
Unit of measure
...



## Merchandise hierarchy

- Department
  - Category
    - Subcategory
      - Product name



# Example of Drilling Down

Sales

80.000

Department

Sales

Bakery	Sweets	Beverages
25.000	40.000	15.000



# Example of Drilling Down

Department  
Category  
Sales

Bakery			Sweets			Beverages	
Bread	Pie	Croissant	Cookie	Cake	Candy	Juice	Tea
7.000	13.000	5.000	8.700	6.300	5.000	7.000	6.600



# Example of Drilling Down

Department

Category

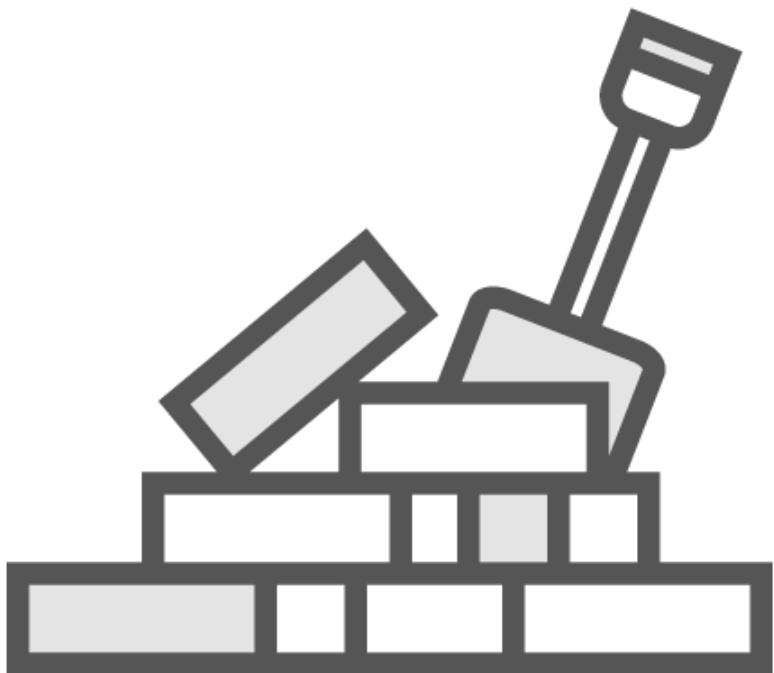
Subcategory

Sales

Bakery							
Bread			Pie			Croissant	
Baguette	Pita	Banana br.	Cream	Fruit	Custard	Sweet	Salty
2.000	3.000	2.000	2.500	2.200	8.300	3.000	2.000



# Drilling Down - Summary



## Drilling down

- Adding another member of the dimension to the report
- The member doesn't need to be part of a hierarchy

## Drilling up/rolling up

- Taking out an attribute of a dimension from a report

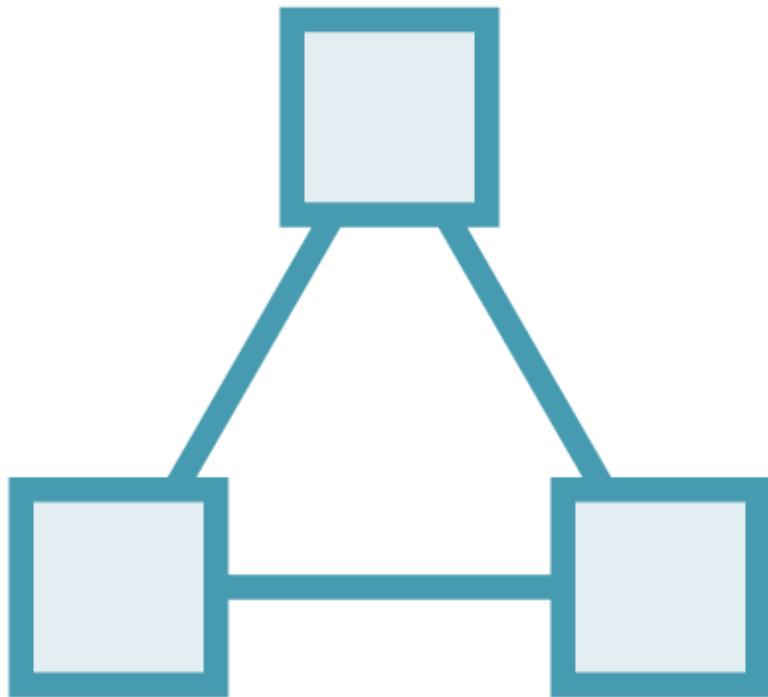


# Fixed-depth Positional Hierarchies

---



# Fixed-depth Positional Hierarchies



**The number of levels is known upfront**

**The levels are attributes in the dimension table**

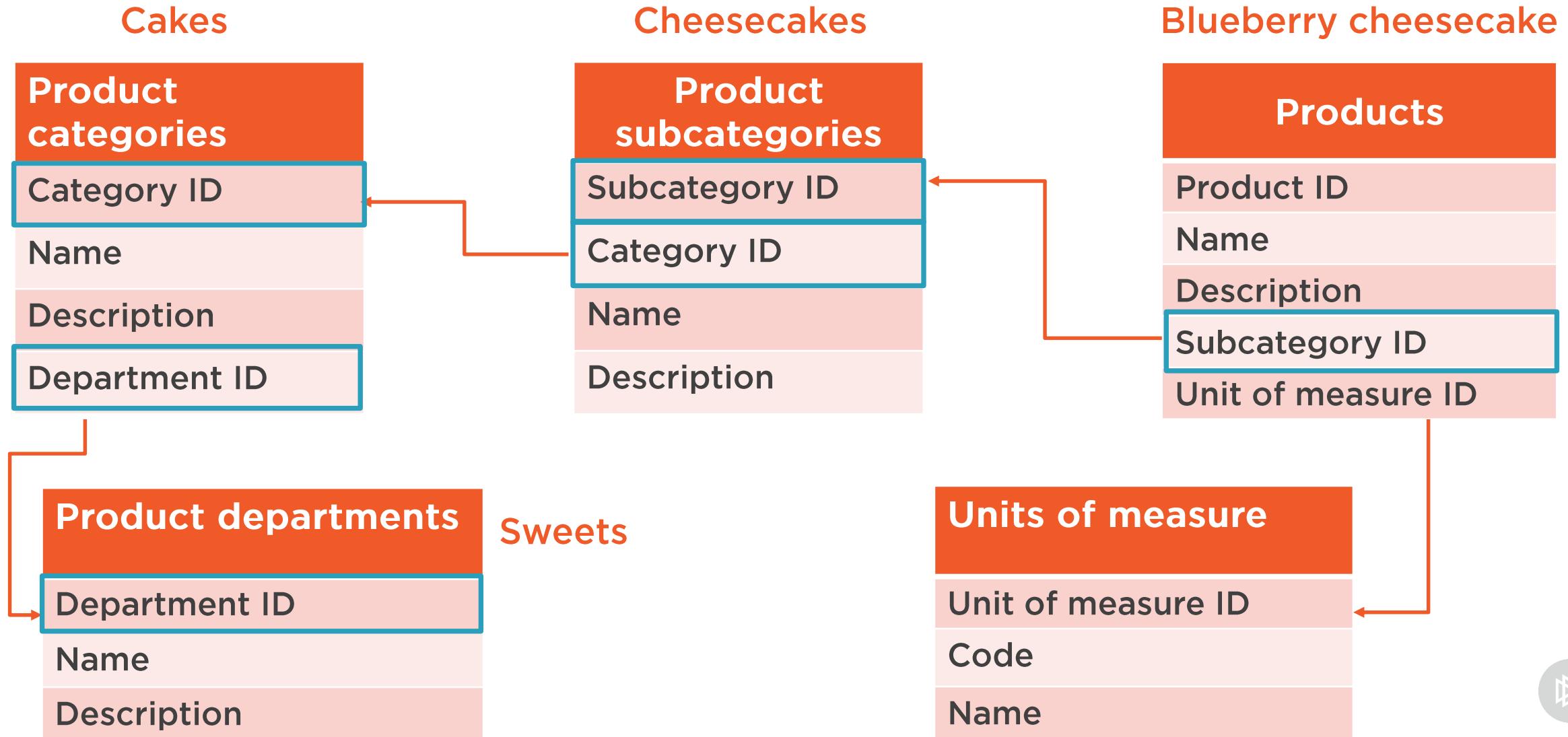
**It is a series of many-to-one relationships**

## Advantages

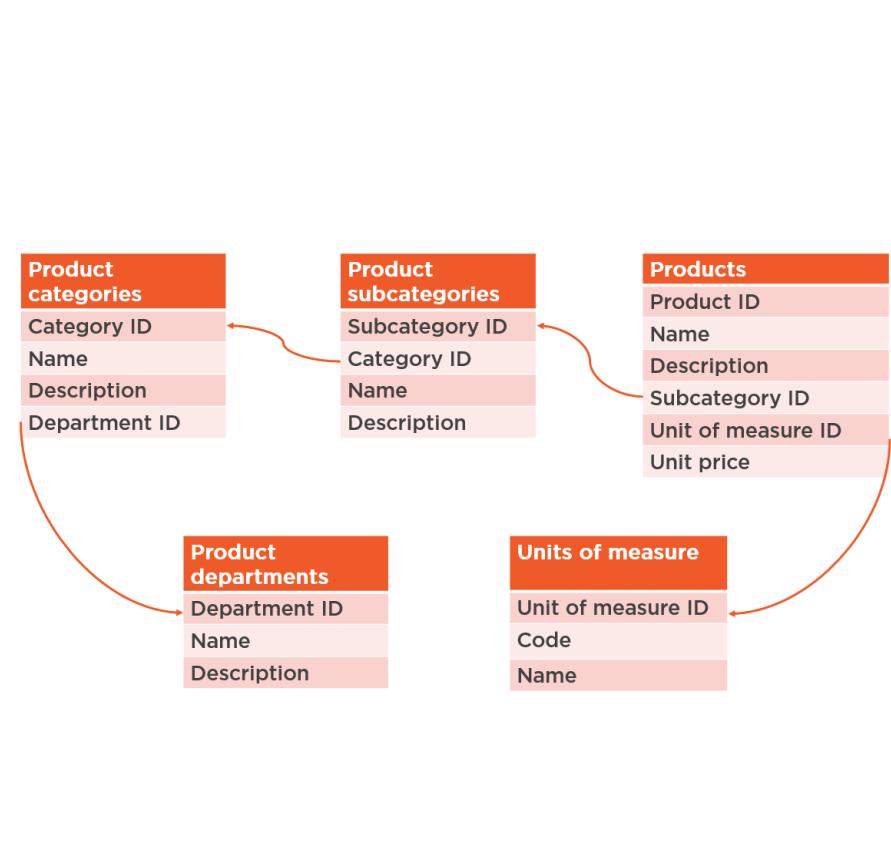
- Easy to navigate
- Offers predictable results
- No impact on performance



# Product Information from Different Tables



# Product Information from the Data Warehouse



## Keep in mind:

- Don't create many snowflake designs
- Most relationships should be from fact tables to the dimensions



# Creating the Merchandise Hierarchy

Department

Bakery

Category

Bread

Pie

Subcategory

Baguette

Banana  
bread

Custard

Cream

Product

Le Petit  
Francais

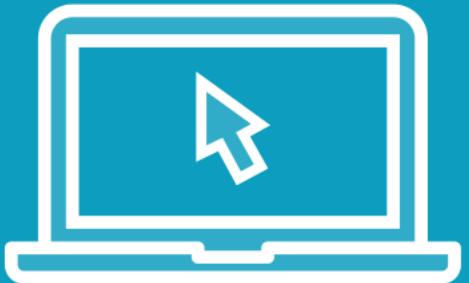
The Big  
Banana  
Bread

Eagle  
Brand  
Coconut  
Pie

Eagle  
Brand  
Banana  
Cream Pie



Demo



**Creating a fixed-depth positional hierarchy**

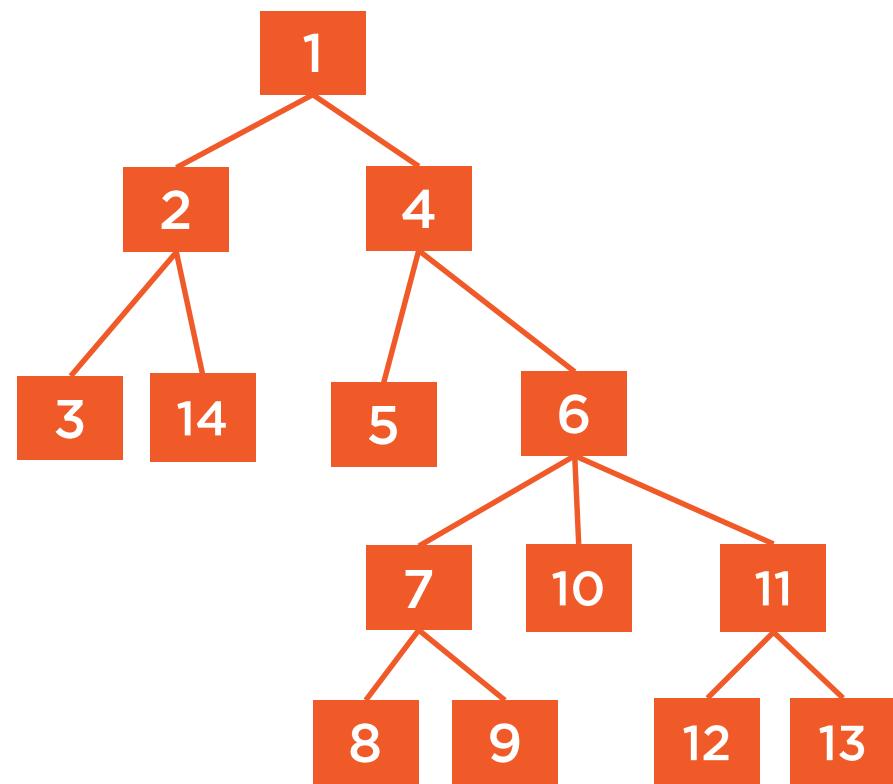


# Variable-depth Positional Hierarchies

---



# Characteristics of Variable-depth Hierarchies



**The number of levels is not known at design time**

- Example: an organizational chart

**Are more complex structures (compared to the fixed-depth hierarchies)**

**Should be used with moderation**



# Classification of Variable-depth Hierarchies



**Slightly ragged**

**Ragged, created with a hierarchy bridge**

**Ragged, created with pathstring attributes**



# Slightly Ragged Hierarchies



- The number of levels is not known beforehand
- The range in depth is small
- Geographic hierarchies are slightly ragged



# Example of a Slightly Ragged Hierarchy

## Location hierarchy

- Country
  - Province (or state)
    - City
      - Neighborhood
        - Address

## Examples of data with missing levels

Singapore (country and city)

Vatican (independent city-state)

Small cities, that don't have neighborhoods



# Fitting Data into a Slightly Ragged Hierarchy



**Step 1: Create all possible levels of the hierarchy**

**Step 2: Fill in the missing values per each level with:**

- An expression, similar to “not applicable”
- The value of the next parent member

**Step 3: Handle the ragged hierarchy as a fixed-depth one**



# Populating a Slightly Ragged Hierarchy

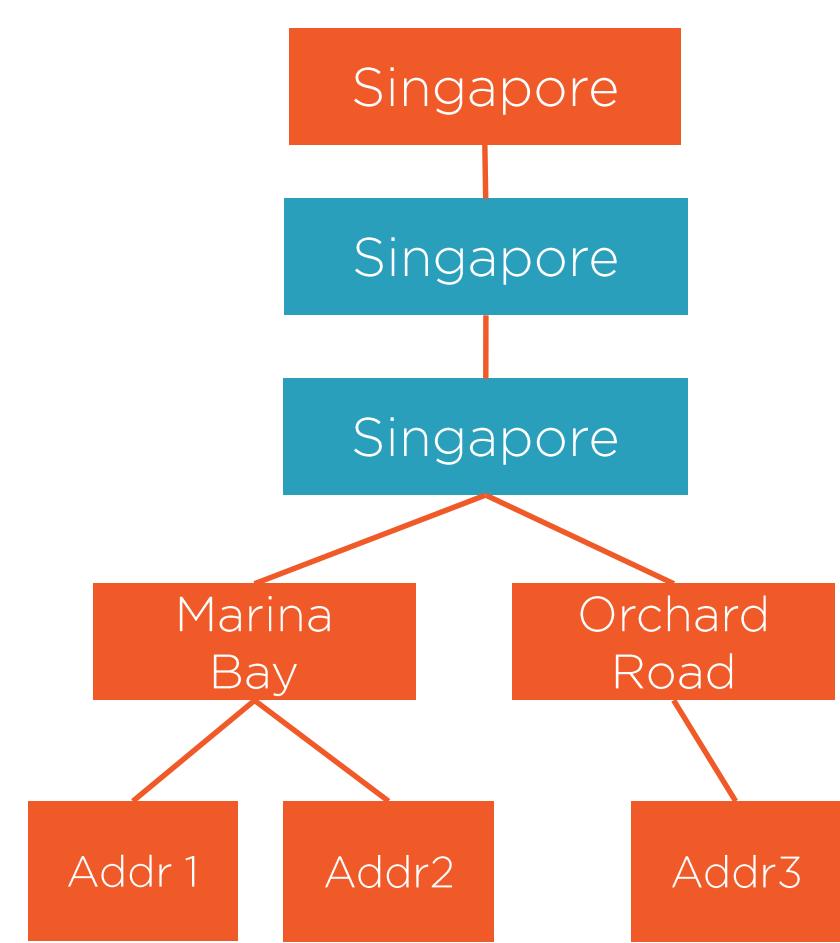
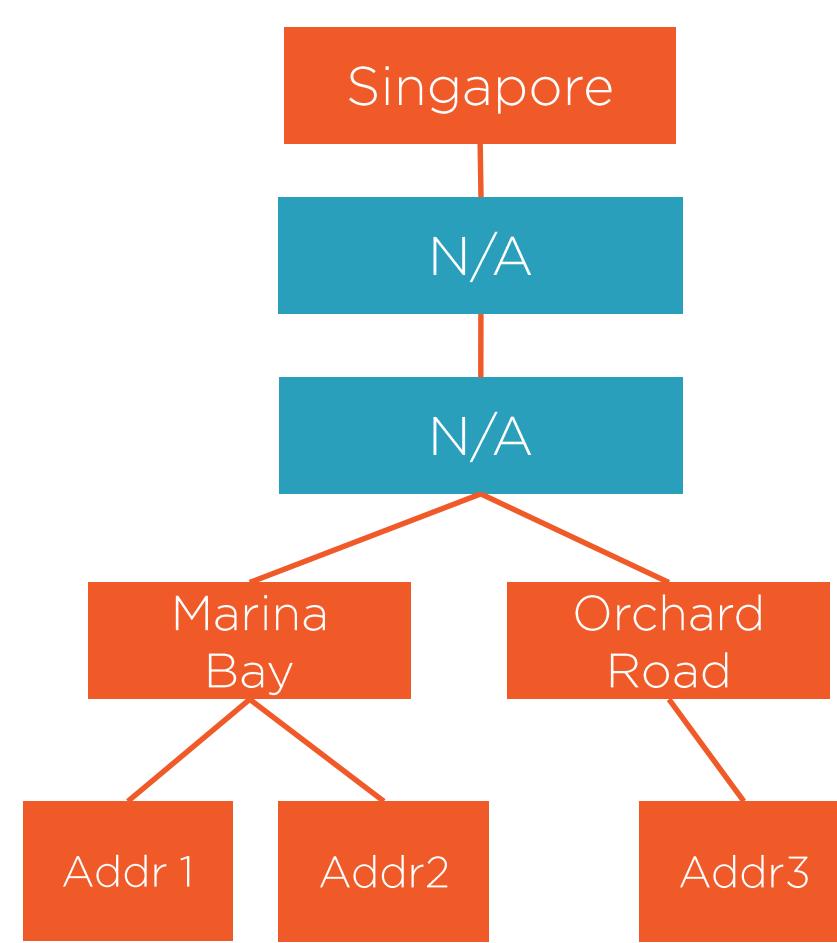
**Country**

↳ **Province**

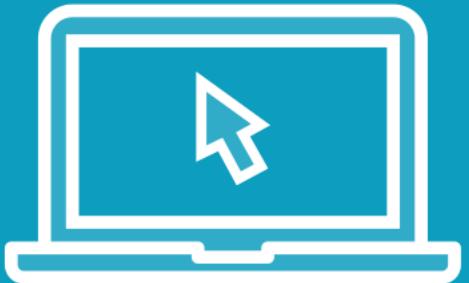
↳ **City**

↳ **Neighbor**

↳ **Address**



## Demo



### Creating and working with a slightly ragged hierarchy

- Based on the Location dimension



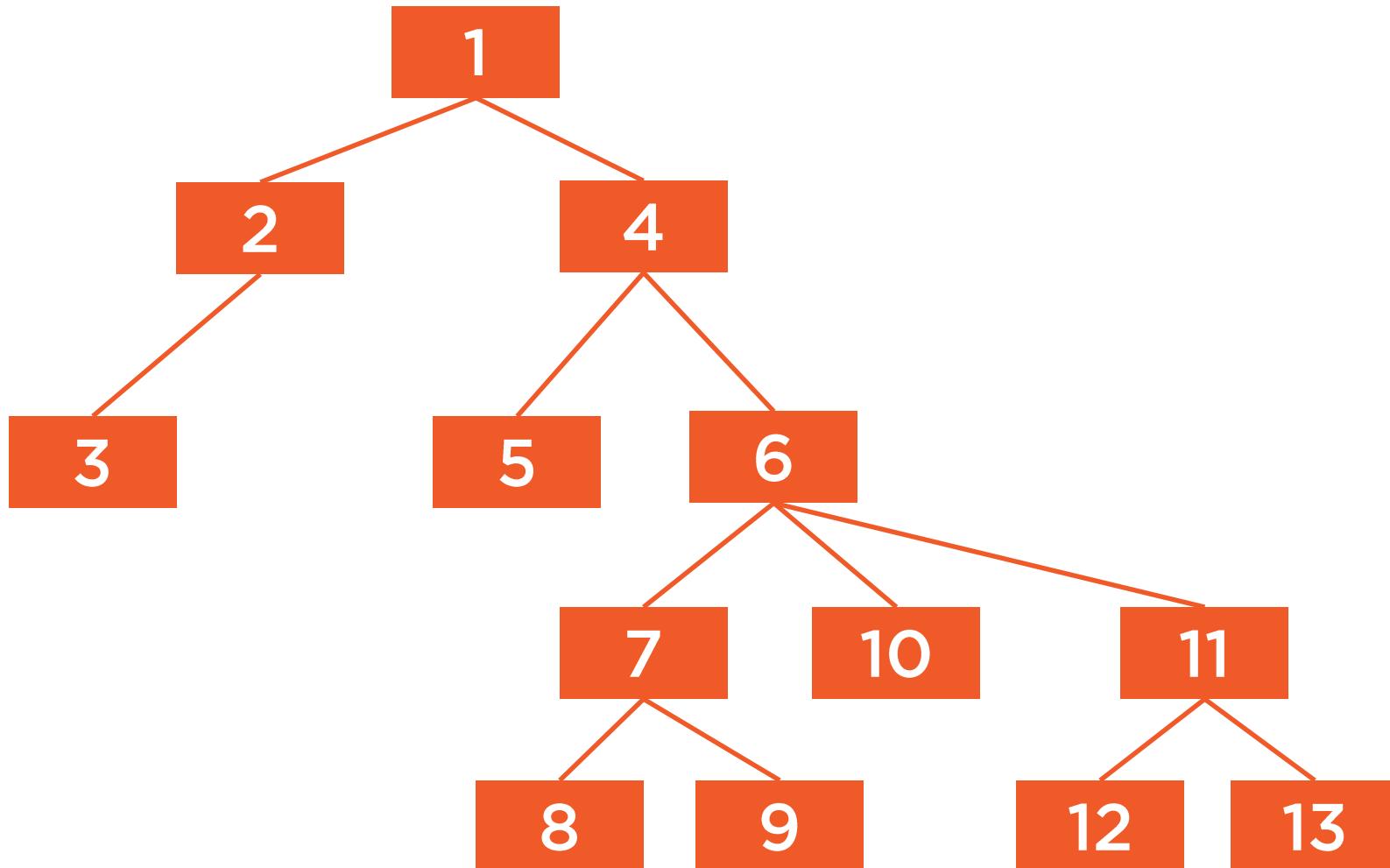
# Ragged Hierarchies

---



# Example of a Ragged Hierarchy

The organizational chart



# Implementing a Ragged Hierarchy Using a Bridge



**A row for each possible path in the hierarchy is stored in a table**

**Columns in the bridge table**

- ID of the parent
- ID of the child
- Number of levels between them
- Whether the node it a top node or bottom node
- Other information relevant for analysis

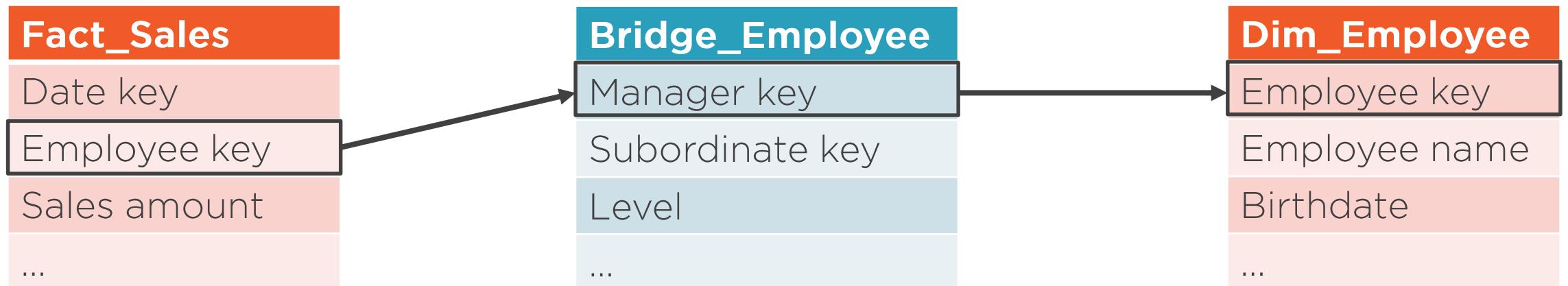
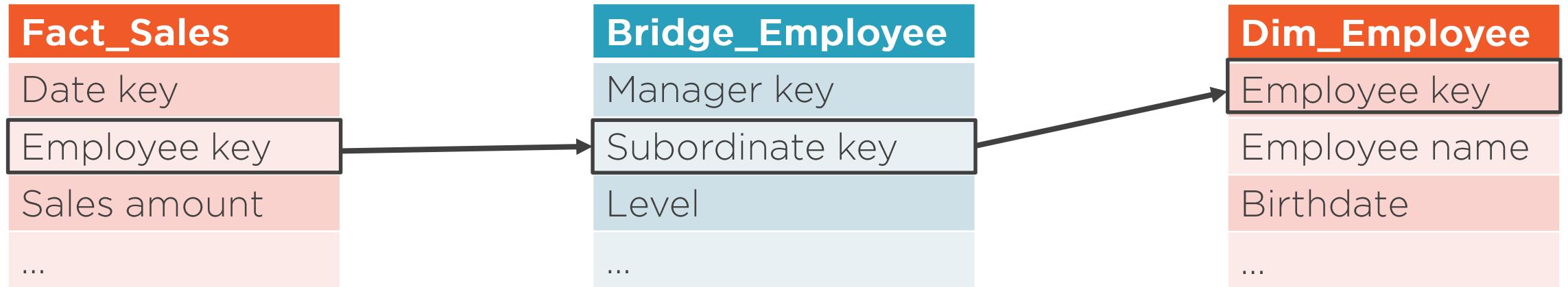


# Example of a Hierarchy Bridge Table

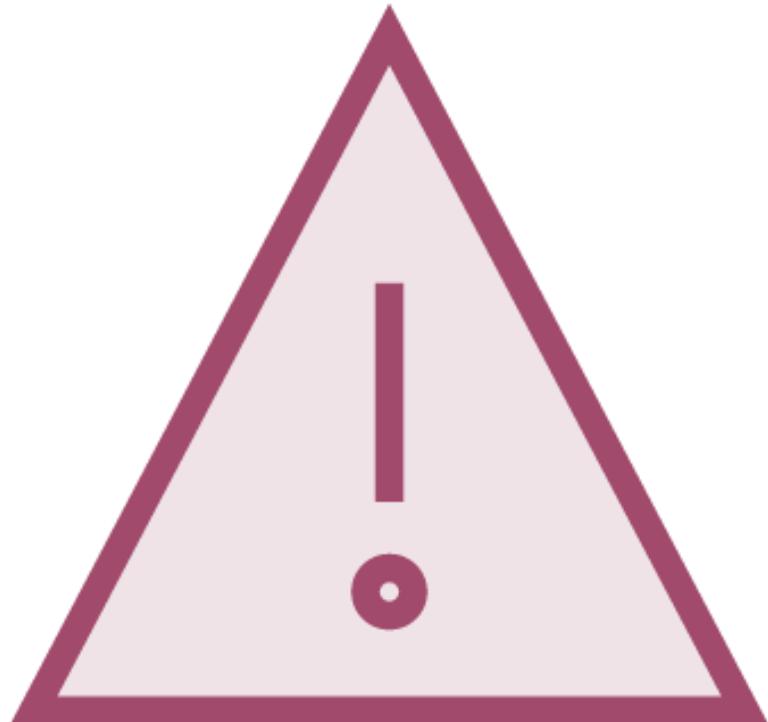
Parent key	Child Key	Level	Top flag	Bottom flag
Julia	Julia	0	Y	N
Julia	Marc	1	N	N
Julia	Theodora	2	N	Y
Julia	Greg	2	N	Y
Marc	Marc	0	Y	N
Marc	Theodora	1	N	Y
Marc	Greg	1	N	Y
Theodora	Theodora	0	Y	Y
Greg	Greg	0	Y	Y



# Linking the Fact and the Dimension Table



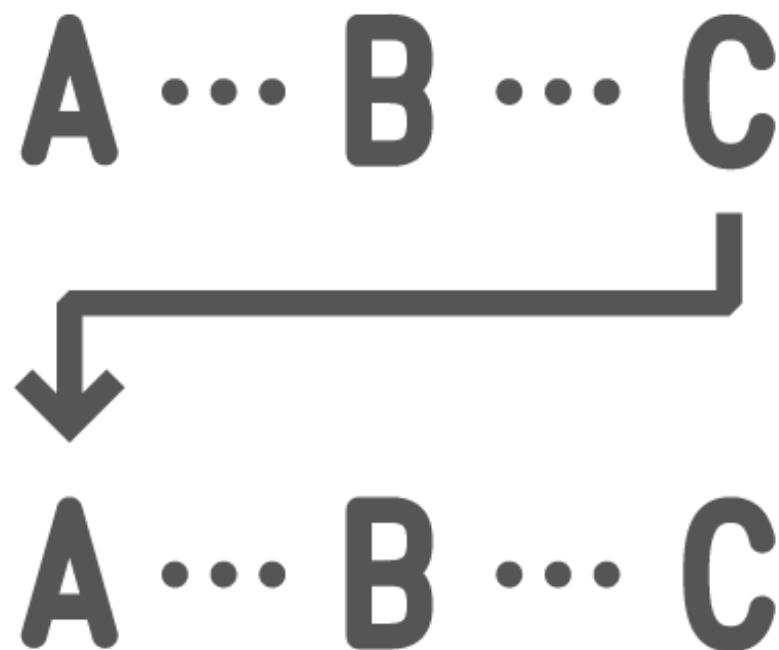
# Limitations of the Hierarchy Bridge



**Can grow a lot in size  
Performance impact  
Difficult to work with**



# Ragged Hierarchies Created with Pathstring Attributes



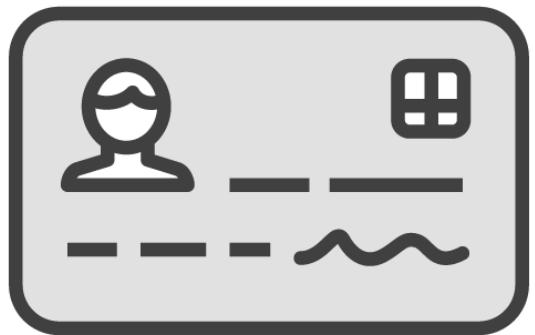
**Alternative to the bridge table**

**The pathstring attribute:**

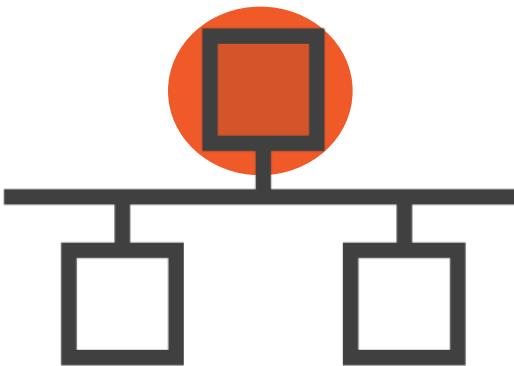
- A special attribute created in the dimension
- A string of characters
- Consists of all the parents of a member from the top of the hierarchy



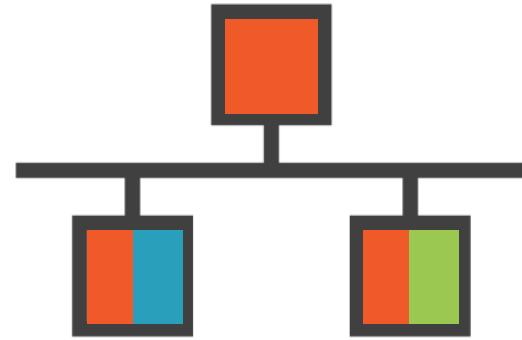
# Creating a Hierarchy with Based on Pathstring Attribute



Each node is labeled with a unique value



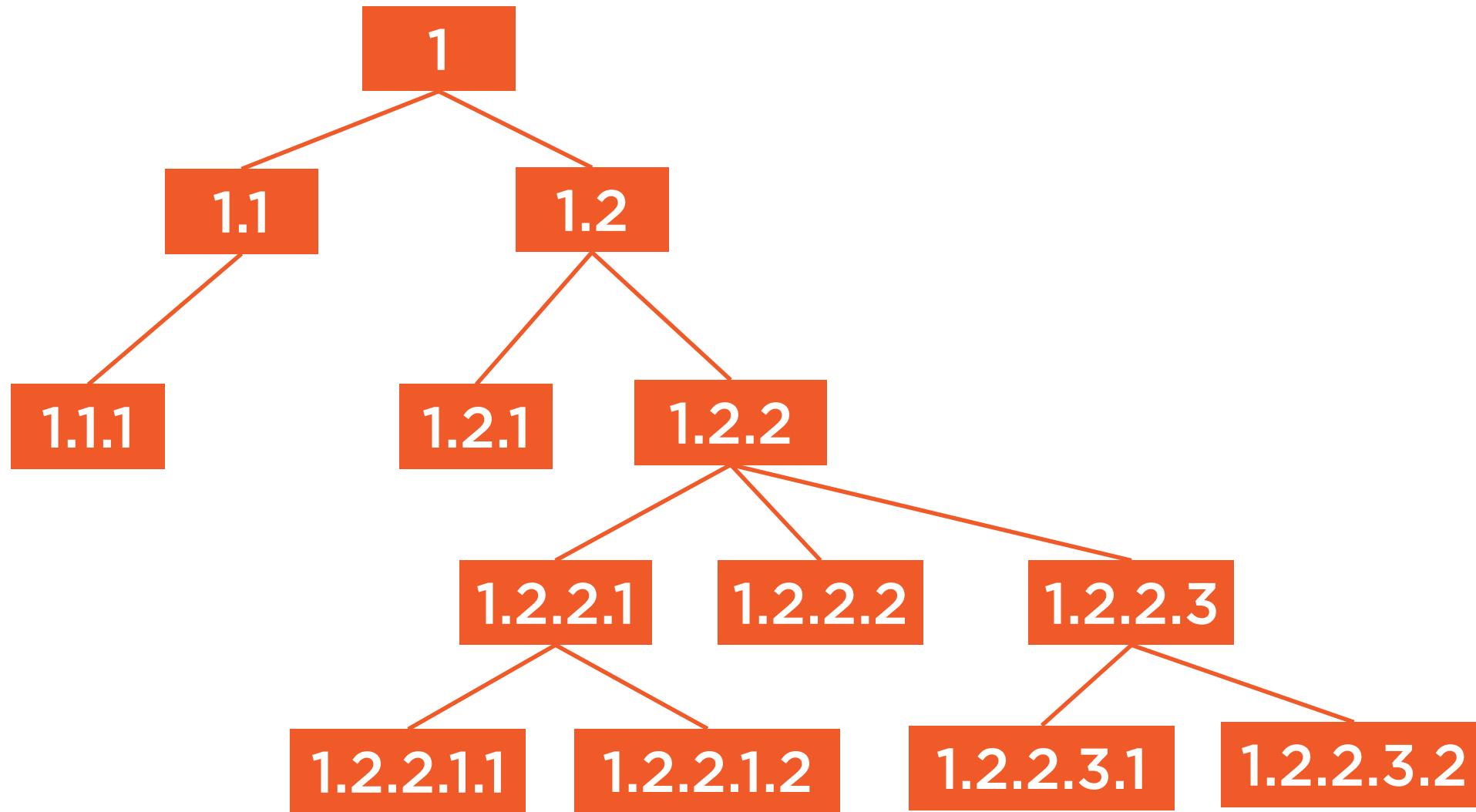
The pathstring of the root node is its unique label



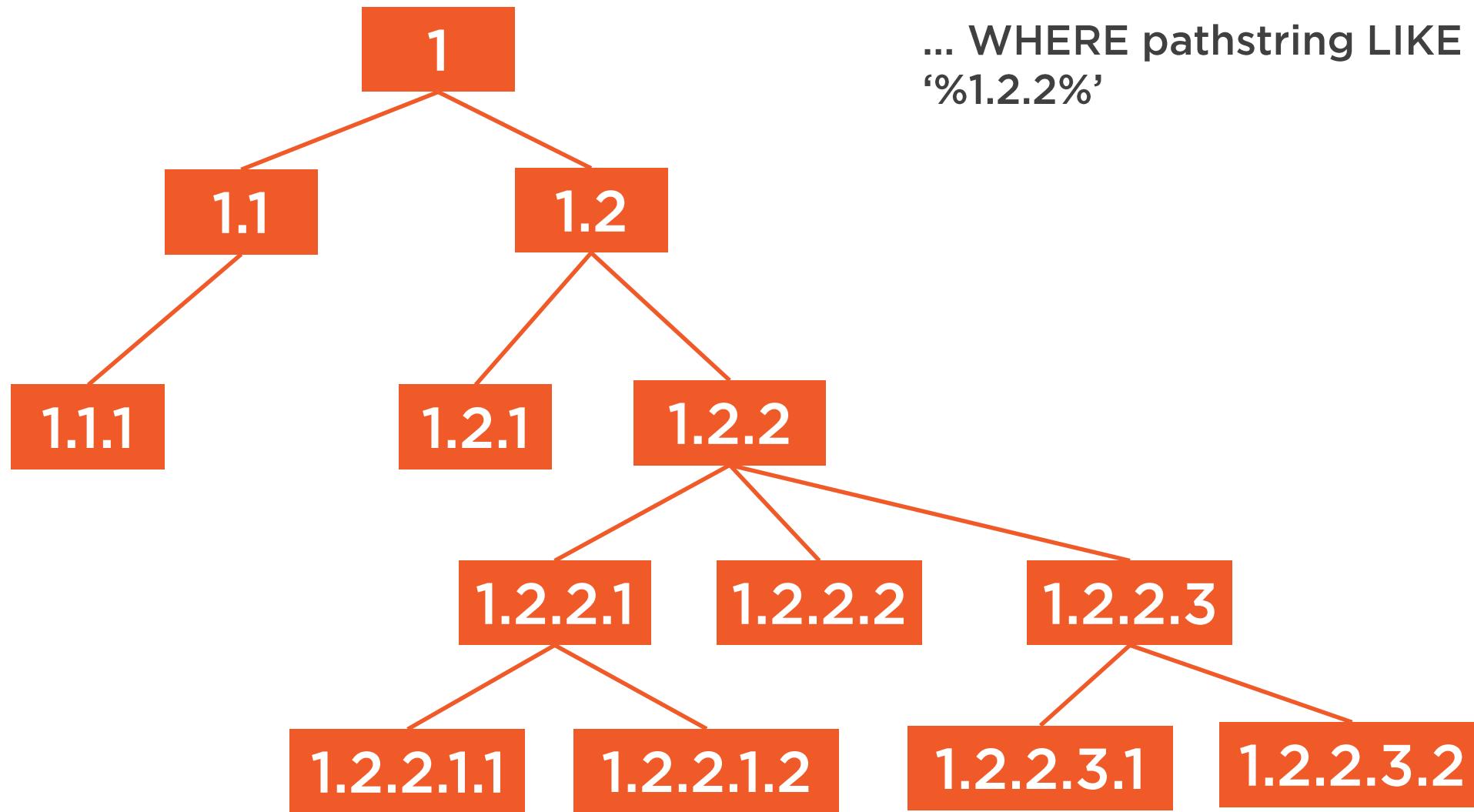
The pathstring on one level includes the pathstring of the parent



# Hierarchy Based on Pathstring Attribute



# Hierarchy Based on Pathstring Attribute



# Limitations of the Pathstring Attribute



## Vulnerable to structure changes

- If a member is moved within the organization (or added or deleted)
- Entire hierarchy must be relabeled

## Complicated to use by business users

- Accessing the database directly
- Using the 'LIKE' operator in SQL queries



# Summary



**There is no “best solution” to this problem**

**A good enough solution is generated by having a clear understanding of**

- The data available
- The business requirements



# Summary



## Hierarchy definition

- Data structures with multiple levels
- Levels are formed with attributes of a dimension

## Advantages of hierarchies

- Structure the data
- Multiple perspectives of the same data
- Easily visualize summarized or detailed data

## Types of hierarchies

- Fixed-depth
- Variable-depth
  - Slightly ragged
  - Ragged



# Implementing Fact Tables

---



**Ana Voicu**  
@ana\_voicu



# Overview



## Structure of a fact table

## The most common types of fact tables

- Transaction
- Periodic snapshot
- Accumulating snapshot

## Handling nulls in fact tables

## Demo: Creating and working with fact tables



# Structure of a Fact Table

---



# Examining a Fact Table's Purpose



# Examining a Fact Table's Purpose

Company



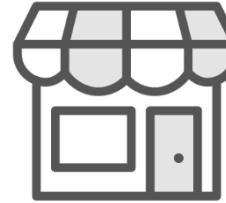
# Examining a Fact Table's Purpose

Business processes



Sales

Company



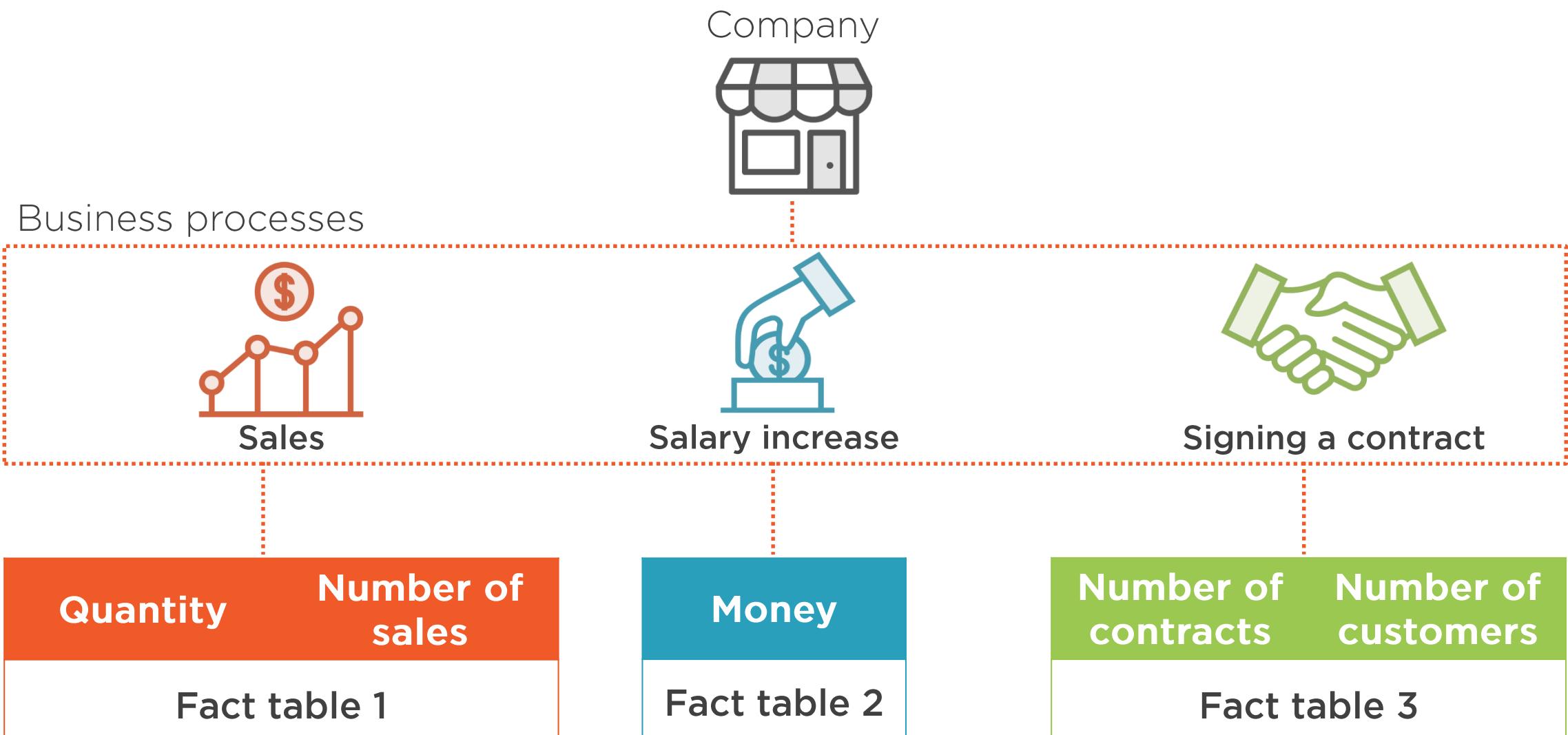
Salary increase



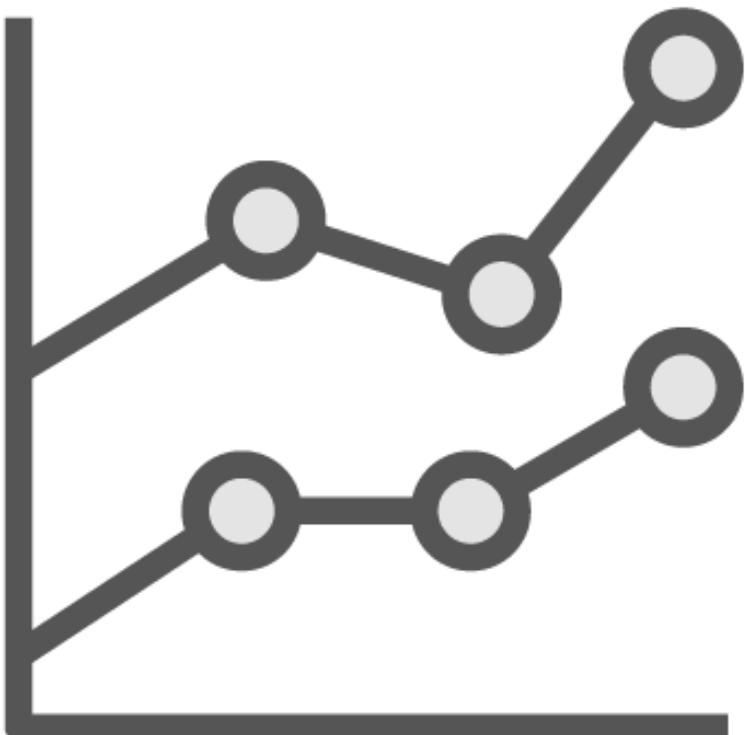
Signing a contract



# Examining a Fact Table's Purpose



# Fact Table Characteristics



**Used with dimensions for:**

- Reporting purposes
- Measuring performance
- Analyzing trends in time

**Contain a lot of measurements**

**They tend to grow very fast**

**Performance is an important design factor**



# Walmart Case Study

As of April 2019

Number of stores 11,300

Annual revenue in previous fiscal year 514 billion dollars

Days in a year 365

Avg. revenue per day 1.4 billion dollars

Avg. revenue per day 124,000 dollars per store

Avg. price of a banana 19 cents



# Walmart Case Study

As of April 2019

Number of stores 11,300

Annual revenue in previous fiscal year 514 billion dollars

Days in a year 365

Avg. revenue per day 1.4 billion dollars

Avg. revenue per day 124,000 dollars per store

Avg. price of a banana 19 cents



# Walmart Case Study

**As of April 2019**

**Number of stores** 11.300

**Annual revenue in previous fiscal year** 514 billion dollars

**Days in a year** 365

**Avg. revenue per day** 1.4 billion dollars

**Avg. revenue per day per store** 124.000 dollars

**Avg. price of a banana** 19 cents

**How many transactions are recorded per day?**

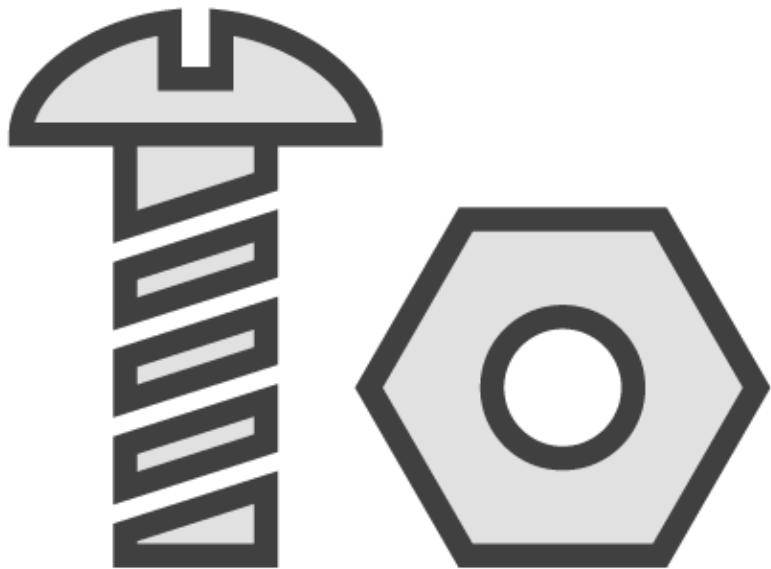
**What are the best sold products from each store?**

**Data warehouse fact tables**

- Sales fact table
- Profit and loss
- Periodic revenue per period



# Main Elements of a Fact Table



## Facts

- Columns storing the measurements of the process

## Primary key

- Uniquely identifies rows

## Foreign keys

- Linking the fact and the dimension tables



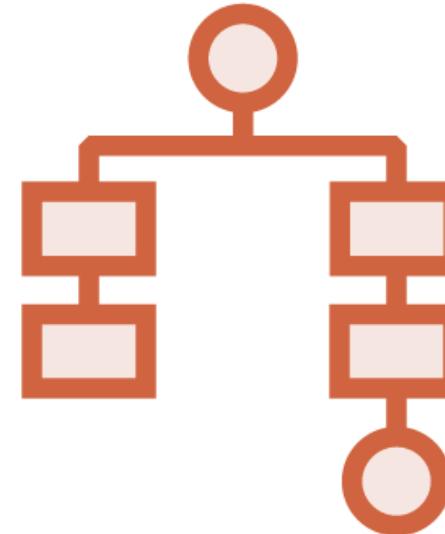
# Steps for Identifying Facts



Analyze what is important in a business process



Identify multiple facts



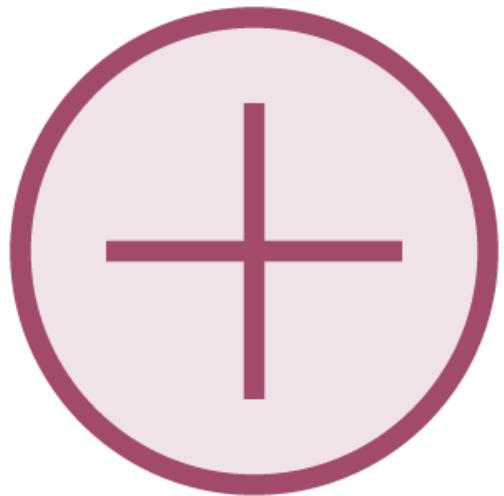
Group facts by granularity



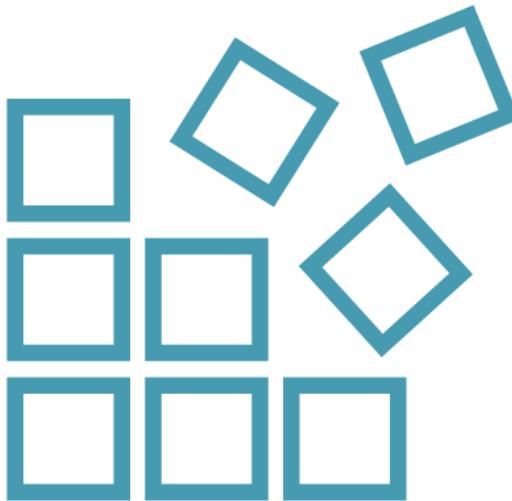
Double-check the facts



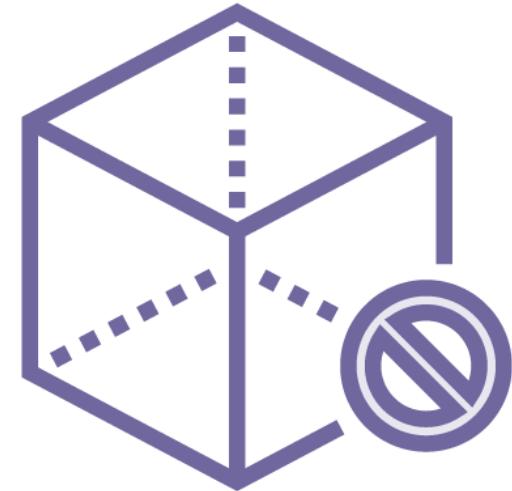
# Types of Facts



Additive

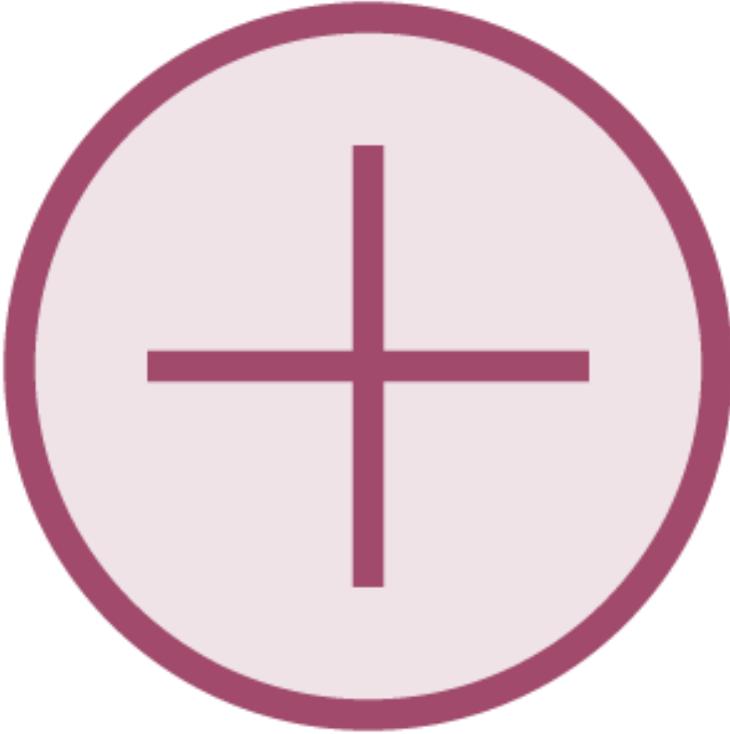


Semi-additive



Non-additive





# Additive Facts

Can be summarized across all dimensions from a fact

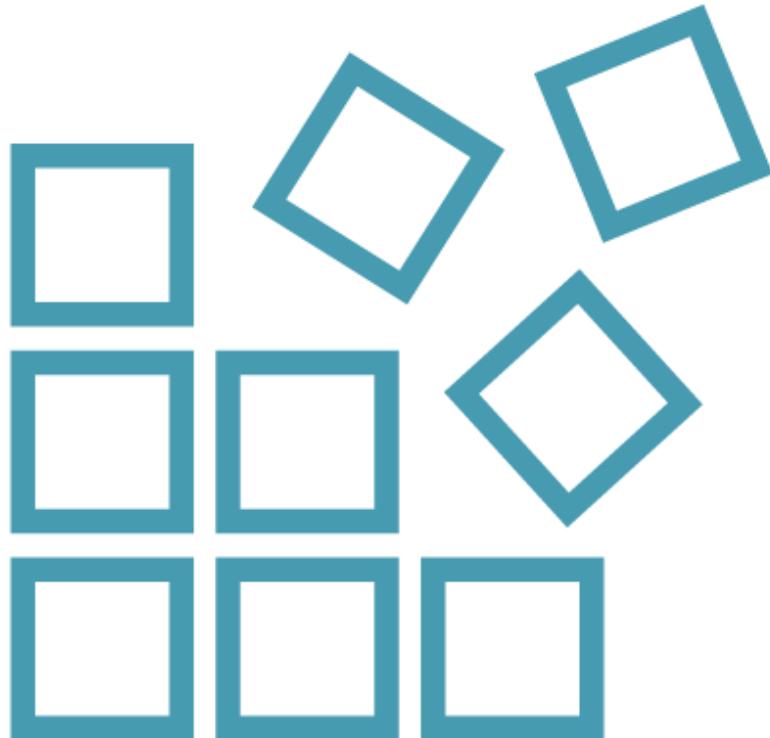
The most common type of fact

Examples:

- Sales amount
- Sales quantity
- Purchase amount
- Number of transactions



# Semi-additive Measures



Can be summarized across *some* dimensions

Cannot be summarized across time



# Example of Semi-additive Measures



# Example of Semi-additive Measures

Date key	Product key	Store key	Quantity	Inventory value at price (\$)
04-03-2019	777	1	200	400
04-04-2019	777	1	100	200



# Example of Semi-additive Measures

Date key	Product key	Store key	Quantity	Inventory value at price (\$)
04-03-2019	777	1	200	400
04-04-2019	777	1	100	200
04-03-2019	80	1	50	50
04-04-2019	80	1	25	25



# Example of Semi-additive Measures

Date key	Product key	Store key	Quantity	Inventory value at price (\$)
04-03-2019	777	1	200	400
04-04-2019	777	1	100	200
04-03-2019	80	1	50	50
04-04-2019	80	1	25	25

How many units were in the store on the 3<sup>rd</sup> of April?

What was their value?



# Example of Semi-additive Measures

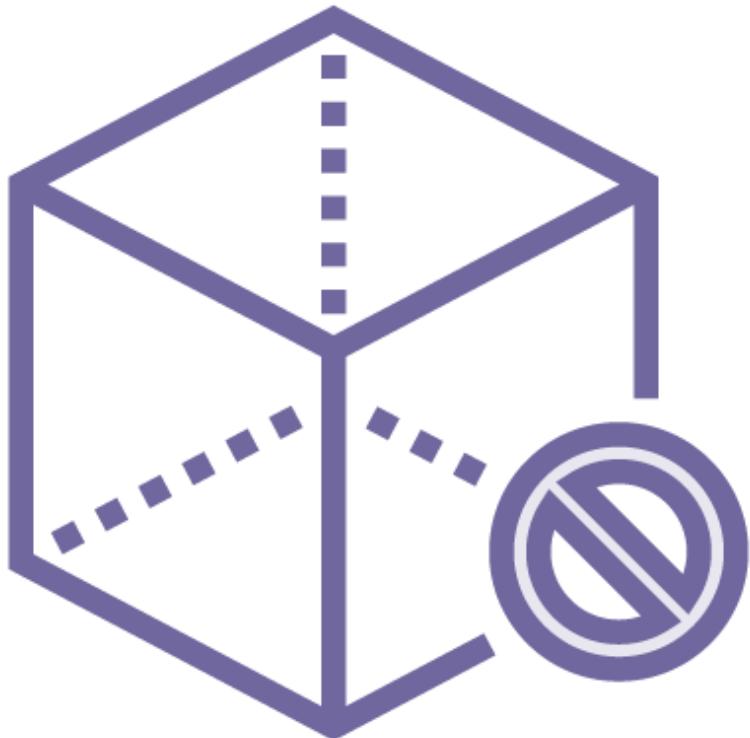
Date key	Product key	Store key	Quantity	Inventory value at price (\$)
04-03-2019	777	1	200	400
04-04-2019	777	1	100	200
04-03-2019	80	1	50	50
04-04-2019	80	1	25	25

## Examples of semi-additive measures:

- Inventory information
- Financial account balances
- Water levels on rivers
- Temperature



# Non-additive Measures



Ratios or percentages  
Cannot be summarized across any dimension



# Example of Non-additive Measures



# Example of Non-additive Measures

Date key	Product key	Store key	Initial quantity	Removed quantity
04-03-2019	777	1	200	100
04-04-2019	777	1	100	40
04-03-2019	80	1	50	25
04-04-2019	80	1	25	9



# Example of Non-additive Measures

Date key	Product key	Store key	Initial quantity	Removed quantity	Removed quantity (%)
04-03-2019	777	1	200	100	50
04-04-2019	777	1	100	40	40
04-03-2019	80	1	50	25	50
04-04-2019	80	1	25	9	36

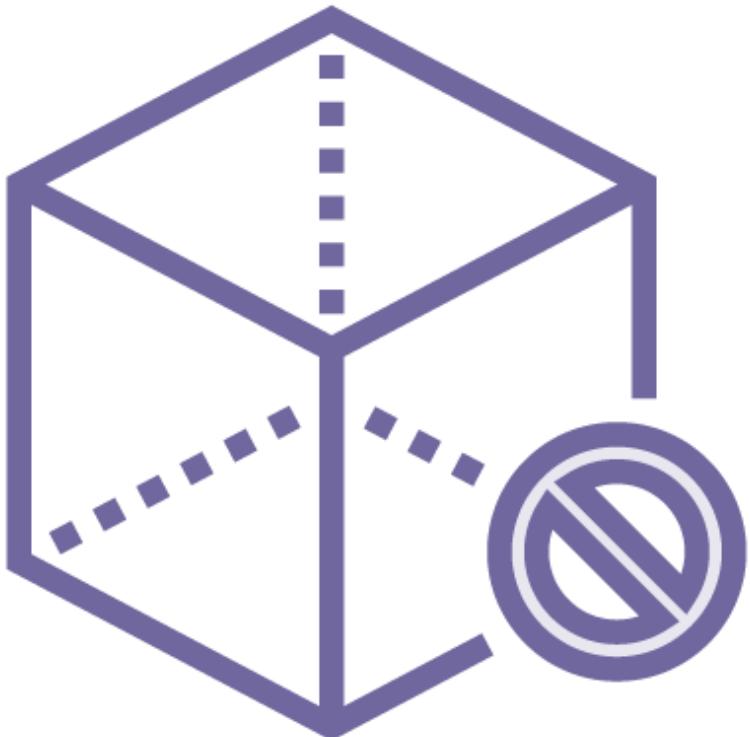


# Example of Non-additive Measures

Date key	Product key	Store key	Initial quantity	Removed quantity	Removed quantity (%)	Unit price
04-03-2019	777	1	200	100	50	2
04-04-2019	777	1	100	40	40	2
04-03-2019	80	1	50	25	50	1
04-04-2019	80	1	25	9	36	1



# Non-additive Measures



**Ratios or percentages**

**Cannot be summarized across any dimension**

**Why are non-additive measures considered facts?**

- Design preferences
  - Storing them in fact tables vs. in dimension tables, as attributes
- They are calculated based on columns from the fact table

**They should be used with care by users**



# Keys in a Fact Table

---



# Identifying the Keys



## Primary key

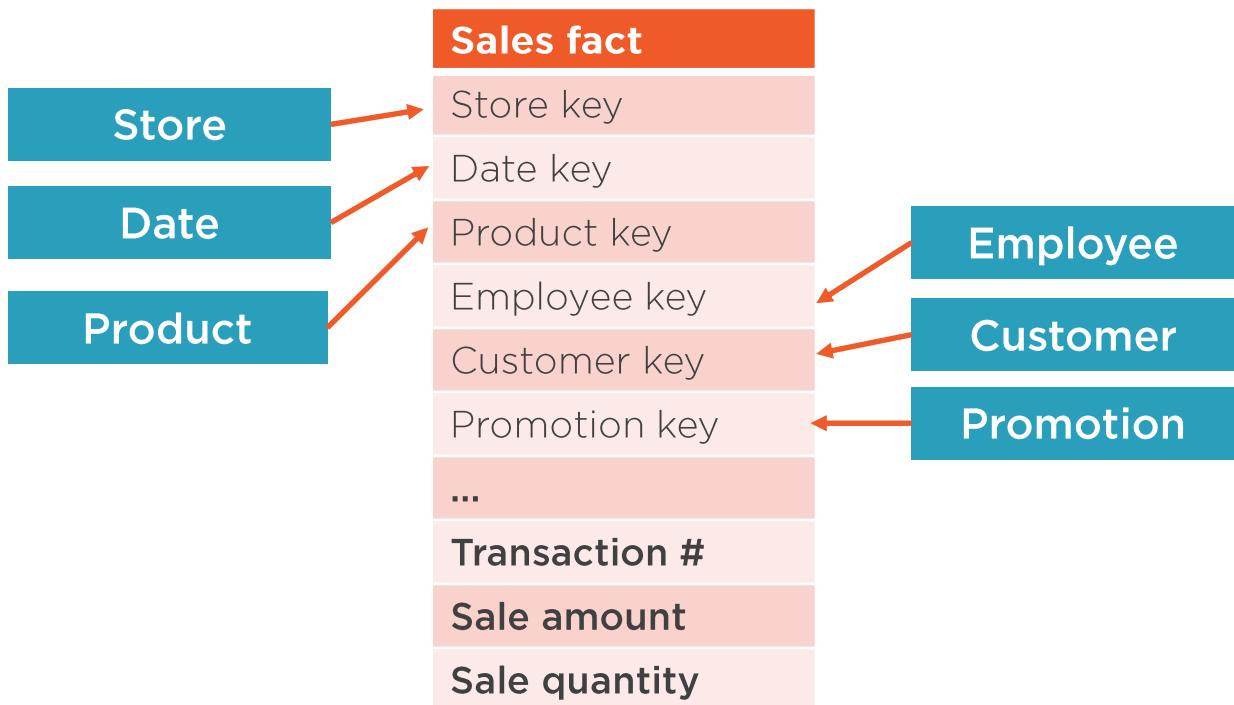
- Composite key
- Surrogate key

## Foreign keys for dimensions

# Identifying the Primary Key



# Identifying the Primary Key



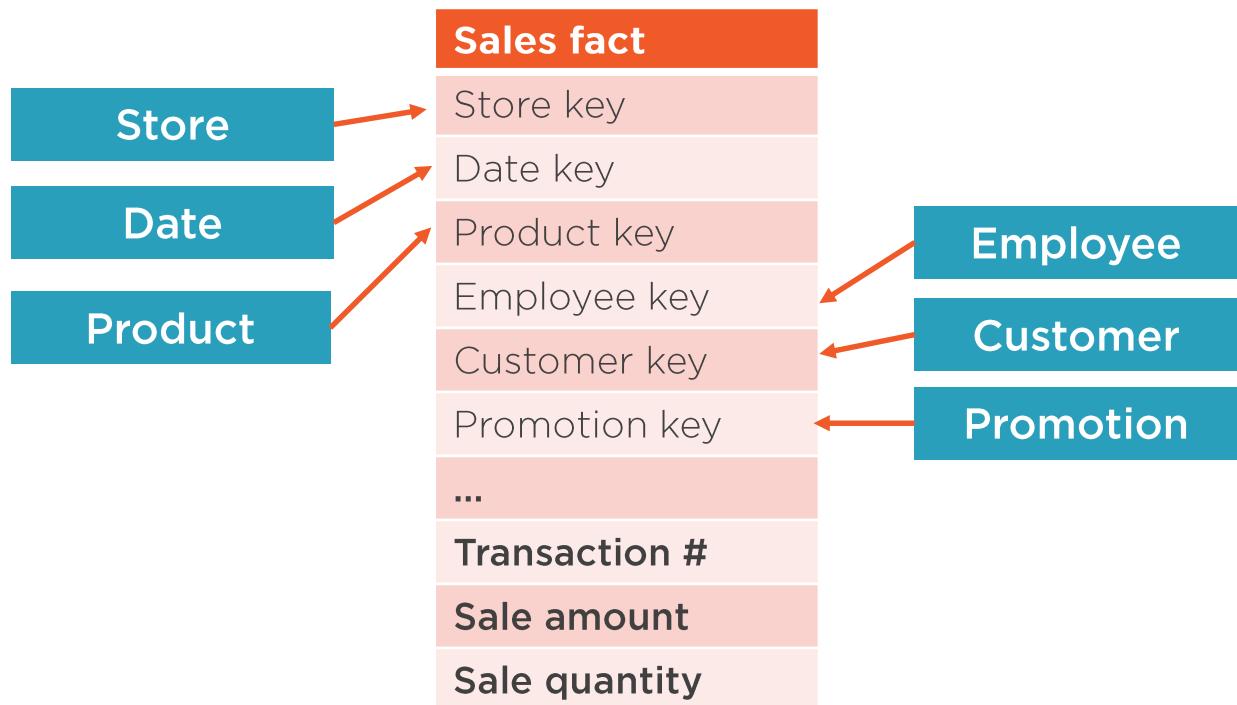
**Relationships between fact and dimension are many-to-many**

## Examples

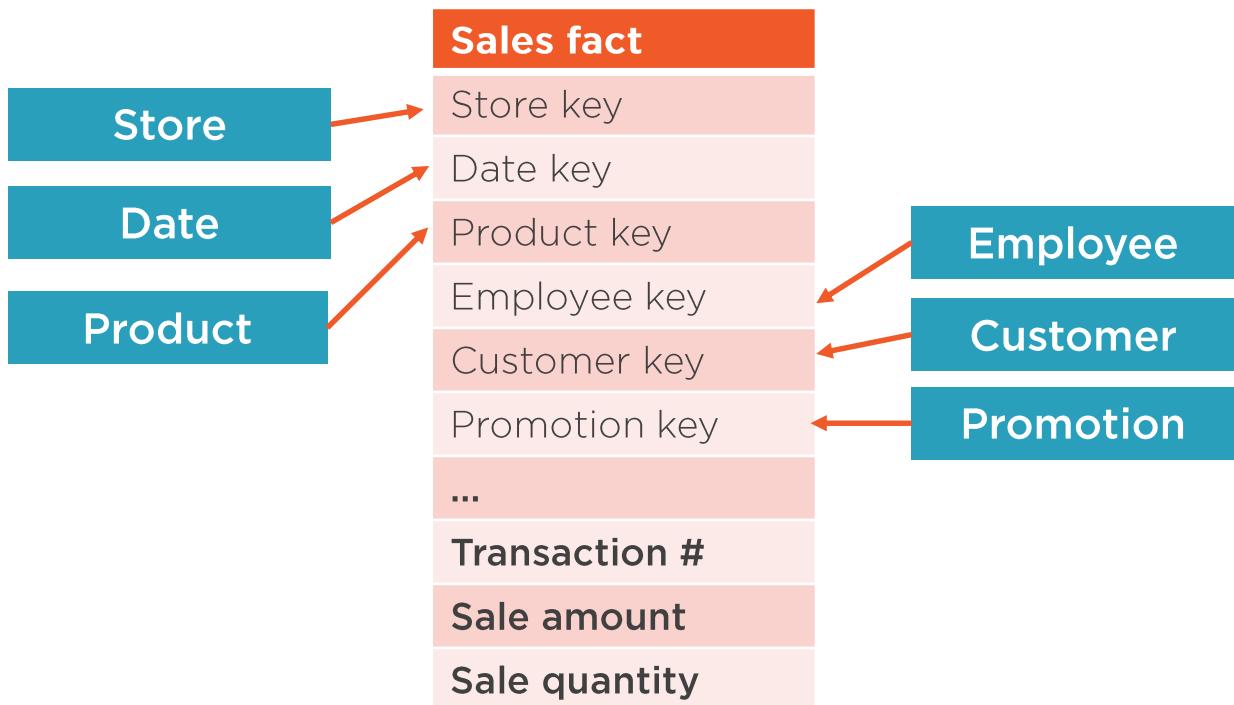
- 1.a. One sale can have multiple products
- 1.b. One product can be part of multiple sales
  
- 2.a. One sale can have multiple promotions applied
- 2.b. One promotion can be applied to many sales



# Identifying the Primary Key



# Identifying the Primary Key



1. Combine a subset of the foreign keys

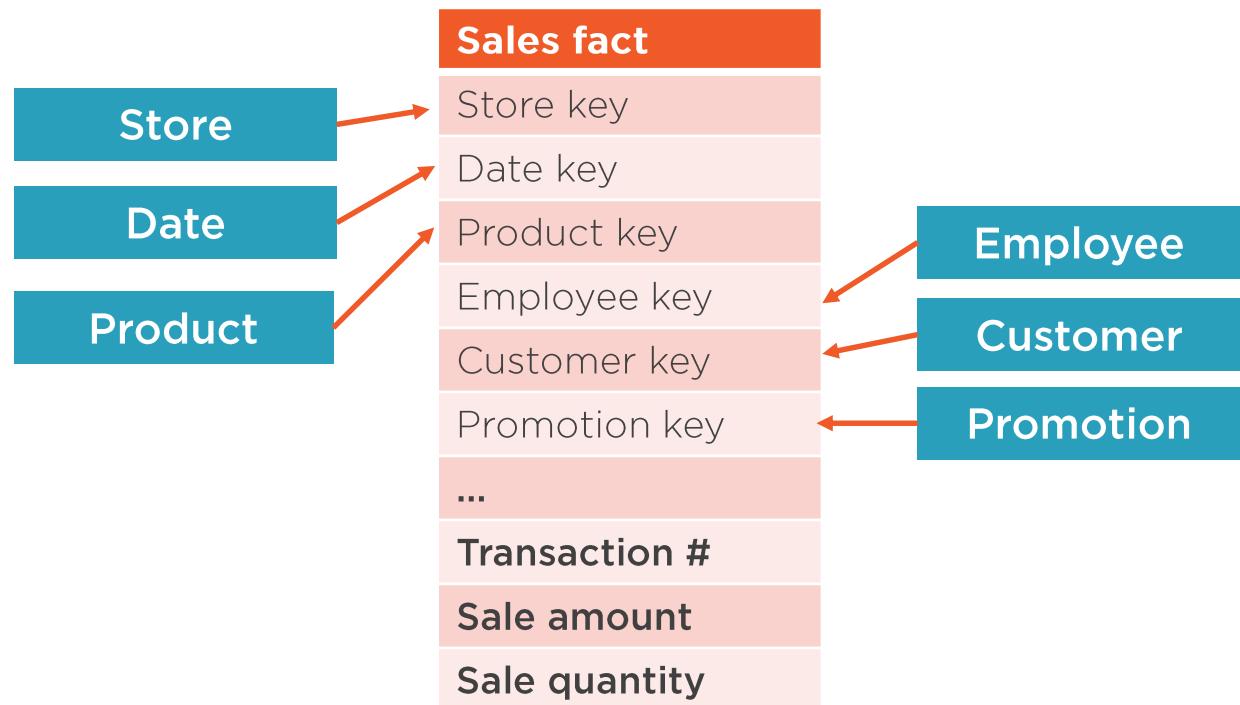
Also called

- Composite key
- Compound key
- Concatenated key

The combination must ensure row uniqueness



# Identifying the Primary Key



## 1. Combine a subset of the foreign keys

Also called

- Composite key
- Compound key
- Concatenated key

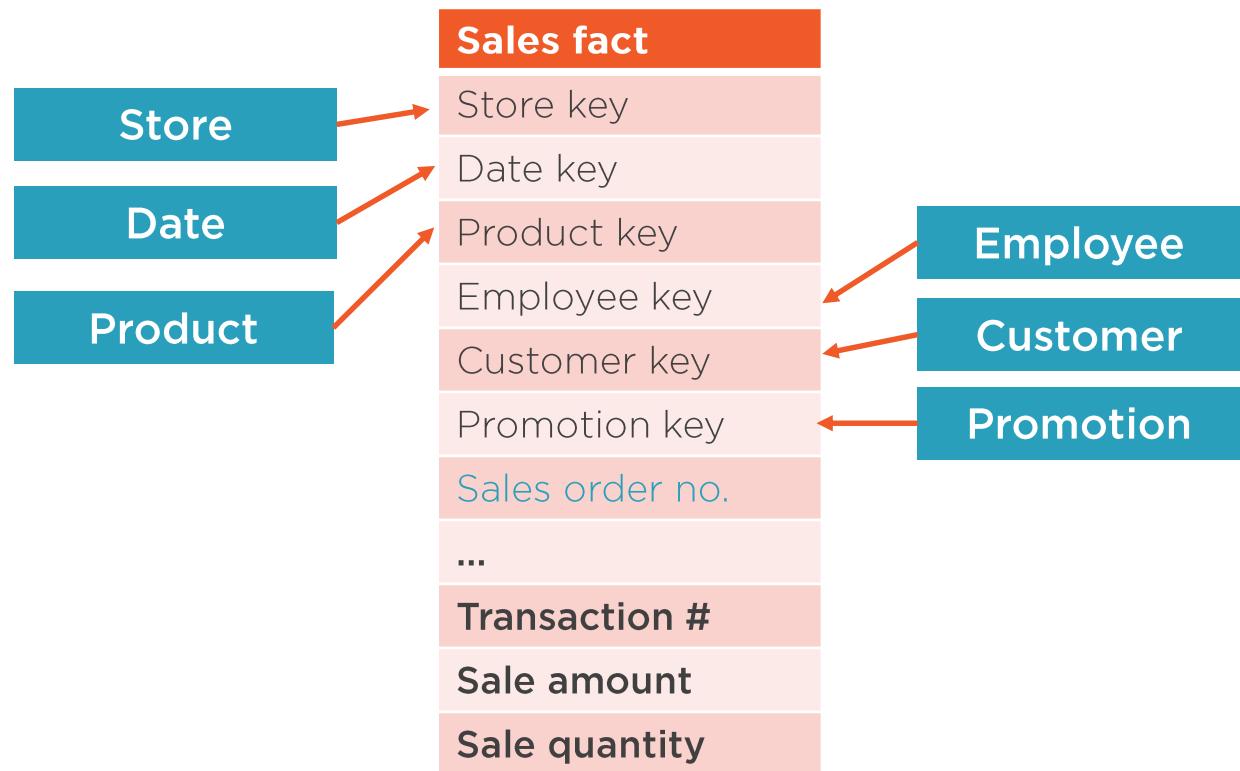
The combination must ensure row uniqueness

## 2. Use another column from the source system: the primary key for the transactions

Merge the new column to the subset of FKS



# Identifying the Primary Key



## 1. Combine a subset of the foreign keys

Also called

- Composite key
- Compound key
- Concatenated key

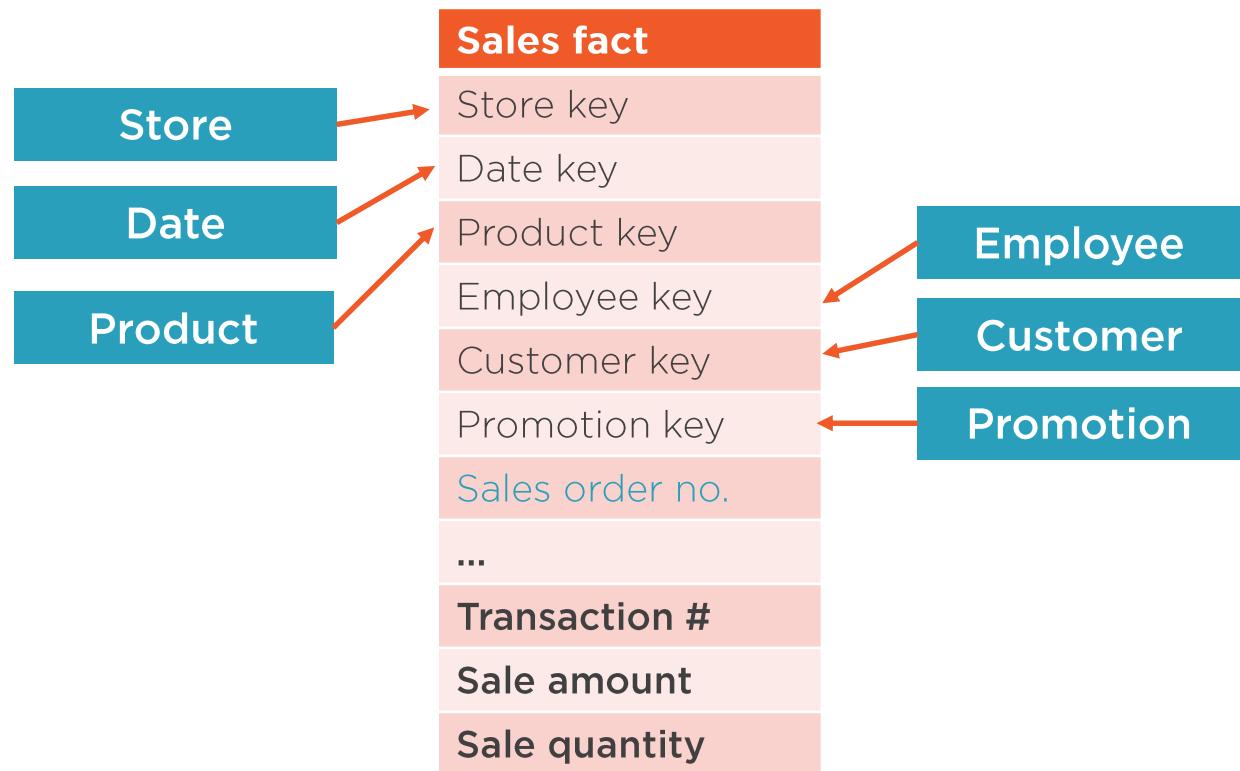
The combination must ensure row uniqueness

## 2. Use another column from the source system: the primary key for the transactions

Merge the new column to the subset of FKS



# Identifying the Primary Key



## 1. Combine a subset of the foreign keys

Also called

- Composite key
- Compound key
- Concatenated key

The combination must ensure row uniqueness

## 2. Use another column from the source system: the primary key for the transactions

Merge the new column to the subset of Fks

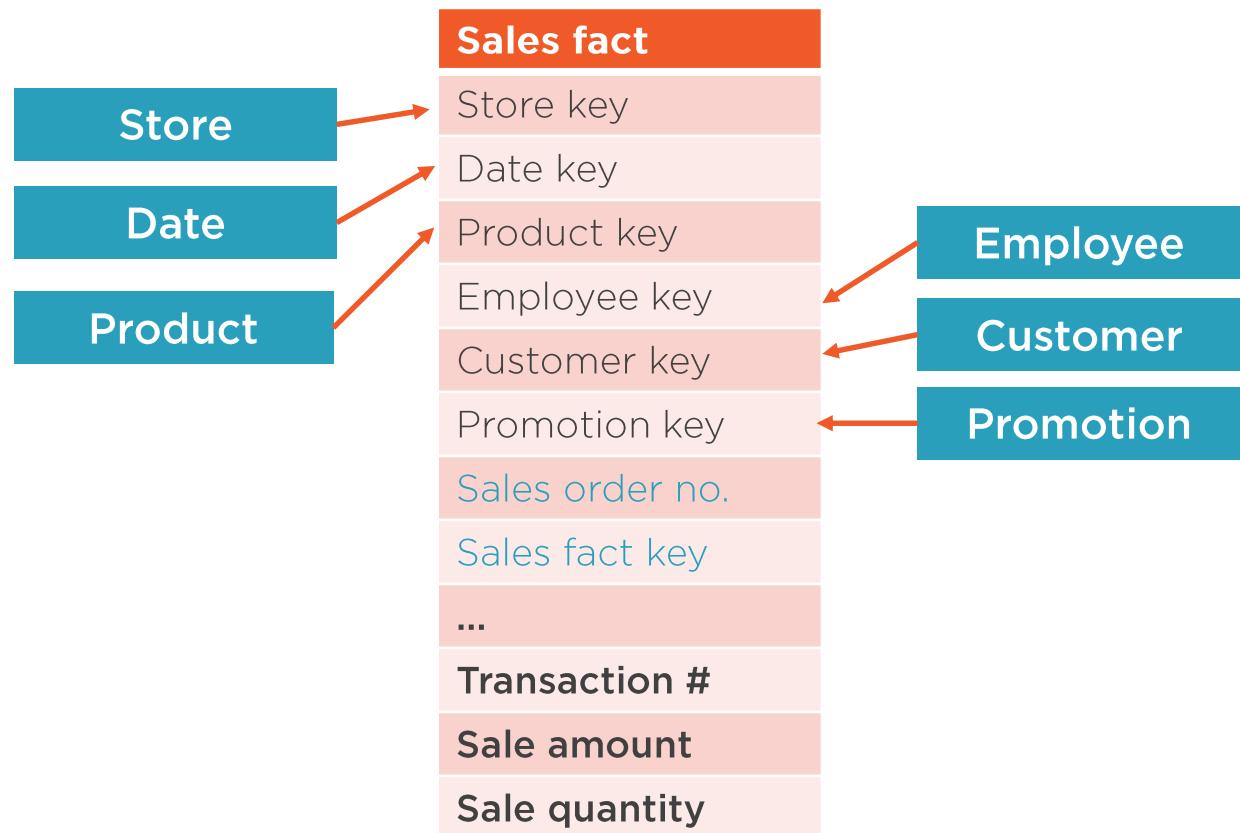
## 3. Create a surrogate key

- Small integer
- Auto-incremented
- No business value

The Fks don't play a special role anymore



# Identifying the Primary Key



## 1. Combine a subset of the foreign keys

Also called

- Composite key
- Compound key
- Concatenated key

The combination must ensure row uniqueness

## 2. Use another column from the source system: the primary key for the transactions

Merge the new column to the subset of FKS

## 3. Create a surrogate key

- Small integer
- Auto-incremented
- No business value

The FKS don't play a special role anymore



# Advantages of Using a Surrogate Key



**Easier to identify a row in the fact table**

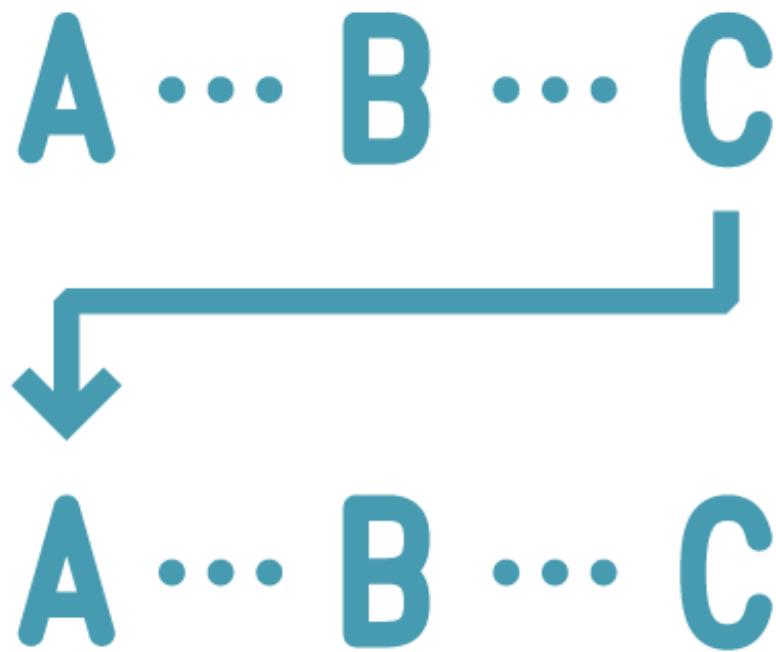
**Debugging a load that stopped midway**

**Transforming an update into a combination  
of**

- Inserts
- Deletes



# Creating the Foreign Keys



**Each dimension has**

- A surrogate key (PK)
- A business key (PK of the source system)

**A fact table has**

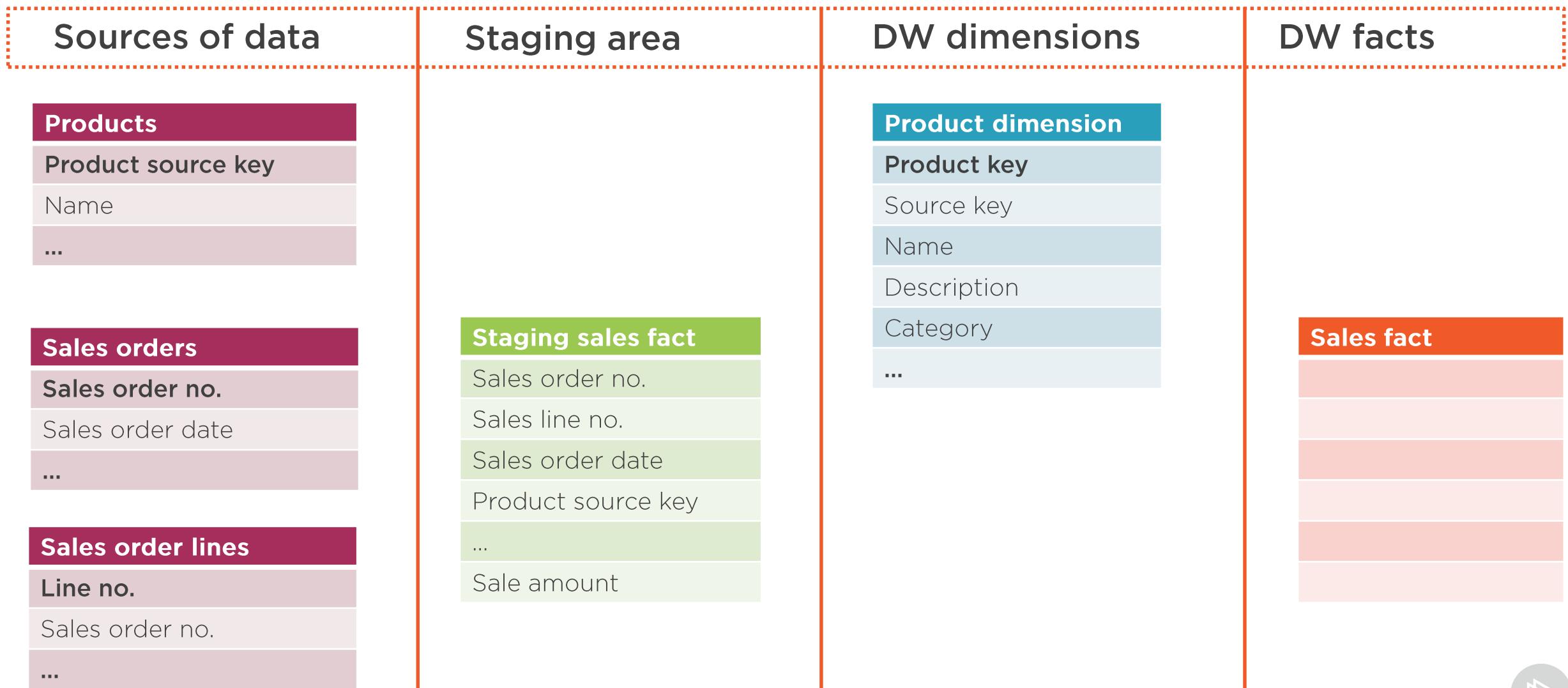
- A primary key
- Foreign keys to its dimensions, linked to the dimensions' surrogate key

**Relationship between the fact and the dimension**

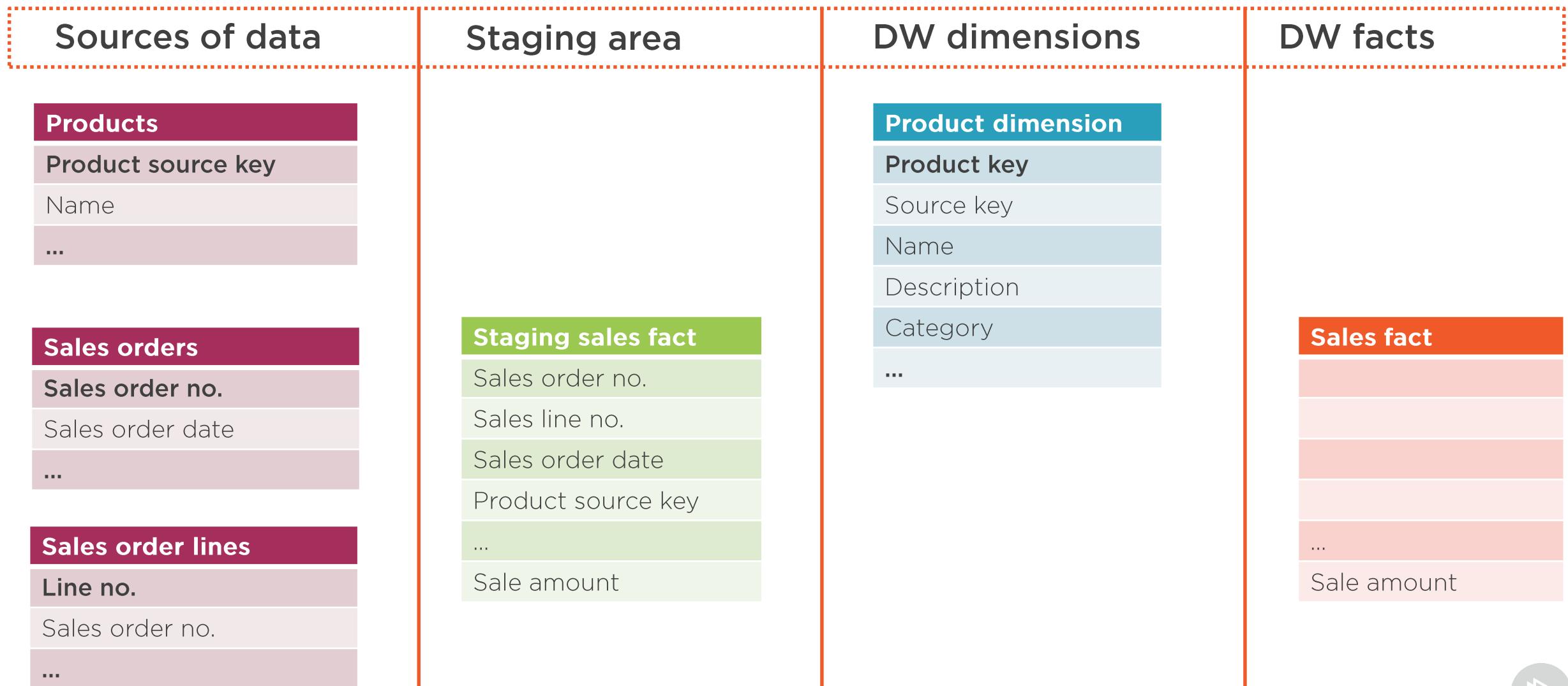
- The business key from the dimension
- An intermediary (staging) fact table
  - Fact table data is joined with dimension table data on the real join conditions
  - Staging table stores the PKs from the source tables for dimensions



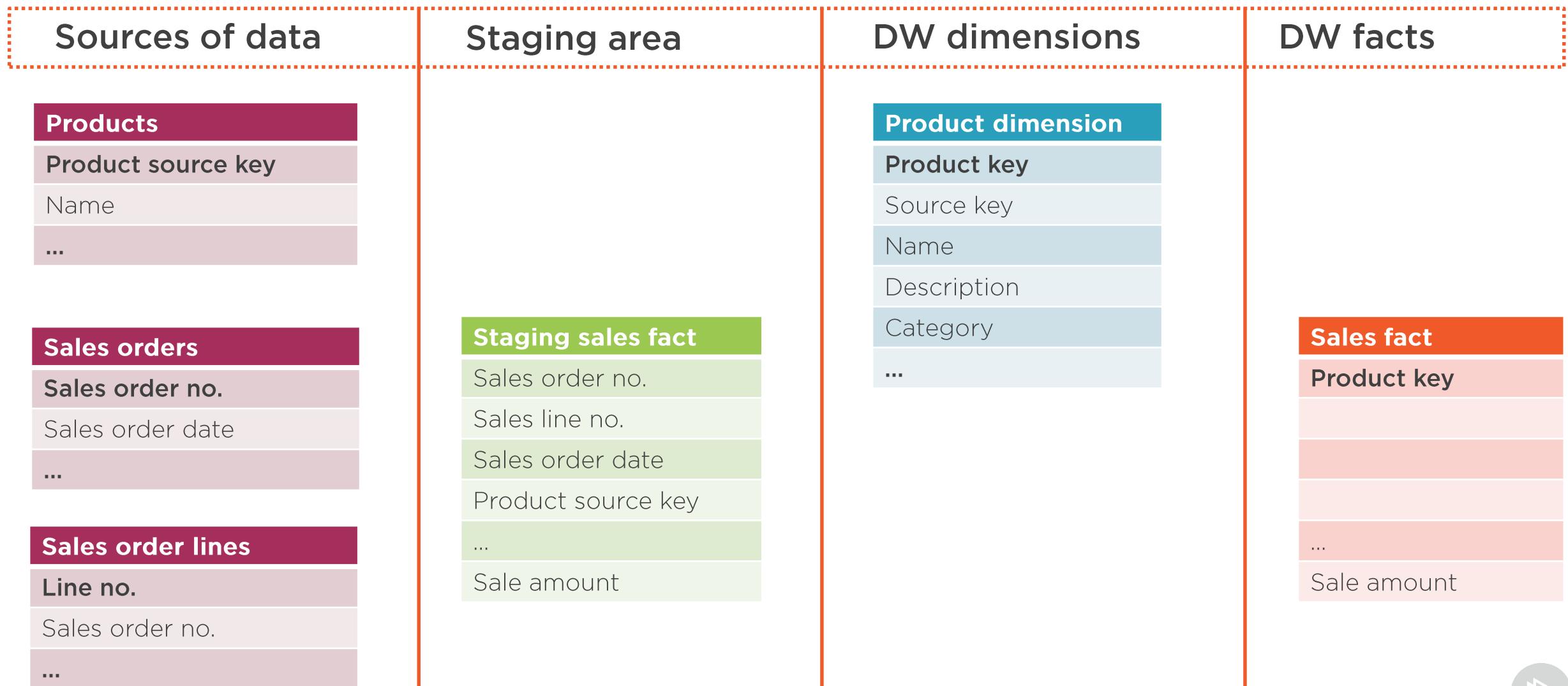
# Creating the Foreign Keys



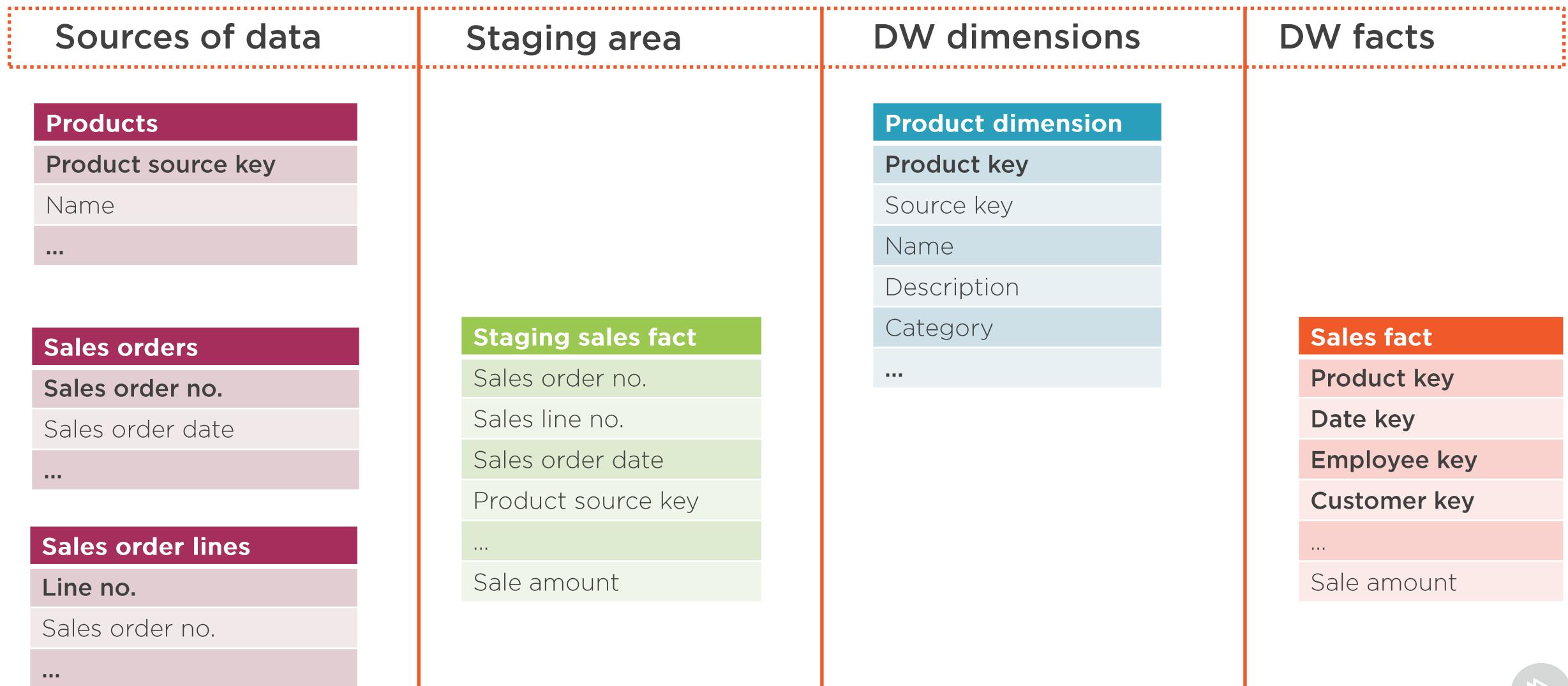
# Creating the Foreign Keys



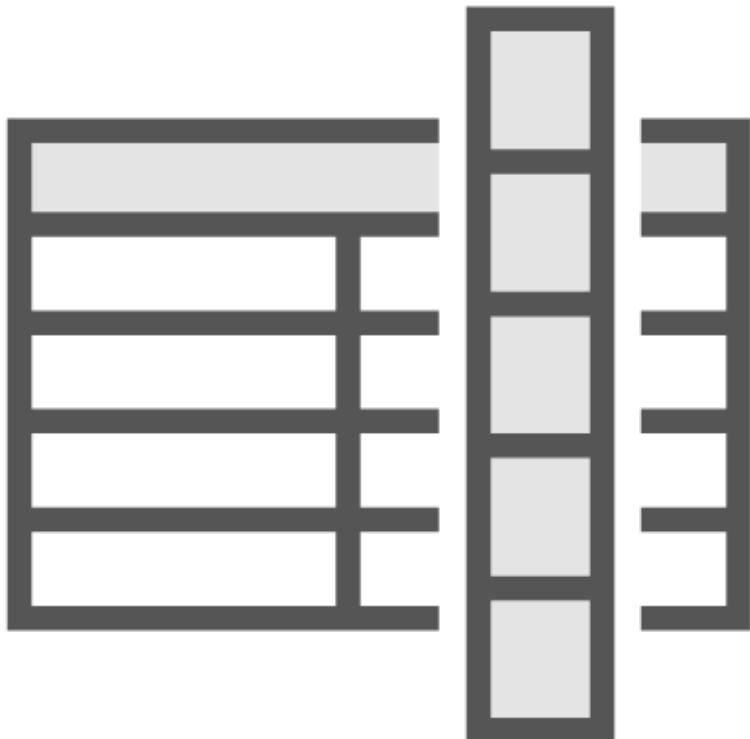
# Creating the Foreign Keys



# Creating the Foreign Keys



# Degenerate Dimensions



## Columns added to fact tables

- Different than facts
- Different than keys

Similar to foreign keys, but don't point to any dimension

## Examples:

- Sales order number
- Invoice number
- Other transaction numbers

## Advantages of degenerate dimensions

- Usually part of the fact's PK
- Group all rows that were part of the same transaction
- Track data back to the operational system

Storing too many may impact the performance of the fact table

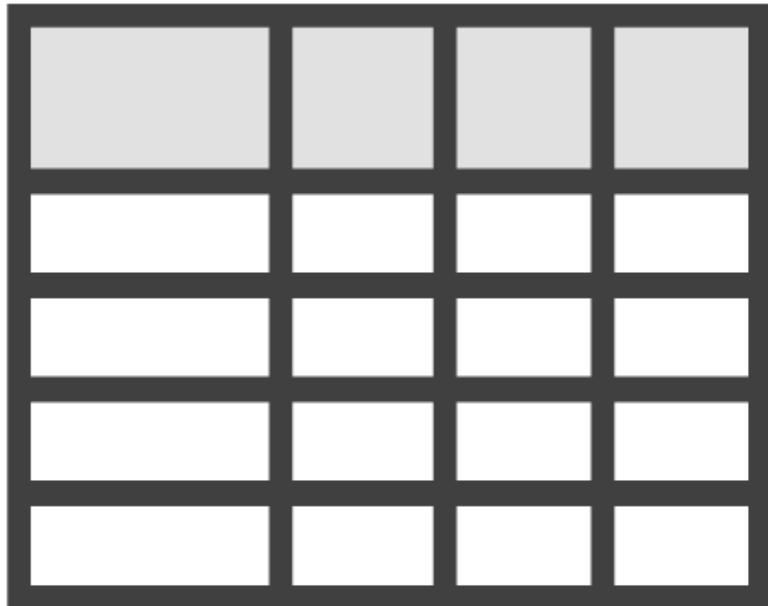


# Types of Fact Tables

---



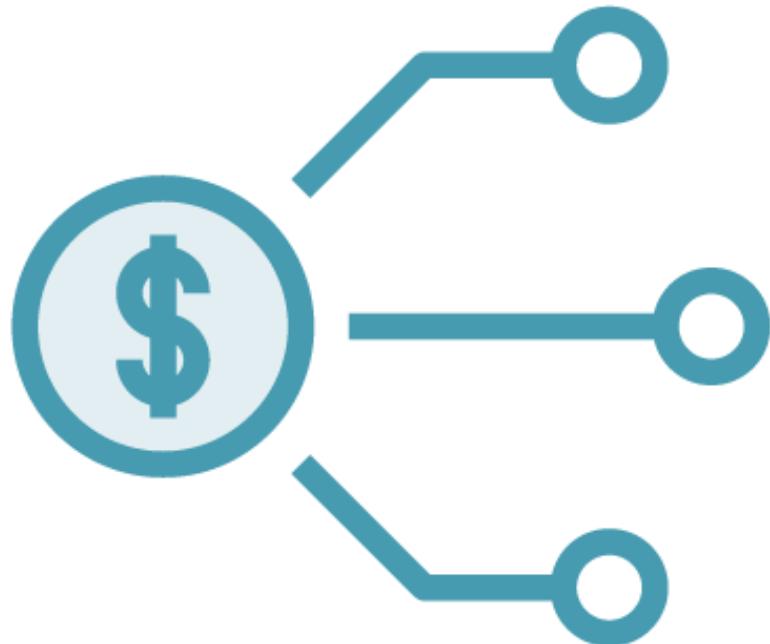
# Most Common Types of Fact Tables



**Transaction fact tables**  
**Periodic snapshot**  
**Accumulating snapshot**



# Transaction Fact Tables



The most common type of fact table in a data warehouse

Data inside it represents a process that happened at a certain point in time

A row exists for a dimension member only if the member was involved in a transaction



# Properties of Transaction Fact Tables

The granularity is the transaction or transaction line

Tendency to become very large

Relationships with many dimensions

Sparingly populated

The facts are additive



# Periodic Snapshot Fact Tables



**Show information as it was at the end of a time interval**

- End of each day
- End of each month, etc.

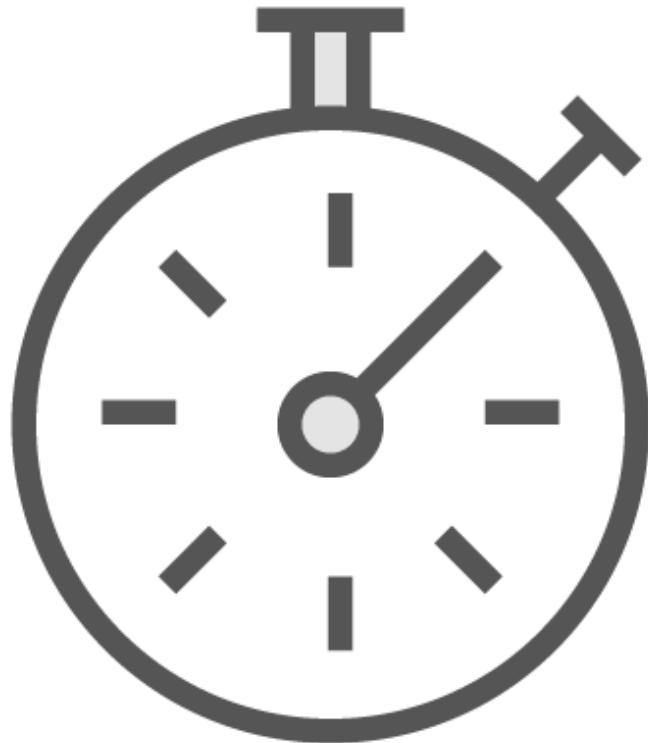
**Are not periodically refreshed**

- The new snapshots are added on top of the existing ones
- Are useful for analyzing trends in time

**Transactional fact tables can be a starting point for the periodic snapshots**



# Accumulating Snapshot Fact Tables



**Store information about a process that has**

- A clear beginning
- A clear end
- A number of intermediary steps

**Example: pipeline or workflow processes**

- Fulfilling an order
- Progress of a mortgage application
- Status of a support ticket



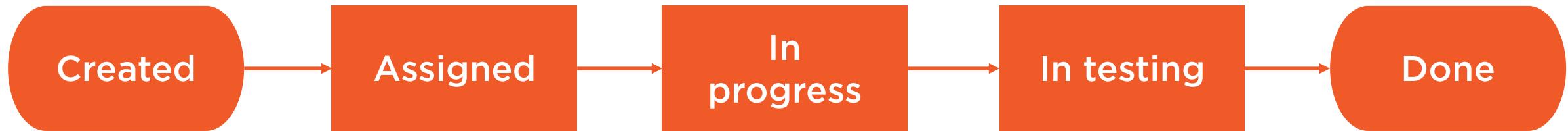
# Accumulating Snapshot Fact Tables

**Handling support calls**



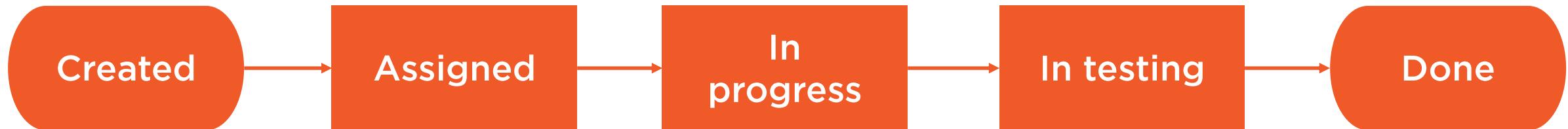
# Accumulating Snapshot Fact Tables

Handling support calls



# Accumulating Snapshot Fact Tables

Handling support calls

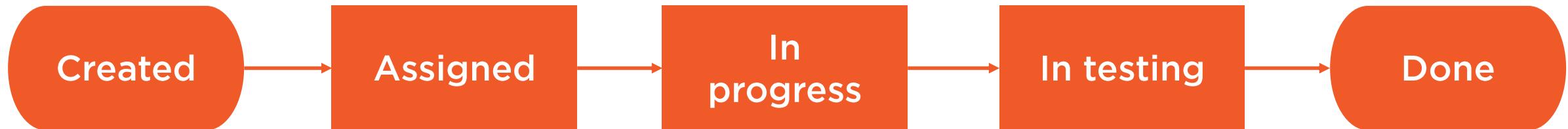


Created date key	Assigned date key	In progress date key	In testing date key	Done date key
20190404	20190405	20190406	20190412	20190419



# Accumulating Snapshot Fact Tables

Handling support calls



Created date key	Assigned date key	In progress date key	In testing date key	Done date key
20190404	20190405	20190406	20190412	20190419
20190407	17530101	17530101	17530101	17530101



# Accumulating Snapshot Fact Tables

Handling support calls



Created date key	Assigned date key	In progress date key	In testing date key	Done date key
20190404	20190405	20190406	20190412	20190419
20190407	20190407	17530101	17530101	17530101



# Accumulating Snapshot Fact Tables

Handling support calls



Created date key	Assigned date key	In progress date key	In testing date key	Done date key
20190404	20190405	20190406	20190412	20190419
20190407	20190407	20190407	17530101	17530101



# Accumulating Snapshot Fact Tables

Handling support calls



Created date key	Assigned date key	In progress date key	In testing date key	Done date key	Duration development	Duration testing
20190404	20190405	20190406	20190412	20190419	6	3
20190407	20190407	20190407	17530101	17530101	NULL	NULL



# Accumulating Snapshot Fact Tables

Handling support calls



Created date key	Assigned date key	In progress date key	In testing date key	Done date key	Duration development	Duration testing	Is done
20190404	20190405	20190406	20190412	20190419	6	3	1
20190407	20190407	20190407	17530101	17530101	NULL	NULL	0



# Accumulating Snapshot Fact Tables

Handling support calls



Created date key	Assigned date key	In progress date key	In testing date key	Done date key	Duration development	Duration testing	Is done	Current status key
20190404	20190405	20190406	20190412	20190419	6	3	1	6
20190407	20190407	20190407	17530101	17530101	NULL	NULL	0	4



# Handling Null Values in Fact Tables

## Nulls in the facts

It is alright to have nulls in the facts

The aggregate functions can handle null values

Replacing null with zeros would influence the calculations performed

Value
2
2
null

Average: 2

Value
2
2
0

Average: 1

## Nulls in the foreign keys

Will produce a referential integrity violation error

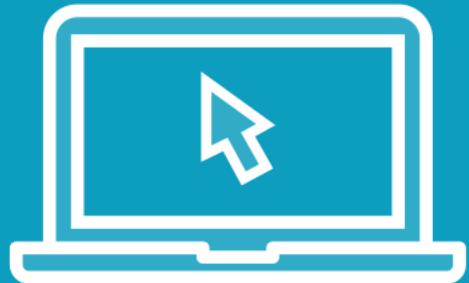
This situation must be avoided

Use the “empty row” technique:

- Every dimension should have an empty row, with its own primary key
- When there is no link between fact and dimension, the foreign key column will store the PK of the empty row



## Demo



### Creating and working with a transaction fact table

- Populate staging table with source data
- Populate fact table with data from staging
- Use the fact table in reports



# Summary



## Components of a fact table:

- Facts (or numeric measurements)
  - Additive
  - Semi-additive
  - Non-additive
- The primary key
  - Surrogate key
  - Composite key
- Foreign keys for dimensions
- Degenerate dimensions

## The most common types of fact tables:

- Transaction fact table
- Periodic snapshot
- Accumulating snapshot

## Be aware of the null values in fact tables

- Measures can have null values
- Foreign key columns must not have nulls



# Loading Data in a Data Warehouse

---



**Ana Voicu**  
@ana\_voicu



# Review of the Previous Steps



## Discuss with the business people

- Processes
- Analysis needs

## Investigate data sources

## Implement the dimensional model

- End result will be the physical design
- Fact and dimension tables are defined

## Populate the data warehouse tables with source data

- Also called “loading” the data warehouse
- Implement the ETL system



# The ETL system



**Similar to a “black box” for the data warehouse users**

**Data is:**

- Integrated from multiple sources
- Transformed into meaningful reports
- Helping people make decisions

**The ETL should be properly designed and implemented**



# Overview



## What does ETL mean?

- Extraction
- Transformation
- Load

## Types of loads

- Full (initial) load
- Incremental load

## Data lineage

- What is data lineage?
- Why is it important?
- How can you implement it?



# Demos



**Perform the needed steps to load a dimension table**

**Create and work with the auxiliary objects involved in a load**

- Staging tables
- Lineage table
- Incremental loads table



# Overview of an ETL System

---



# What Is ETL?



**A system that incorporates all operations performed on data**

- Since it's selected from data sources
- Until it reaches the presentation area

**Consists of three phases**

- Extraction (E)
- Transformation (T)
- Loading (L)



# Overview of an ETL System



# Overview of an ETL System

## Sources of data

Products



Subcategories



Categories



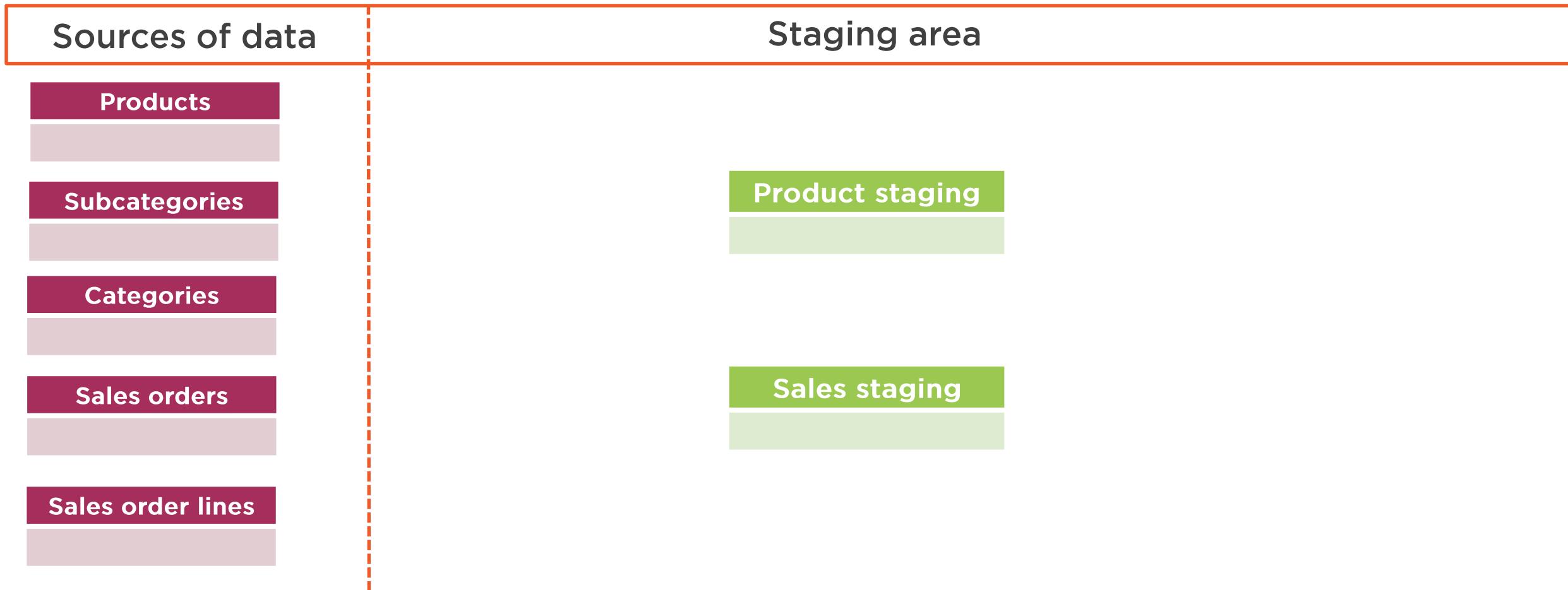
Sales orders



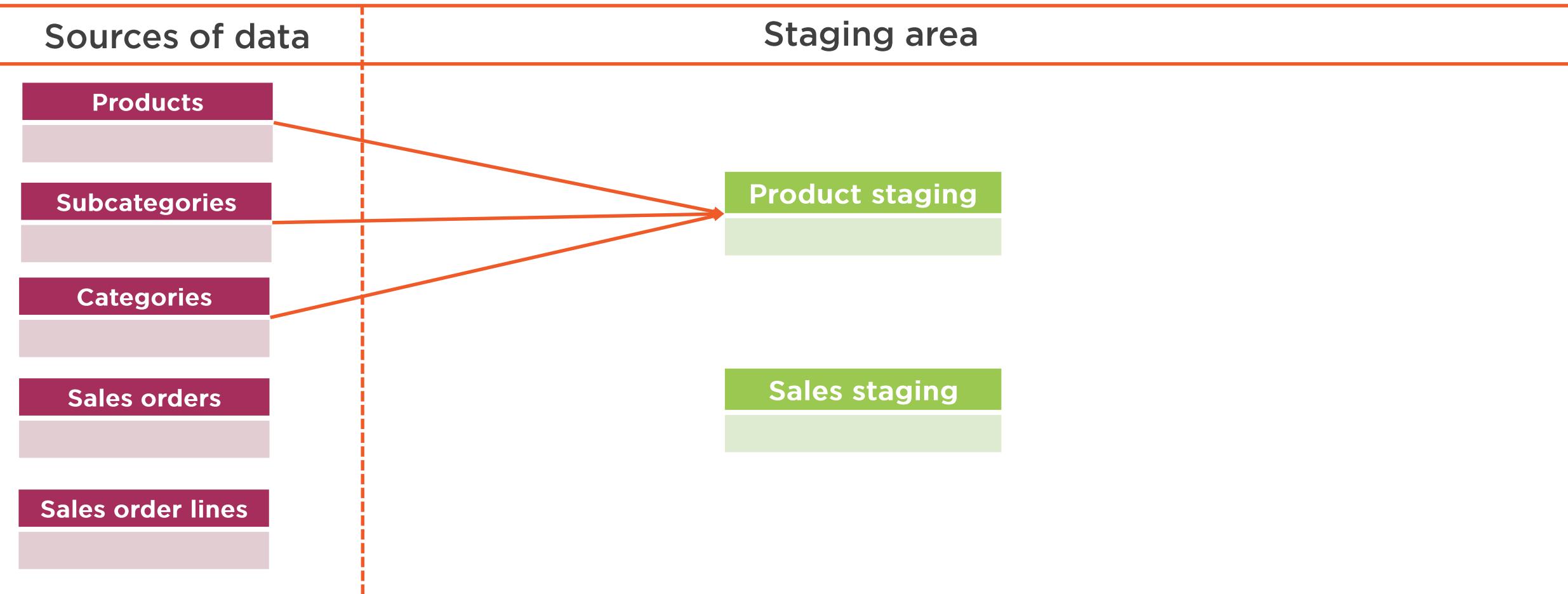
Sales order lines



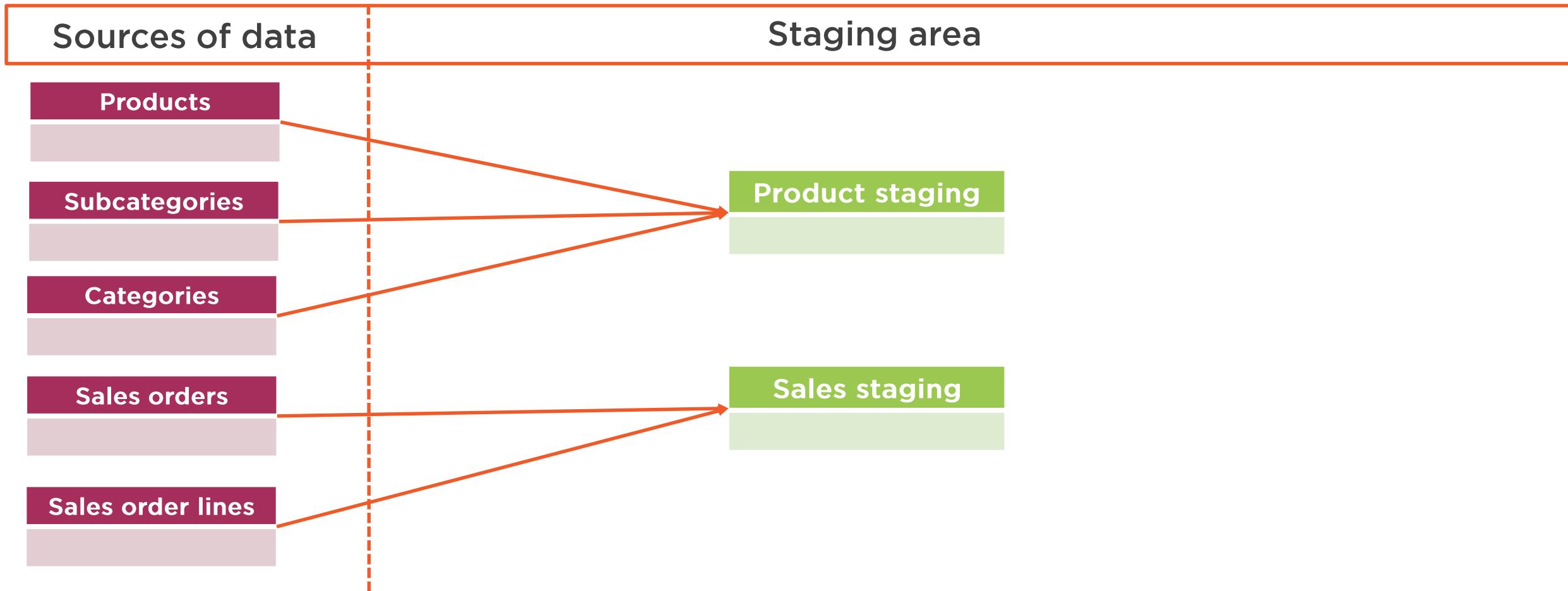
# Overview of an ETL System



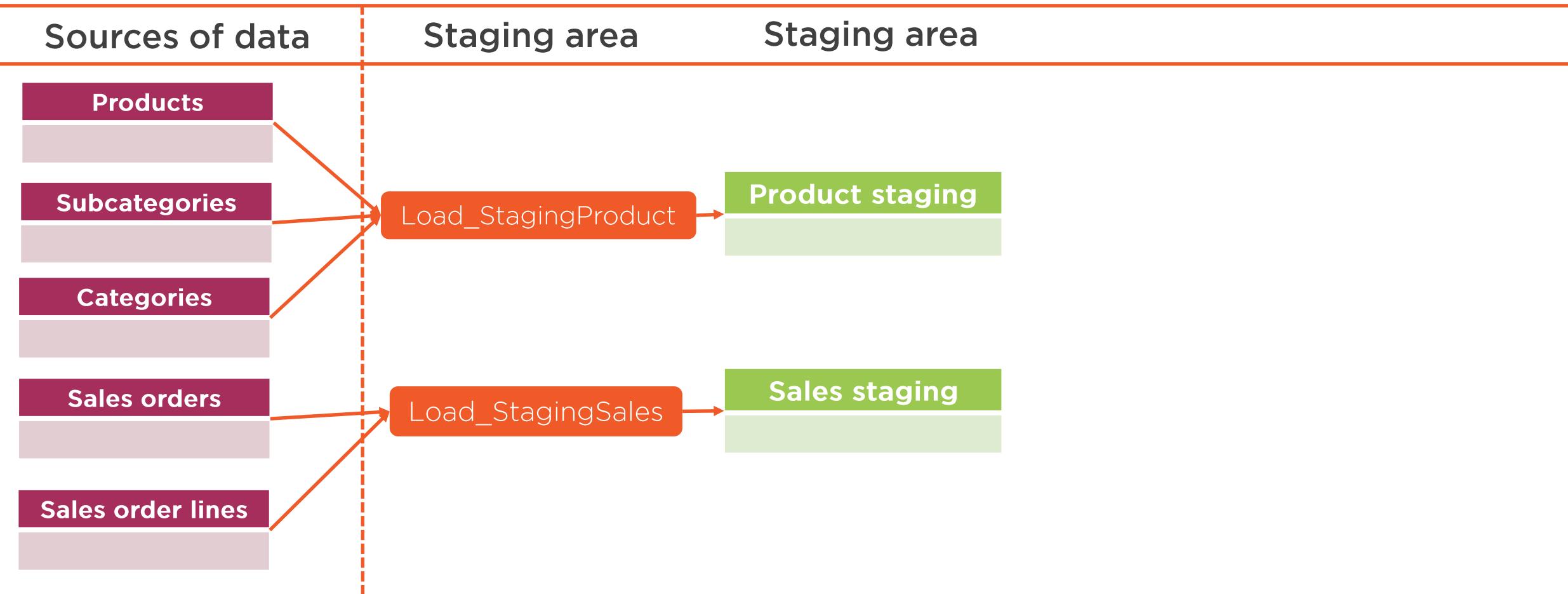
# Overview of an ETL System



# Overview of an ETL System

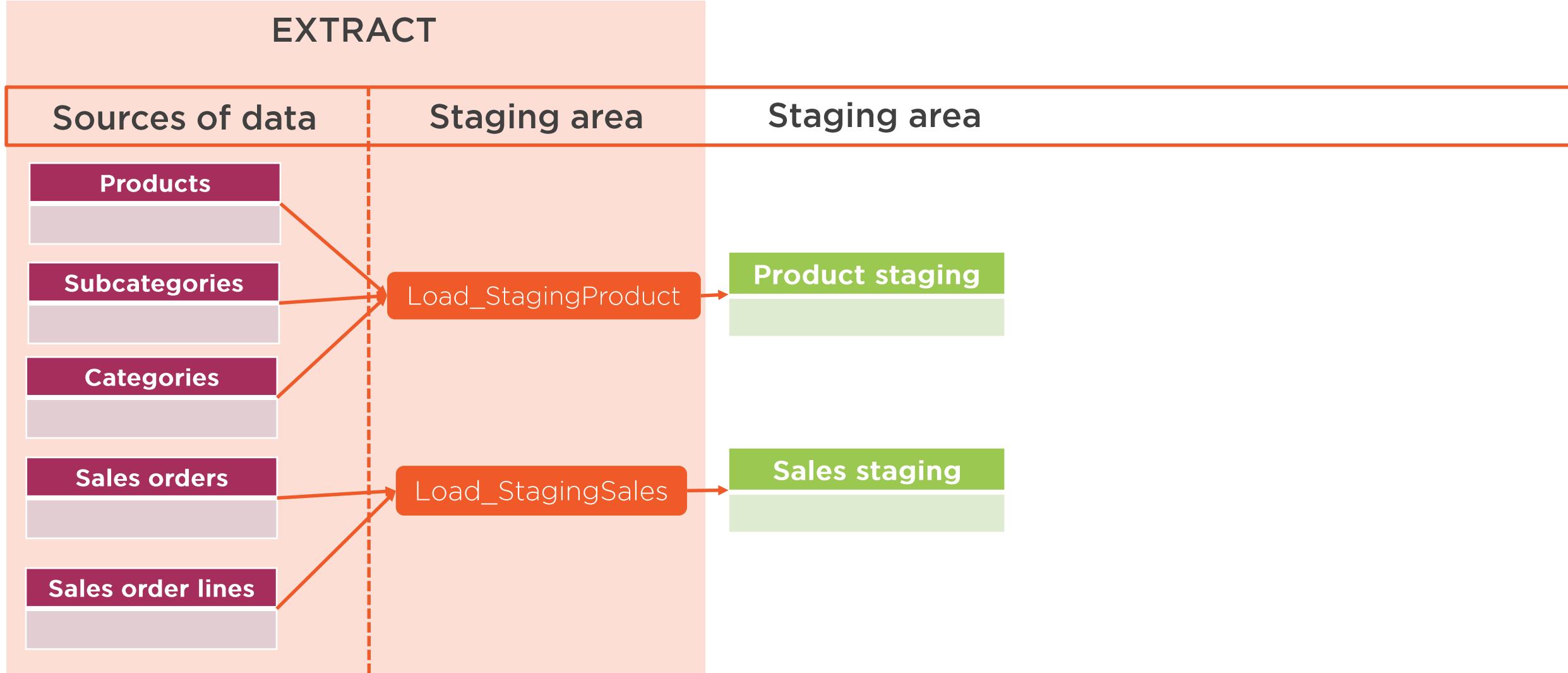


# Overview of an ETL System

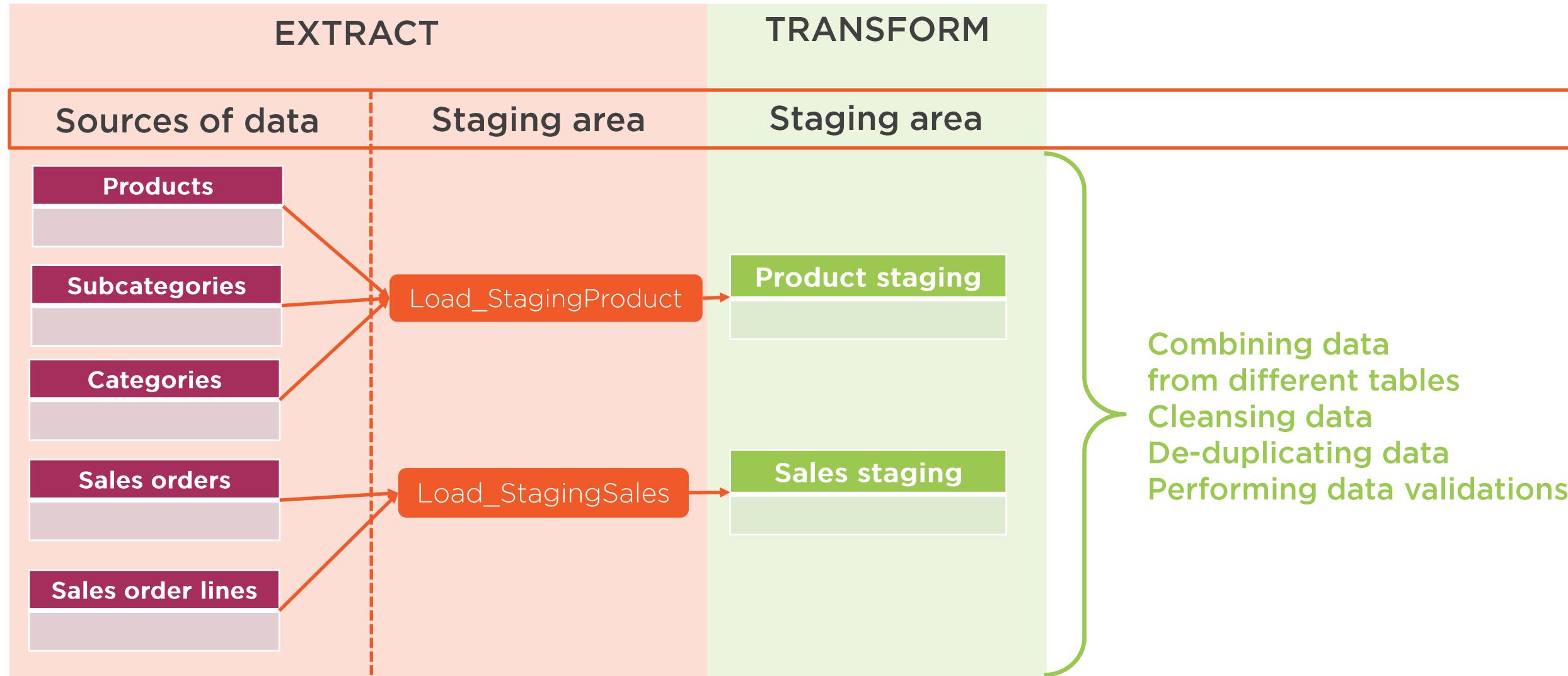


# Overview of an ETL System

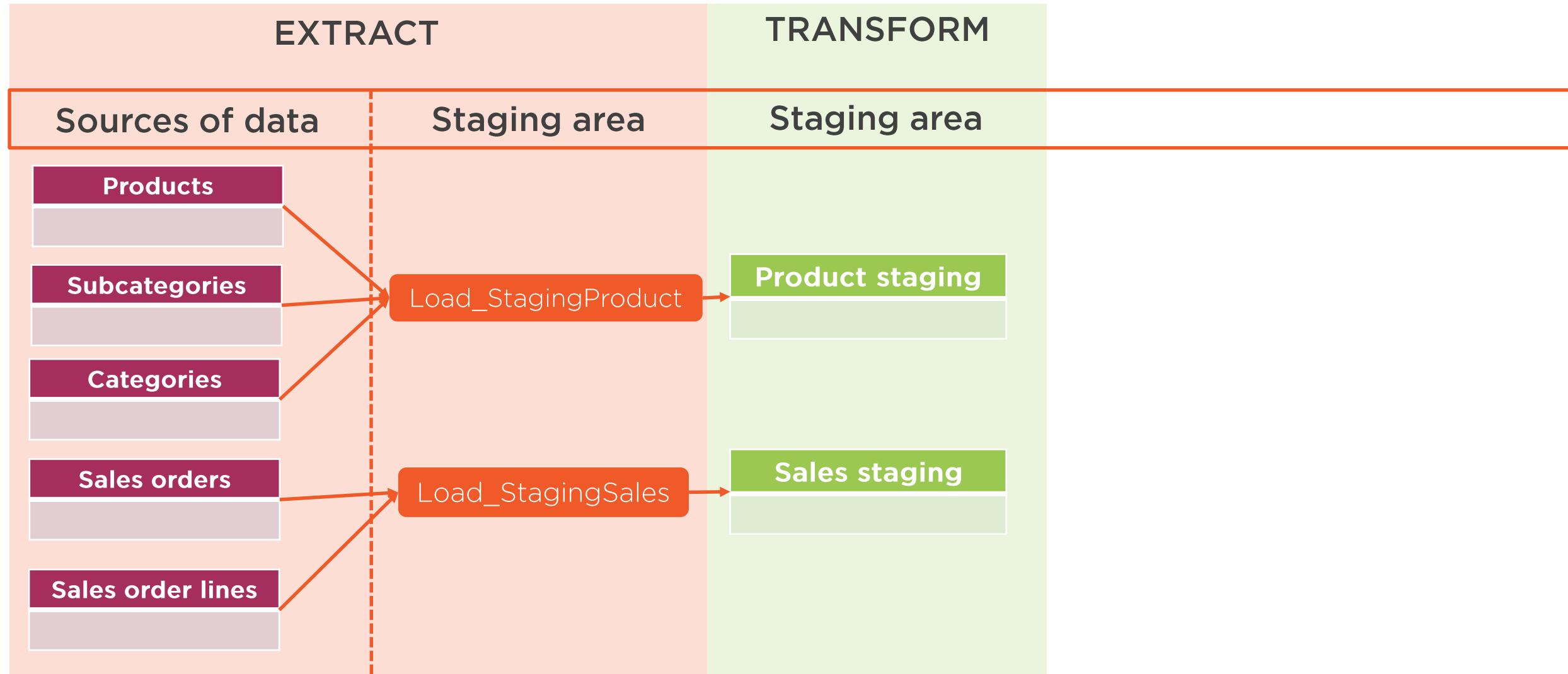
## EXTRACT



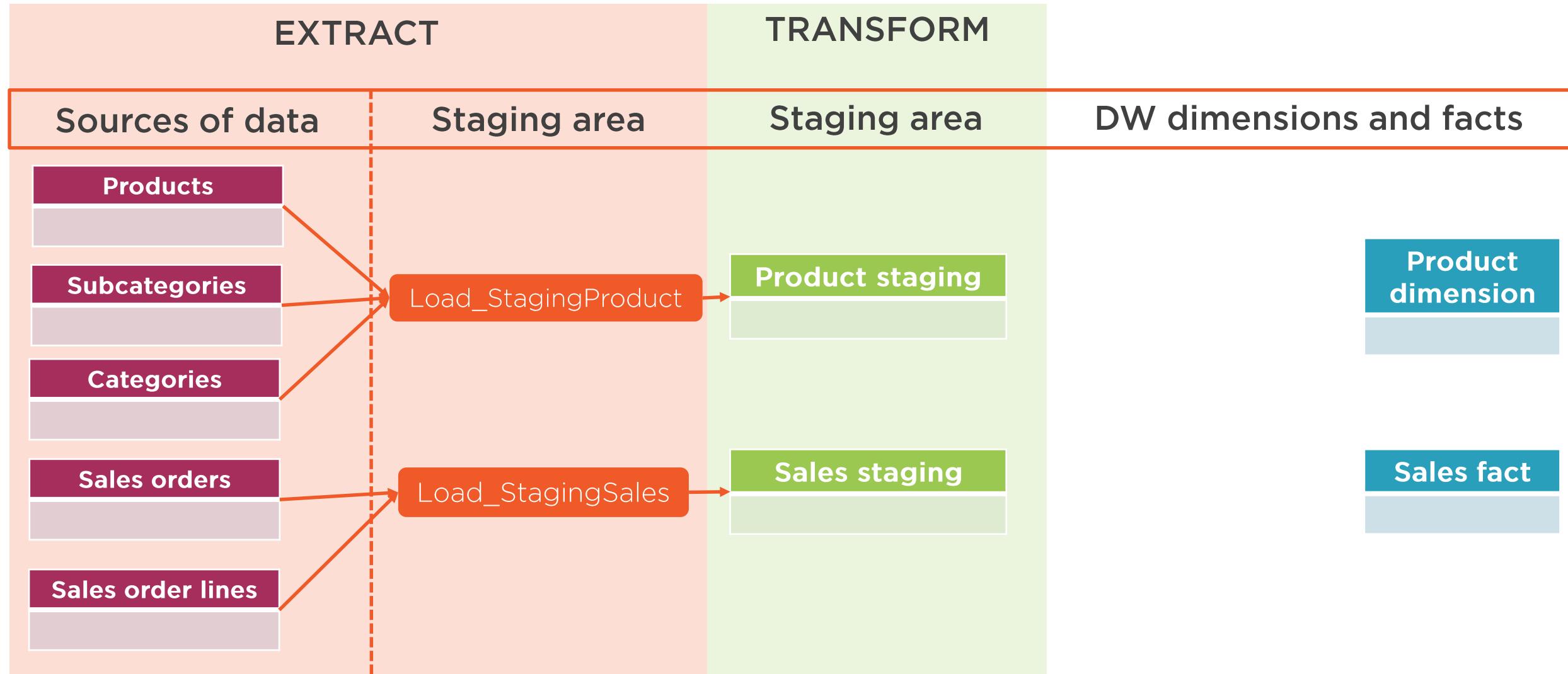
# Overview of an ETL System



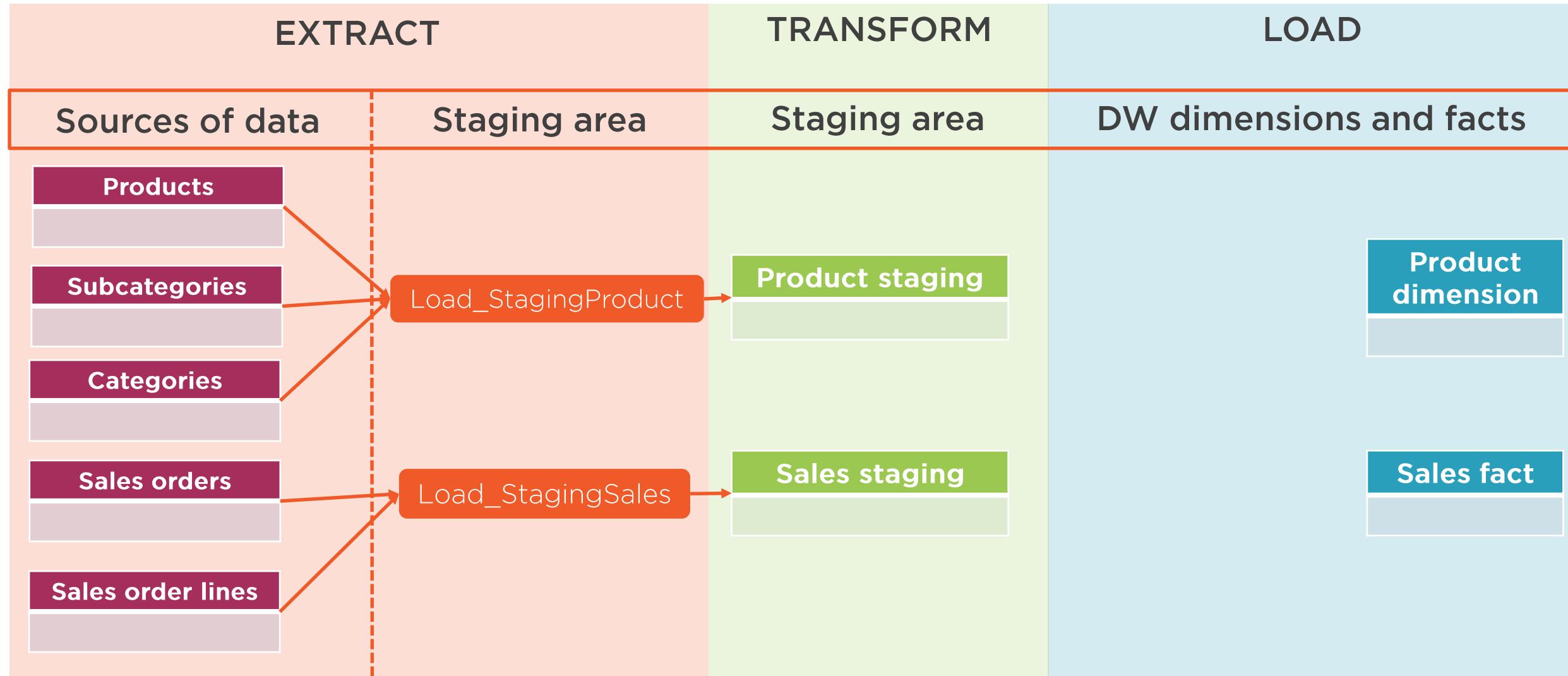
# Overview of an ETL System



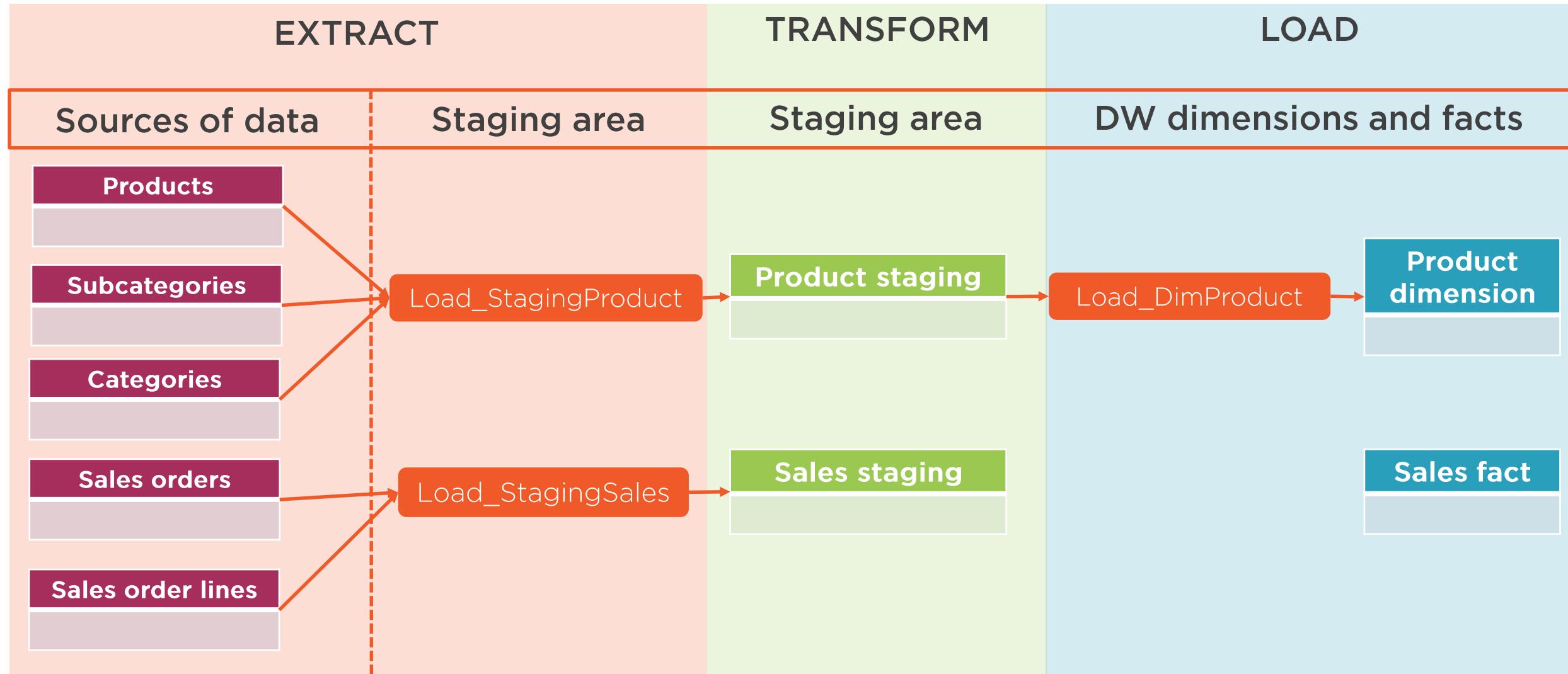
# Overview of an ETL System



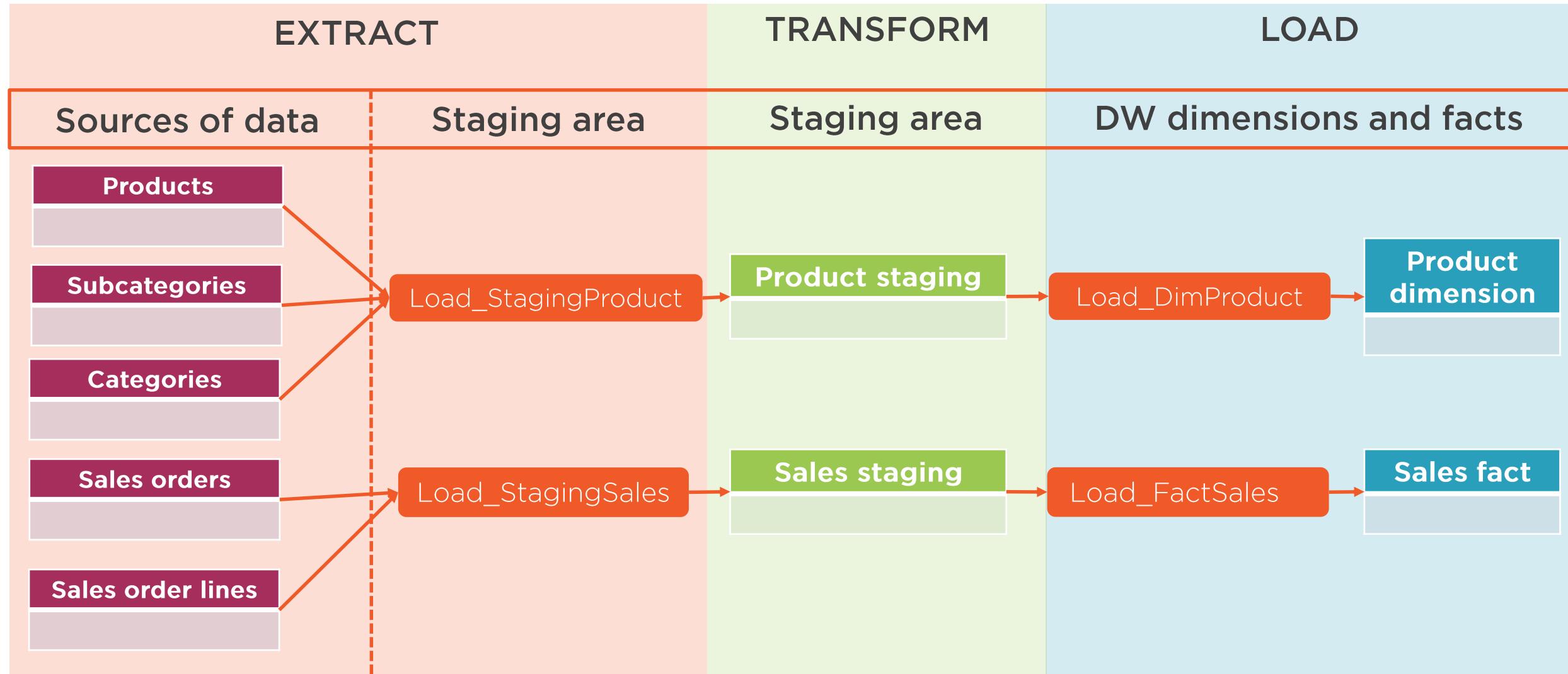
# Overview of an ETL System



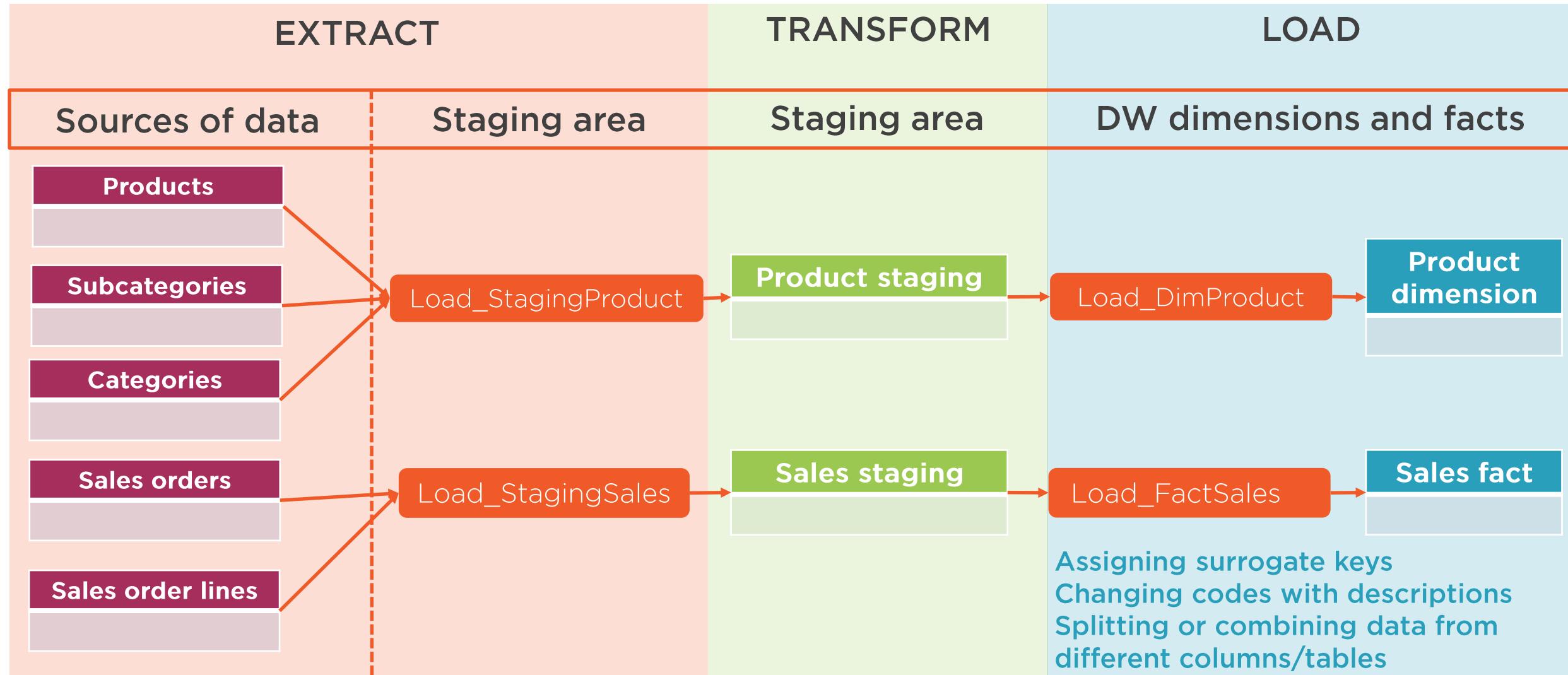
# Overview of an ETL System



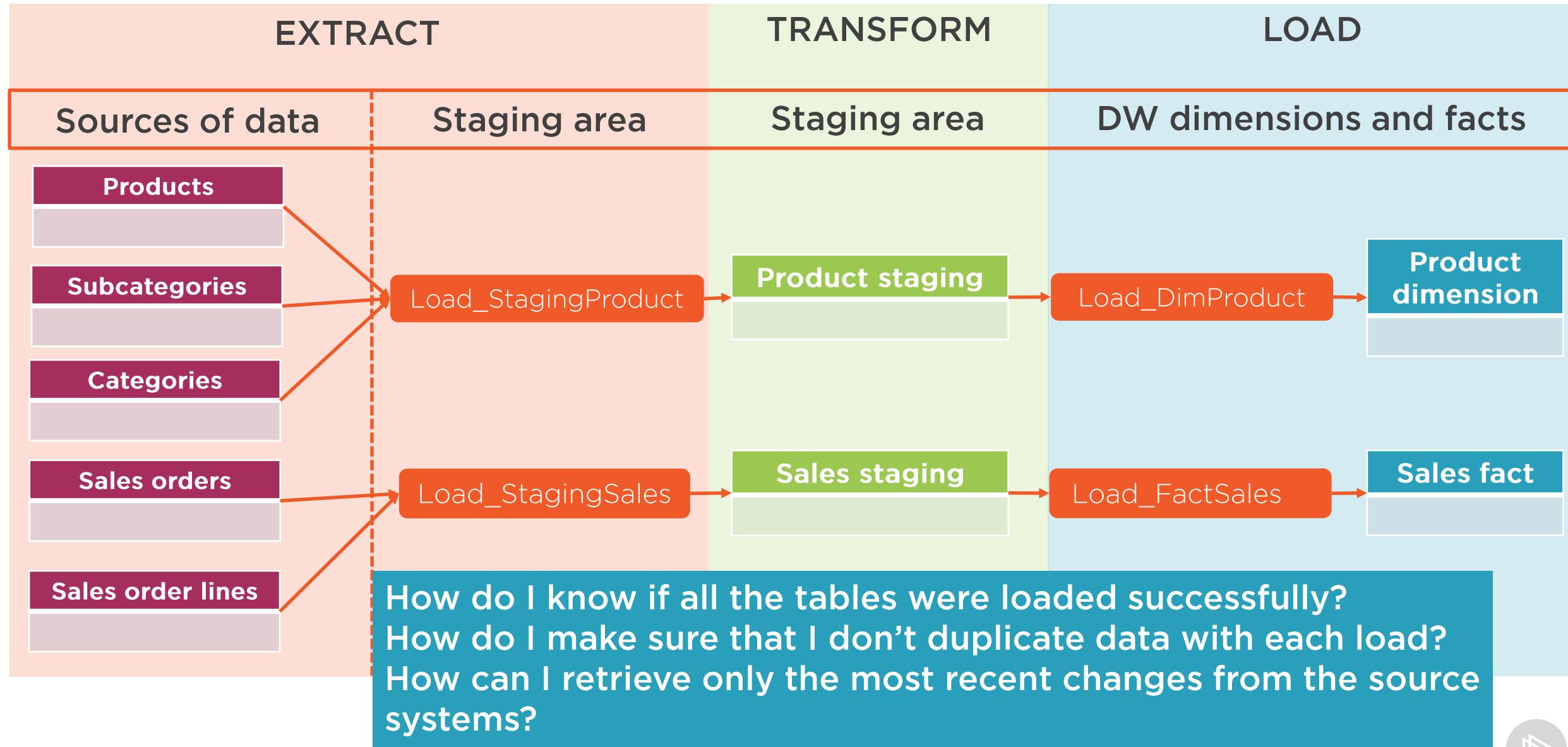
# Overview of an ETL System



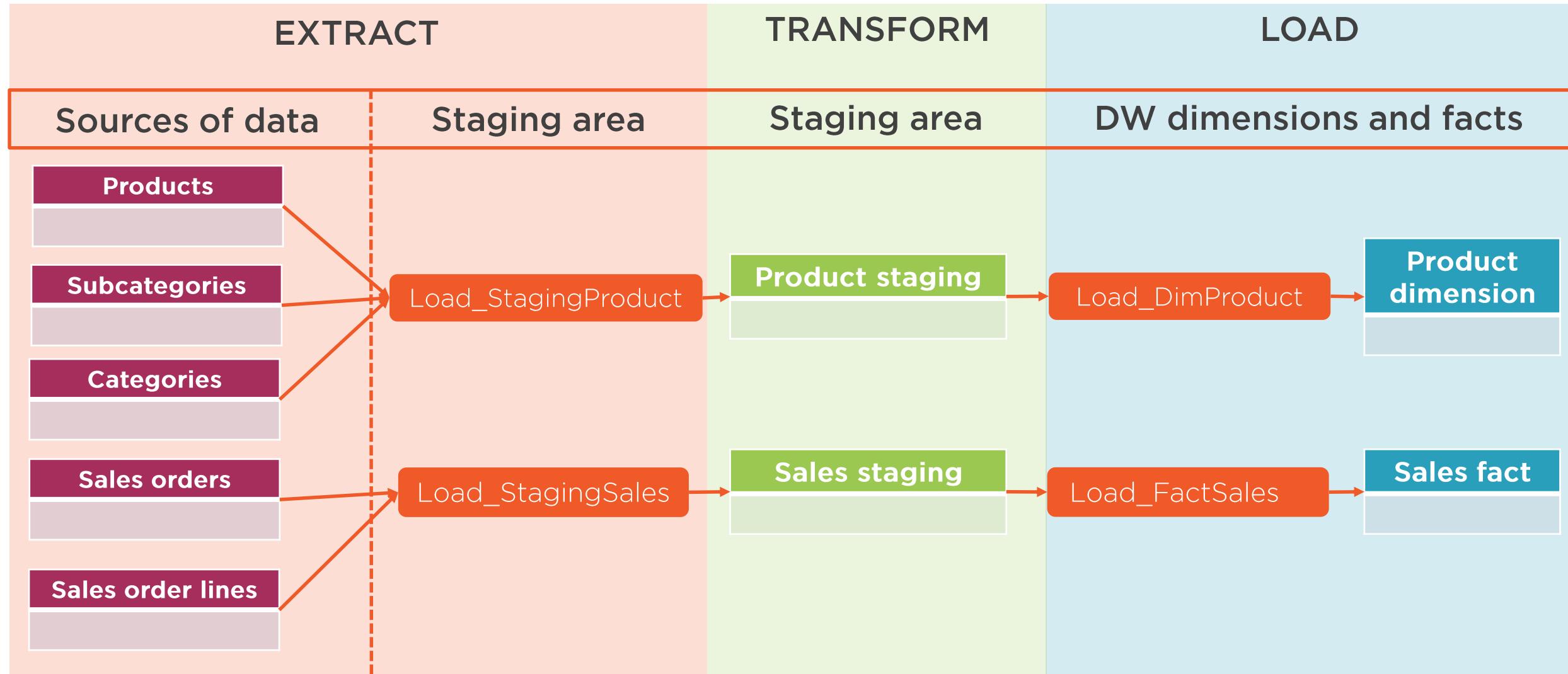
# Overview of an ETL System



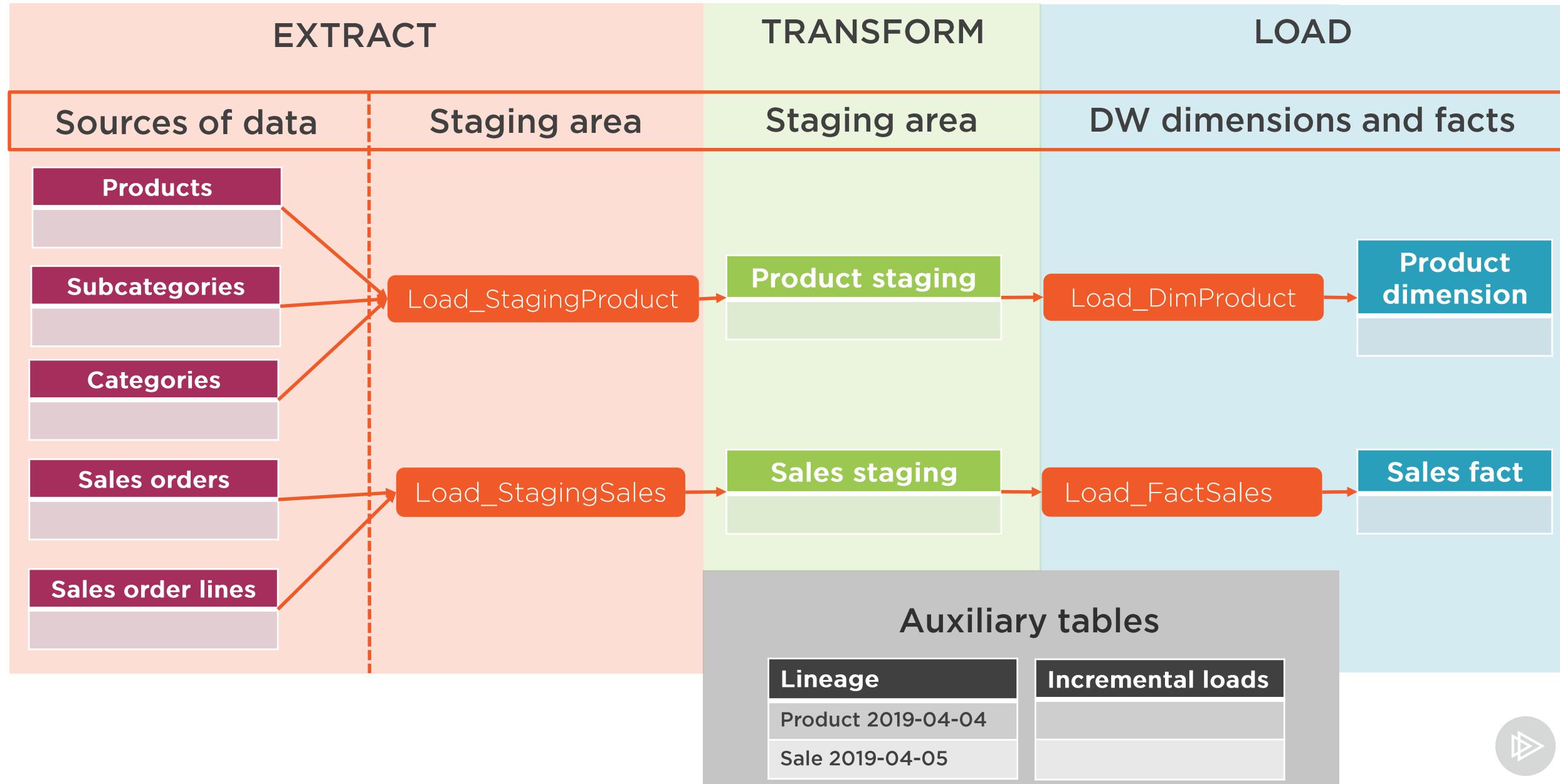
# Overview of an ETL System



# Overview of an ETL System



# Overview of an ETL System



# Types of Data Warehouse Loads

---



# Types of Data Warehouse Loads

## Full/initial load

The process of populating the data warehouse for the first time with data from the operational system

All tables are truncated and reloaded

Old data is lost

Takes a lot of time to finish

Easy to implement

## Incremental load

The process of updating the data warehouse with the operational system changes

Tables are updated with new data

Old data is preserved

Takes less time than the initial load

Implementation is more complex

- Keep track of the previous load date
- Store multiple versions of the same row



# Incremental Load Elements

Source key	Name	Valid from	Valid to
387	Cherry toffee	2019-01-01	2019-09-16
387	Super cherry toffee	2019-09-16	9999-12-31
104	Banana bread	2019-01-01	9999-12-31
105	Grape juice	2019-01-01	9999-12-31

The “Valid from” and “Valid to” columns

Load date key	Table name	Load date
1	Dim_Product	2019-04-13
2	Dim_Employee	2019-01-01
...	...	
16	Dim_Product	2019-04-15

The “Incremental loads” table



# Incremental Load Elements

Source key	Name	Valid from	Valid to
387	Cherry toffee	2019-01-01	2019-09-16
387	Super cherry toffee	2019-09-16	9999-12-31
104	Banana bread	2019-01-01	9999-12-31
105	Grape juice	2019-01-01	9999-12-31

The “Valid from” and “Valid to” columns

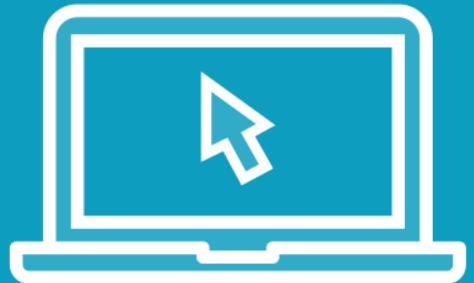
Load date key	Table name	Load date
1	Dim_Product	2019-04-13
2	Dim_Employee	2019-01-01
...	...	
16	Dim_Product	2019-04-15

The “Incremental loads” table

```
SELECT *
FROM Products
WHERE
ModifiedDate > '2019-04-13'
AND ModifiedDate <= '2019-04-15'
```



Demo



**Creating and working with the  
“Incremental loads” table**

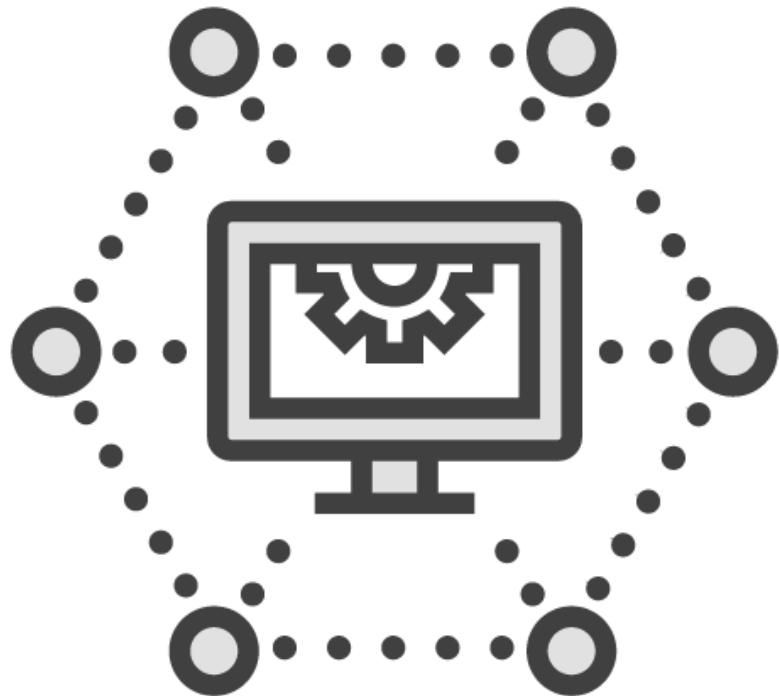


# Setting up Data Lineage

---



# What Is Data Lineage?



**Data lineage tracks the movement of data**

- What are the origins of data
- Where is the data going to
- When was the data loaded or updated
- What transformations are applied to the data

**The complexity of data lineage implementations varies**

- Depends on the necessities of the project

**It is important to set up data lineage in data warehouse projects**



# Advantages of Data Lineage



# Advantages of Data Lineage

**Troubleshooting**



# Advantages of Data Lineage

## Troubleshooting

### Data warehouse

Sale number	Product
123	Cherry toffee

### Source system

Sale number	Product
123	Super cherry toffee



# Advantages of Data Lineage

**Troubleshooting**



# Advantages of Data Lineage

**Troubleshooting**

**Data trust**

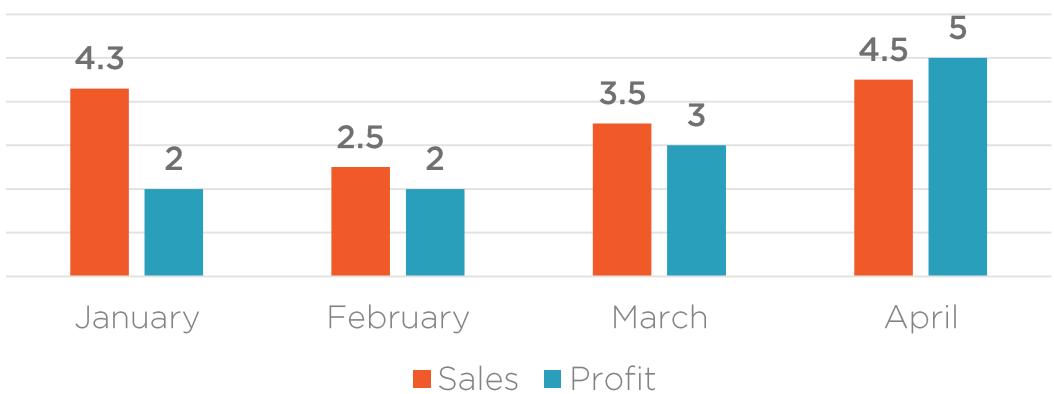


# Advantages of Data Lineage

Troubleshooting

Data trust

Sales and profit



# Advantages of Data Lineage

**Troubleshooting**

**Data trust**



# Advantages of Data Lineage

**Troubleshooting**

**Data trust**

**Transparent business rules**



# Advantages of Data Lineage

**Troubleshooting**

**Data trust**

**Transparent business rules**

Rate	% of VAT	Applies to
Standard	20%	Mixed ice cream, frozen yogurt
Reduced	5%	Some goods and services
Zero	0%	Herbal tea, pita bread, cold sandwiches, crocodile meat



# Advantages of Data Lineage

**Troubleshooting**

**Data trust**

**Transparent business rules**



# Advantages of Data Lineage

**Troubleshooting**

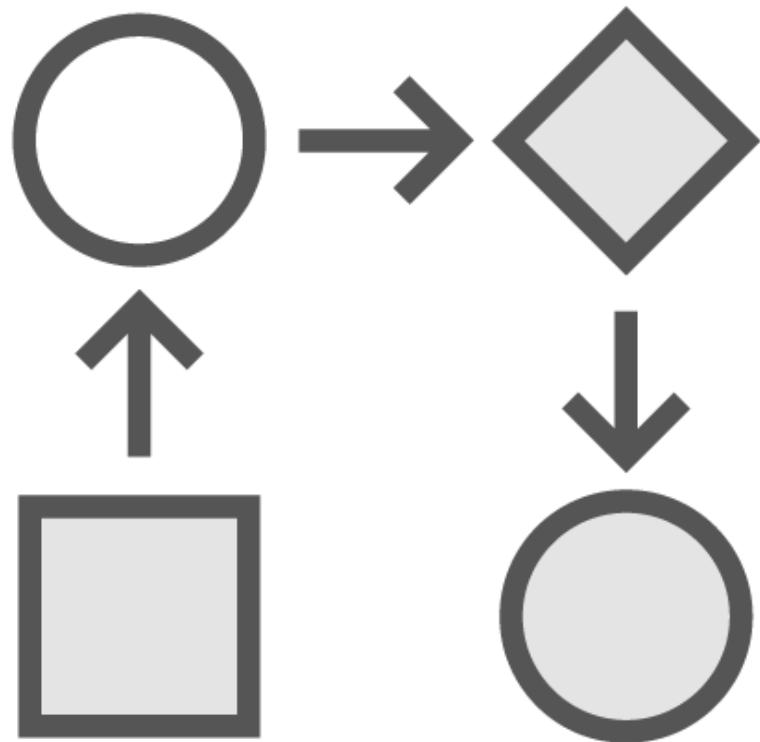
**Data trust**

**Transparent business rules**

**Data audit**



# Implementing Data Lineage



## Ensure row uniqueness

- Use surrogate keys
- Data passes multiple stages from source to destination
- Keeping the same key through all the stages helps tracing back the data

## Keep track of the operation that loaded each row

- Create a “Lineage” column in all tables from the data warehouse



# Implementing Data Lineage

Lineage key	Table name	Start	End	Type	Status



# Implementing Data Lineage

Lineage key	Table name	Start	End	Type	Status

Product key	Product name	...	Lineage key

Employee key	Employee name	...	Lineage key



# Implementing Data Lineage

Lineage key	Table name	Start	End	Type	Status
10	Dim_Product	2019-03-04 20:39:10.000	null	F	P

Product key	Product name	...	Lineage key

Employee key	Employee name	...	Lineage key



# Implementing Data Lineage

Lineage key	Table name	Start	End	Type	Status
10	Dim_Product	2019-03-04 20:39:10.000	null	F	P

Product key	Product name	...	Lineage key
1	Cotton candy		10
2	Green tea		10

Employee key	Employee name	...	Lineage key



# Implementing Data Lineage

Lineage key	Table name	Start	End	Type	Status
10	Dim_Product	2019-03-04 20:39:10.000	2019-03-04 20:54:35.000	F	S

Product key	Product name	...	Lineage key
1	Cotton candy		10
2	Green tea		10

Employee key	Employee name	...	Lineage key



# Implementing Data Lineage

Lineage key	Table name	Start	End	Type	Status
10	Dim_Product	2019-03-04 20:39:10.000	2019-03-04 20:54:35.000	F	S
11	Dim_Employee	2019-03-04 20:45:21.000	null	F	P

Product key	Product name	...	Lineage key
1	Cotton candy		10
2	Green tea		10

Employee key	Employee name	...	Lineage key



# Implementing Data Lineage

Lineage key	Table name	Start	End	Type	Status
10	Dim_Product	2019-03-04 20:39:10.000	2019-03-04 20:54:35.000	F	S
11	Dim_Employee	2019-03-04 20:45:21.000	null	F	P

Product key	Product name	...	Lineage key
1	Cotton candy		10
2	Green tea		10

Employee key	Employee name	...	Lineage key
1	Mary Baker		11
2	Jack Peanut		11
3	Gigi Knopper		11
...	...		...



# Implementing Data Lineage

Lineage key	Table name	Start	End	Type	Status
10	Dim_Product	2019-03-04 20:39:10.000	2019-03-04 20:54:35.000	F	S
11	Dim_Employee	2019-03-04 20:45:21.000	2019-03-04 20:59:12.000	F	S

Product key	Product name	...	Lineage key
1	Cotton candy		10
2	Green tea		10

Employee key	Employee name	...	Lineage key
1	Mary Baker		11
2	Jack Peanut		11
3	Gigi Knopper		11
...	...		...



# Implementing Data Lineage

Lineage key	Table name	Start	End	Type	Status
10	Dim_Product	2019-03-04 20:39:10.000	2019-03-04 20:54:35.000	F	S
11	Dim_Employee	2019-03-04 20:45:21.000	2019-03-04 20:59:12.000	F	S
...	...	...	...		...

Product key	Product name	...	Lineage key
1	Cotton candy		10
2	Green tea		10

Employee key	Employee name	...	Lineage key
1	Mary Baker		11
2	Jack Peanut		11
3	Gigi Knopper		11
...	...		...



# Implementing Data Lineage

Lineage key	Table name	Start	End	Type	Status
10	Dim_Product	2019-03-04 20:39:10.000	2019-03-04 20:54:35.000	F	S
11	Dim_Employee	2019-03-04 20:45:21.000	2019-03-04 20:59:12.000	F	S
...	...	...	...		...
28	Dim_Product	2019-03-15 20:07:10.000	null	I	P

Product key	Product name	...	Lineage key
1	Cotton candy		10
2	Green tea		10

Employee key	Employee name	...	Lineage key
1	Mary Baker		11
2	Jack Peanut		11
3	Gigi Knopper		11
...	...		...



# Implementing Data Lineage

Lineage key	Table name	Start	End	Type	Status
10	Dim_Product	2019-03-04 20:39:10.000	2019-03-04 20:54:35.000	F	S
11	Dim_Employee	2019-03-04 20:45:21.000	2019-03-04 20:59:12.000	F	S
...	...	...	...		...
28	Dim_Product	2019-03-15 20:07:10.000	null	I	P

Product key	Product name	...	Lineage key
1	Cotton candy		10
2	Green tea		10
3	Grape juice		28
4	Banana bread		28

Employee key	Employee name	...	Lineage key
1	Mary Baker		11
2	Jack Peanut		11
3	Gigi Knopper		11
...	...		...



# Implementing Data Lineage

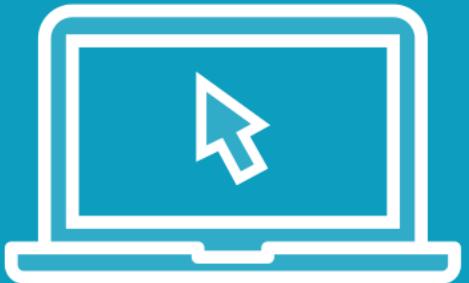
Lineage key	Table name	Start	End	Type	Status
10	Dim_Product	2019-03-04 20:39:10.000	2019-03-04 20:54:35.000	F	S
11	Dim_Employee	2019-03-04 20:45:21.000	2019-03-04 20:59:12.000	F	S
...	...	...	...		...
28	Dim_Product	2019-03-15 20:07:10.000	2019-03-15 20:12:33.000	I	S

Product key	Product name	...	Lineage key
1	Cotton candy		10
2	Green tea		10
3	Grape juice		28
4	Banana bread		28

Employee key	Employee name	...	Lineage key
1	Mary Baker		11
2	Jack Peanut		11
3	Gigi Knopper		11
...	...		...



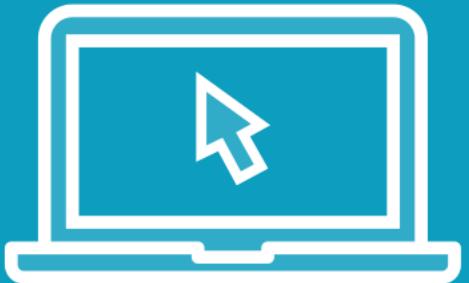
Demo



**Creating and working with the Lineage table**



# Demo



## Populating a dimension table with the help of stored procedures

- Load\_StagingProduct
- Load\_DimProduct



# Objects Participating in a Load Process

Object (table/stored procedure)	Was discussed
Fact/dimension tables	✓
Staging tables	✓
SP for loading the staging table	✓
SP for loading the fact/dim table	✓
Lineage table	✓
Incremental loads table	✓
SPs for updating the log tables	✓



# Summary



## Overview of an ETL system

## Types of data warehouse loads

- Full (initial) load
- Incremental load

## Setting up data lineage

## Demos

- Creating and using the auxiliary tables
  - Lineage
  - Incremental loads
- Populating with data the staging and dimension tables



# Setting up an ETL Project in SSIS

---



**Ana Voicu**  
@ana\_voicu



# Overview



**What is an ETL tool**

**Advantages of using an ETL tool**

**Overview of SSIS**

**Demos**

- Creating a new SSIS project with Visual Studio
- Adding connection managers
- Creating control flow components
- Linking tasks with precedence constraints
- Adding project variables
- Executing SQL tasks with parameters



Disclaimer: This is not an intensive course on SSIS

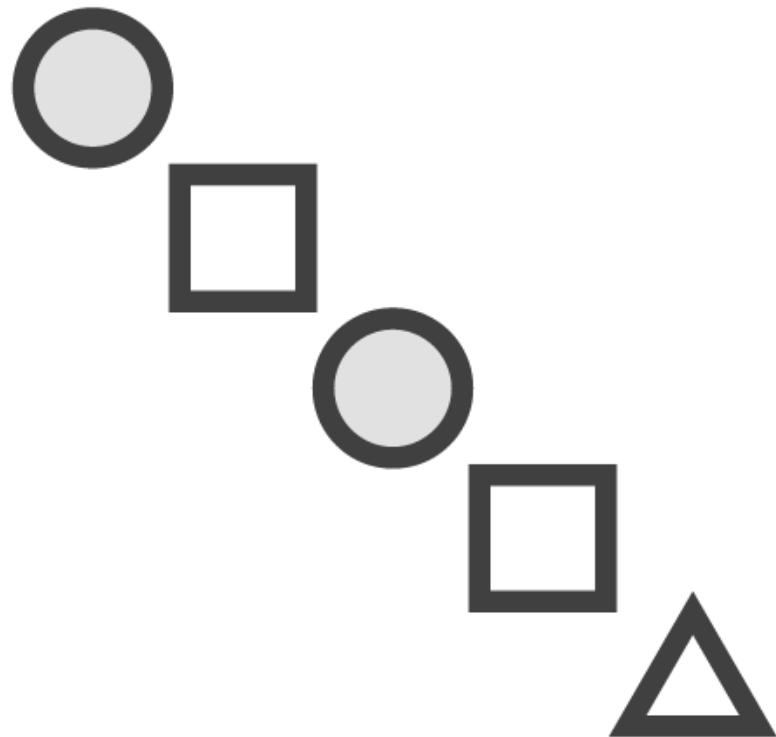


# Advantages of Using an ETL Tool

---



# What Are ETL Tools?

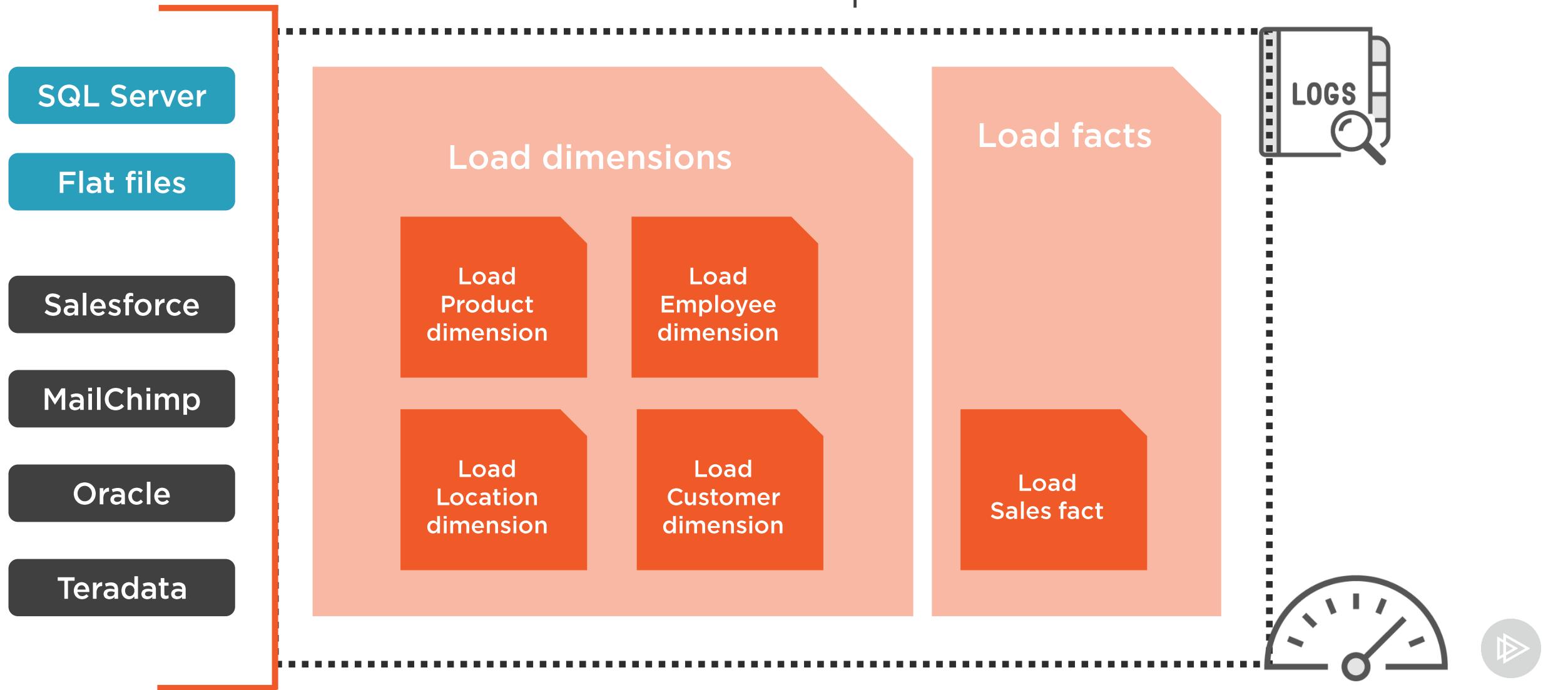


**Software systems for building and automating ETL processes**

**Offer an alternative to creating all the code for a DW solution by hand**



# Loading a Data Warehouse Only Using SQL Scripts



# What Is an ETL Tool?



A system comprised of all the components needed to build ETL processes

## Components:

- Data flow management
- Connectors to other systems
- Data transformation components
- Logging and auditing
- Debugging
- Deployment



# Advantages of Using an ETL Tool

**Flexibility**

**Maintainability**

- Easier than working with hundreds of stored procedures
- ETL tools are very graphical and easy to interpret
- Easy to learn and use



# Advantages of Using an ETL Tool

**Flexibility**

**Maintainability**



# Advantages of Using an ETL Tool

Flexibility

Maintainability

Performance

Logging and audit

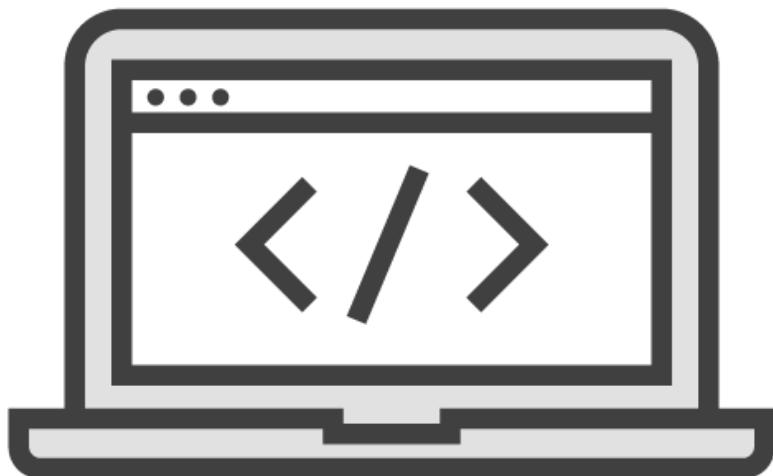


# Overview of SSIS

---



# What Is SSIS?



**Component of SQL Server**

**Used for data integration and automating processes**

**Provides all the components necessary for creating an ETL process**

**Graphical tool**

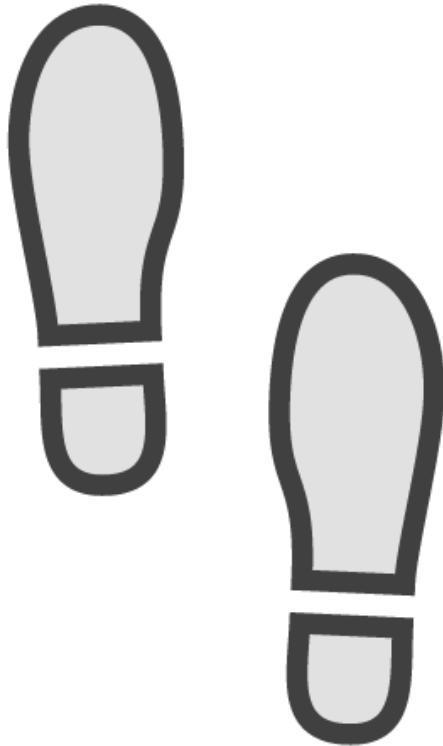
- Doesn't require writing complex code
- Based on drag and drop actions

**The output is a workflow, or pipeline**

**The unit of work in SSIS is called a package**



# Steps for Loading the Product Dimension



1. Save the current load date
2. Insert new row in the Lineage table
3. Truncate the staging table
4. Populate the staging table
5. Transfer data from staging to dimension

These steps are part of a flow or process

- Executed together
- In a specified order
- Executed multiple times



# Overview of a SSIS Package



Get date of current load



# Overview of a SSIS Package



Get date of current load



`@[User::LoadDate] = GETDATE()`



# Overview of a SSIS Package



Get date of current load

..... @User::LoadDate = GETDATE()



Update Lineage table



# Overview of a SSIS Package



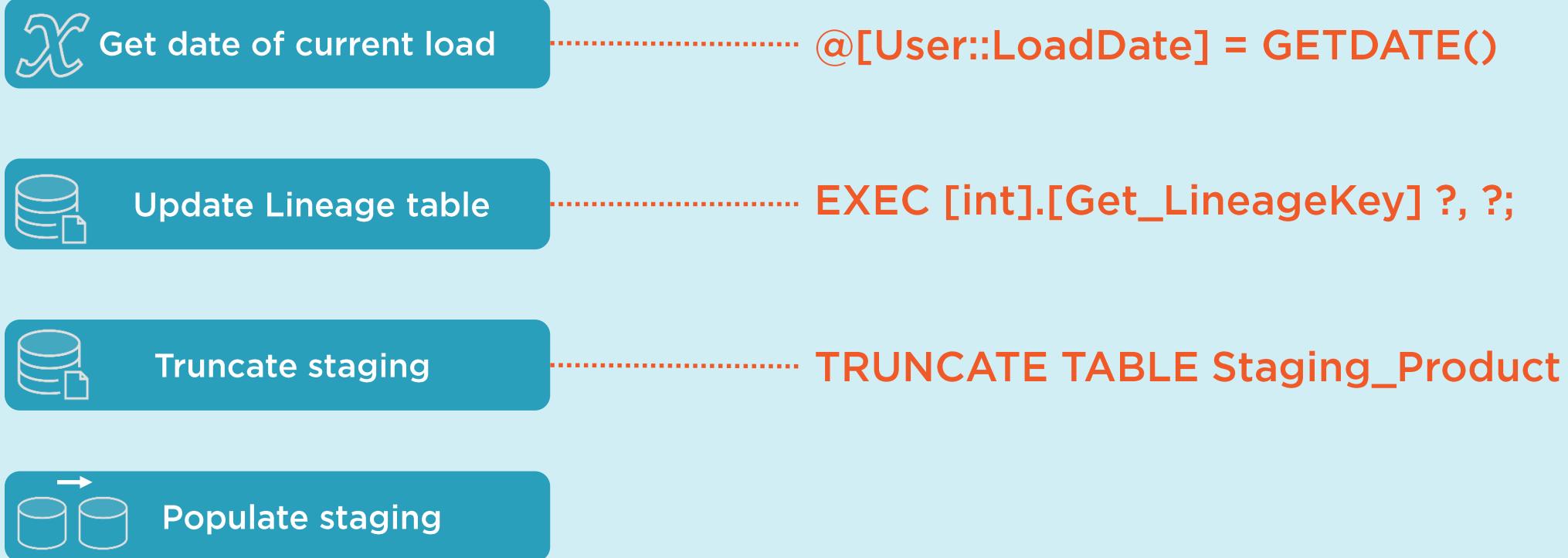
# Overview of a SSIS Package



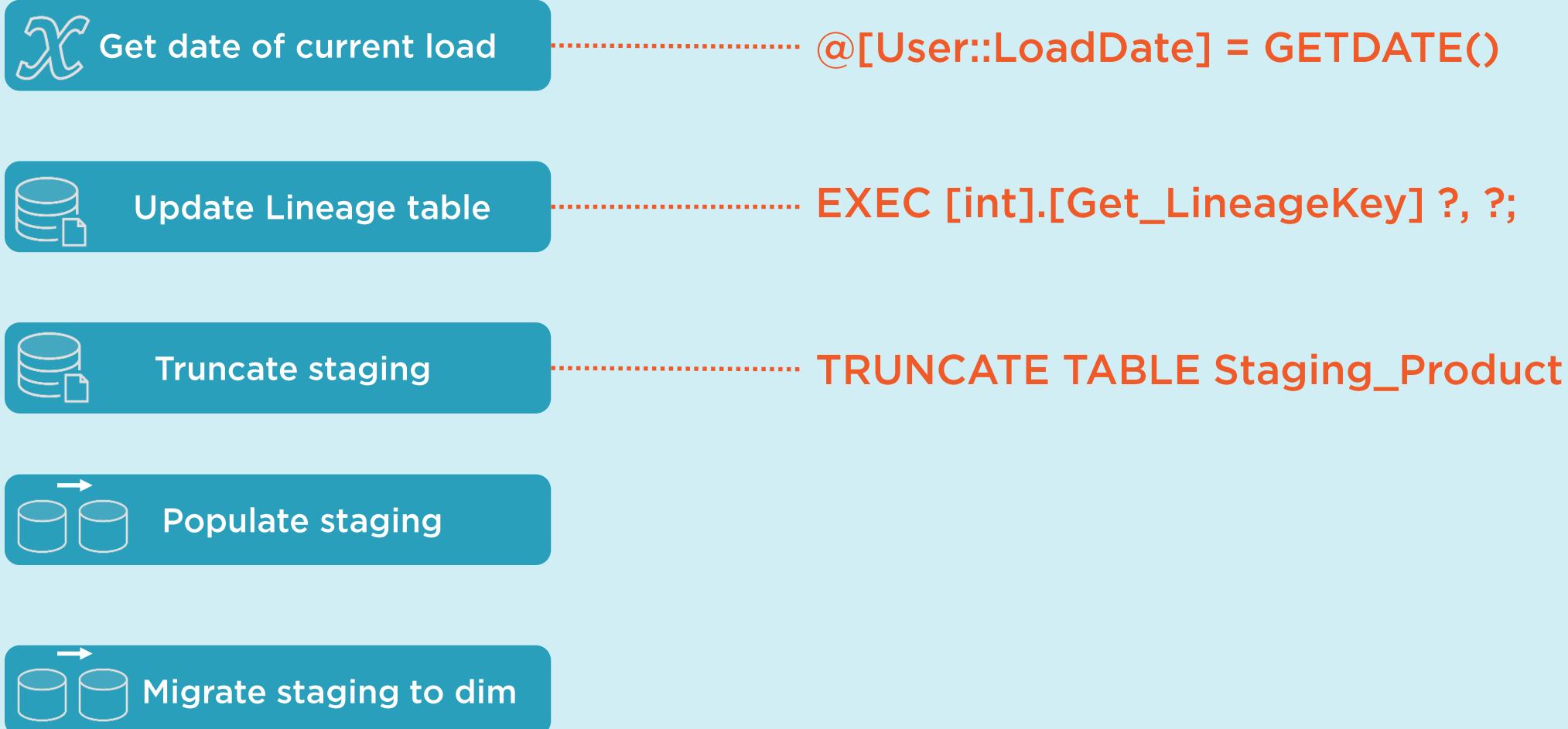
# Overview of a SSIS Package



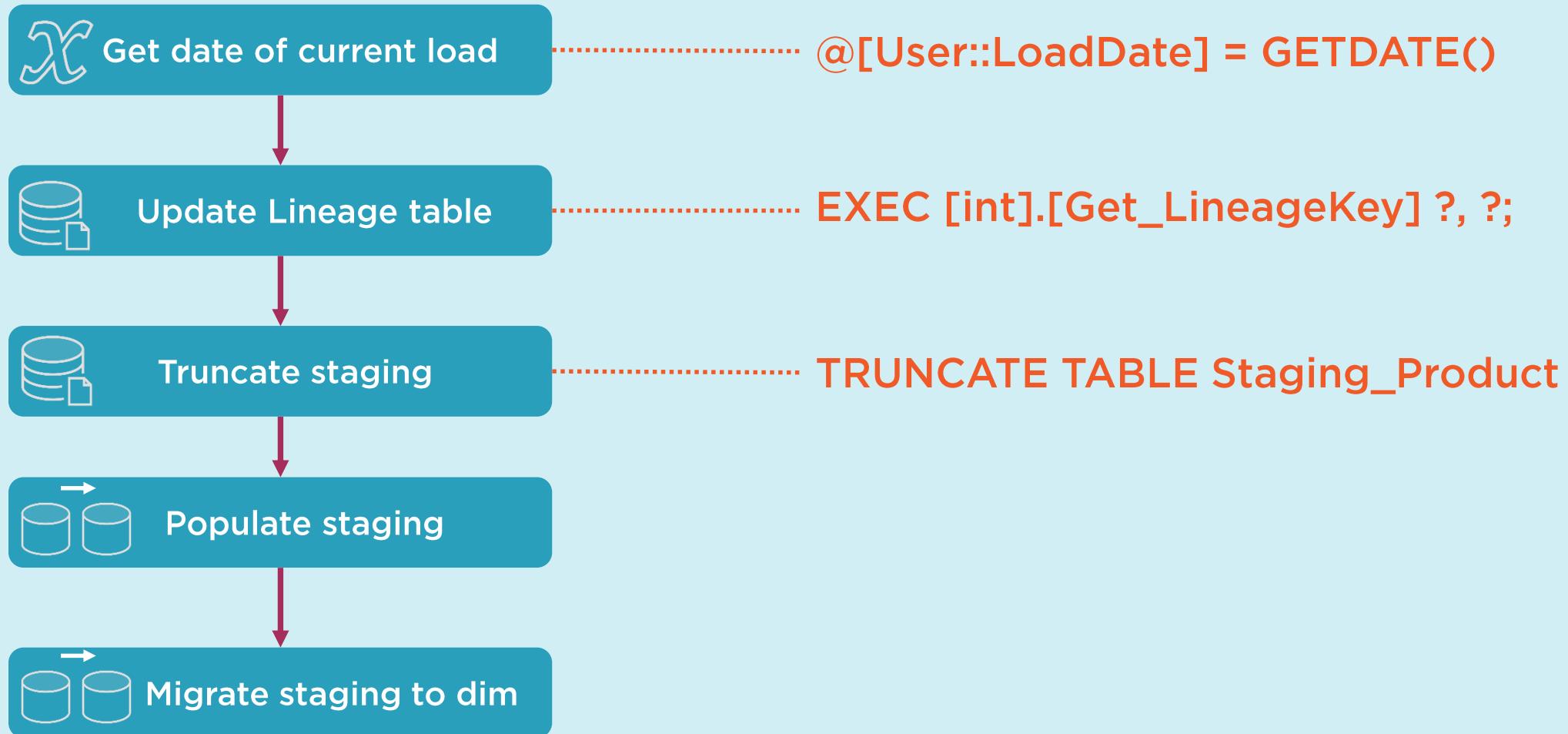
# Overview of a SSIS Package



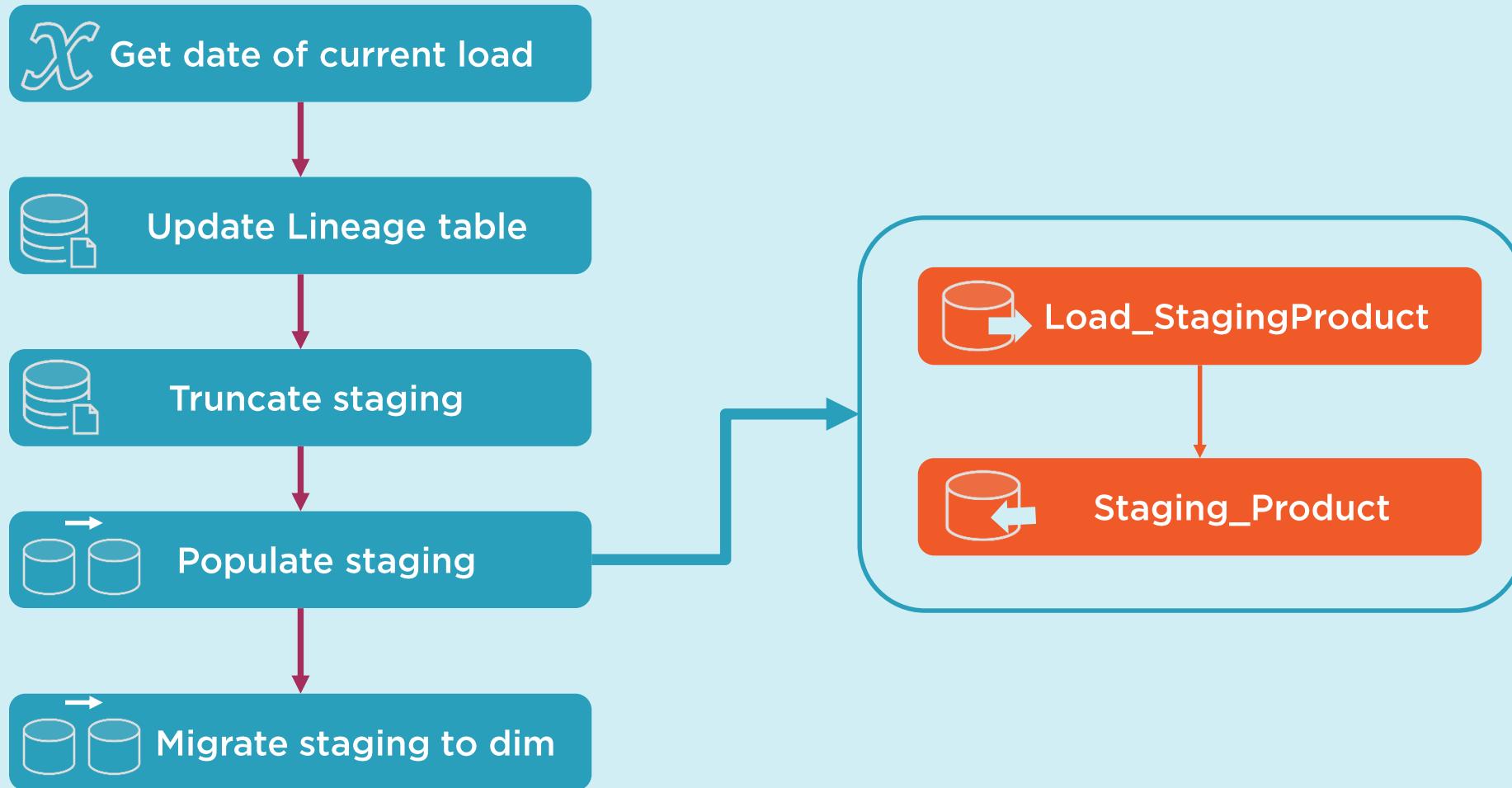
# Overview of a SSIS Package



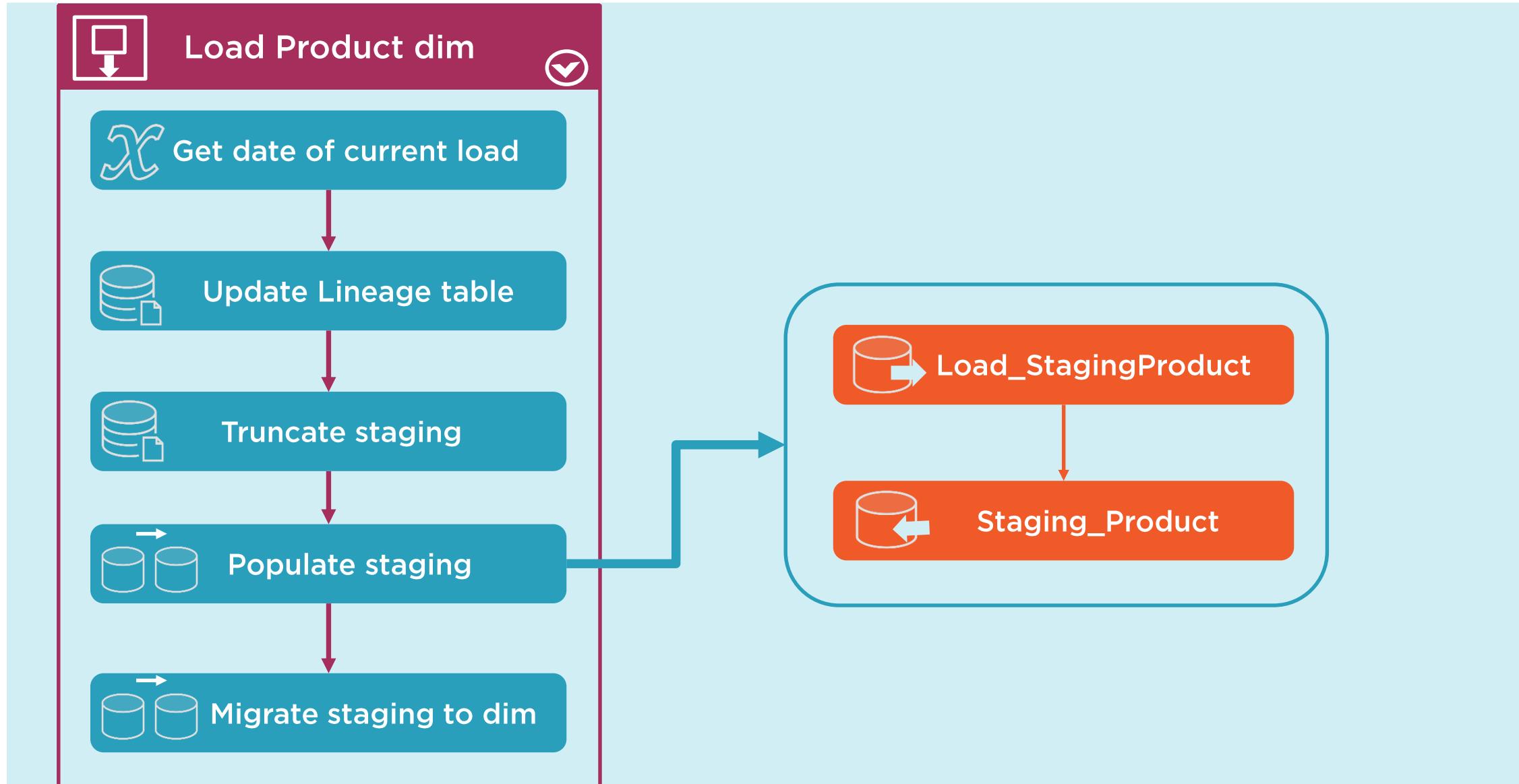
# Overview of a SSIS Package



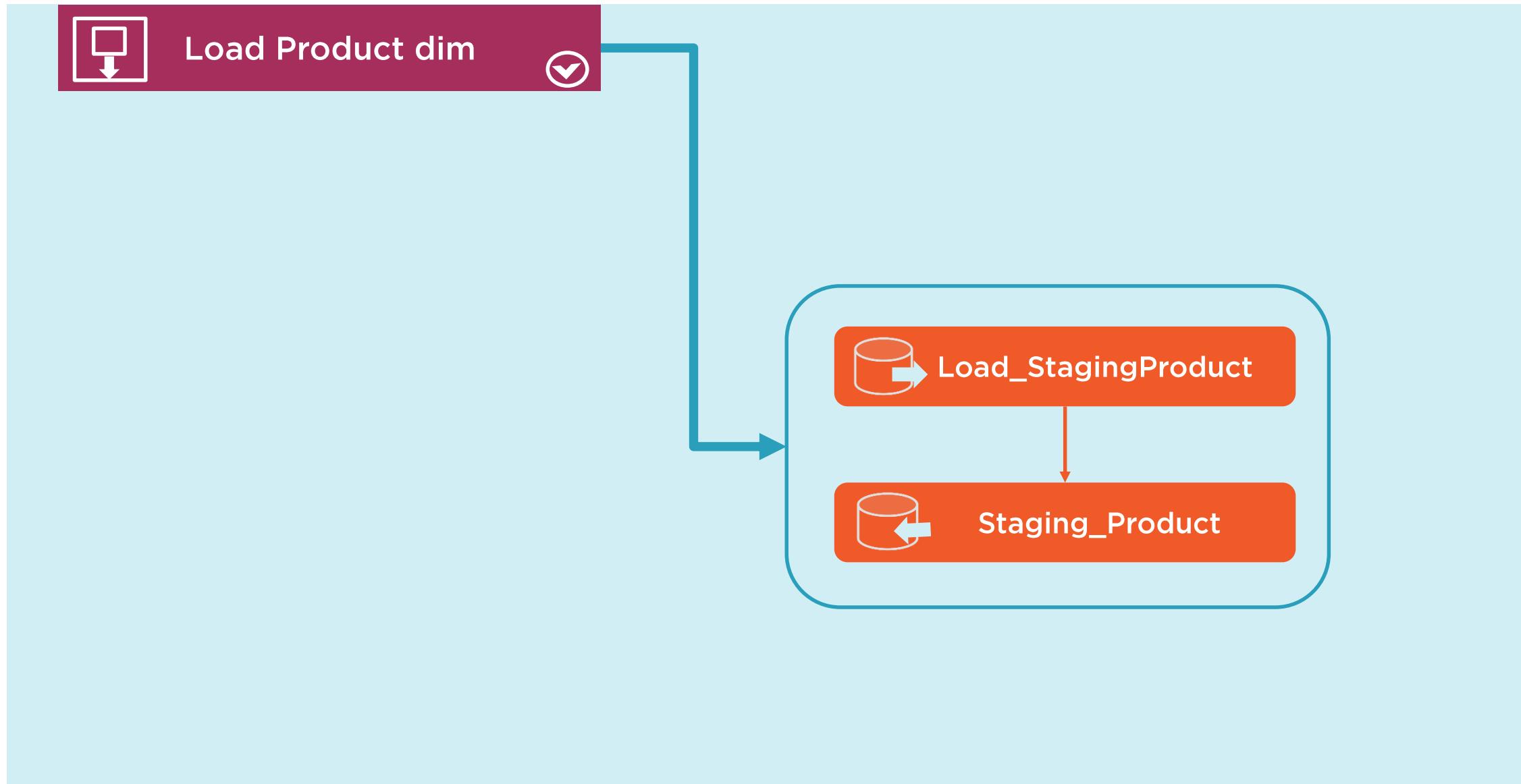
# Overview of a SSIS Package



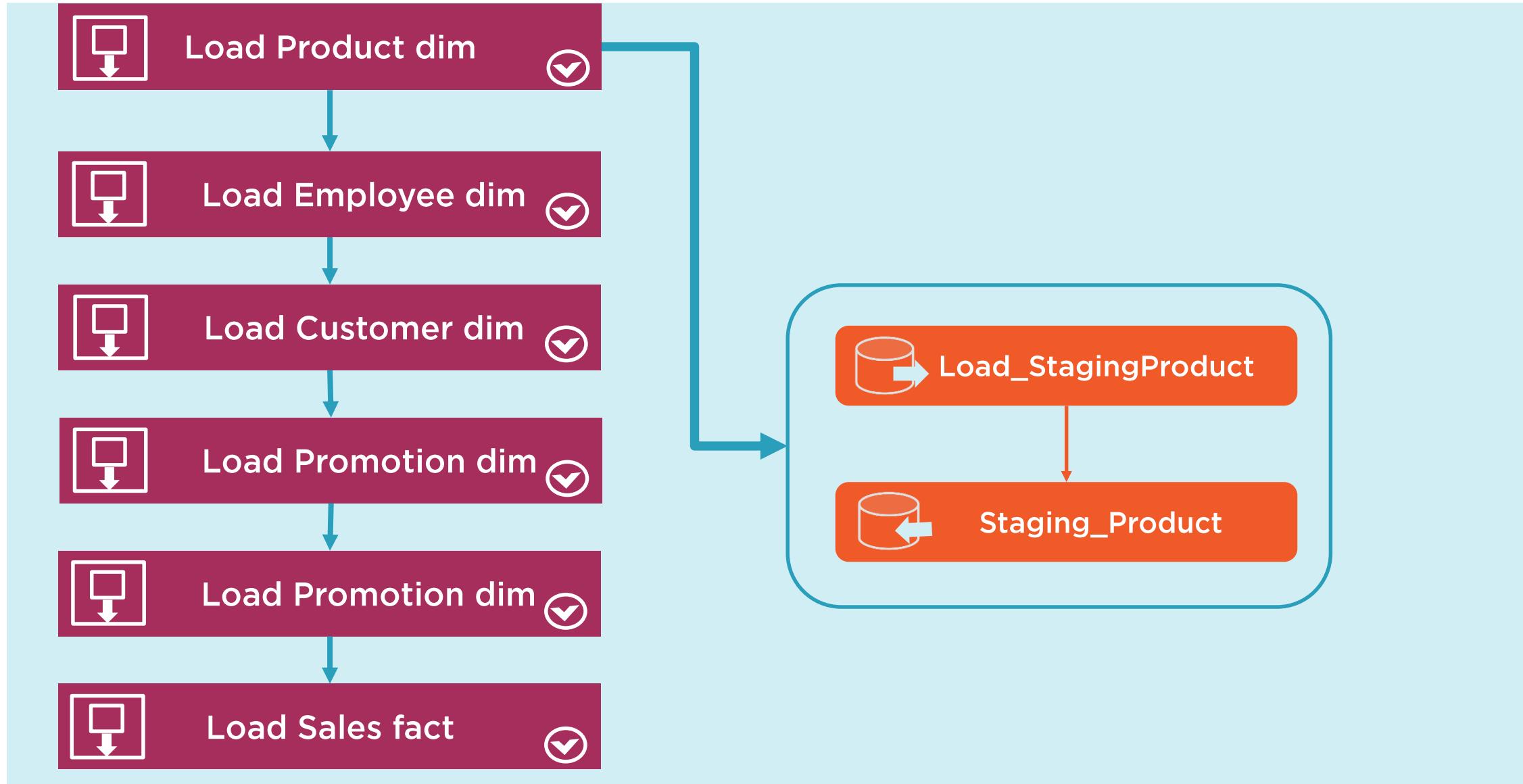
# Overview of a SSIS Package



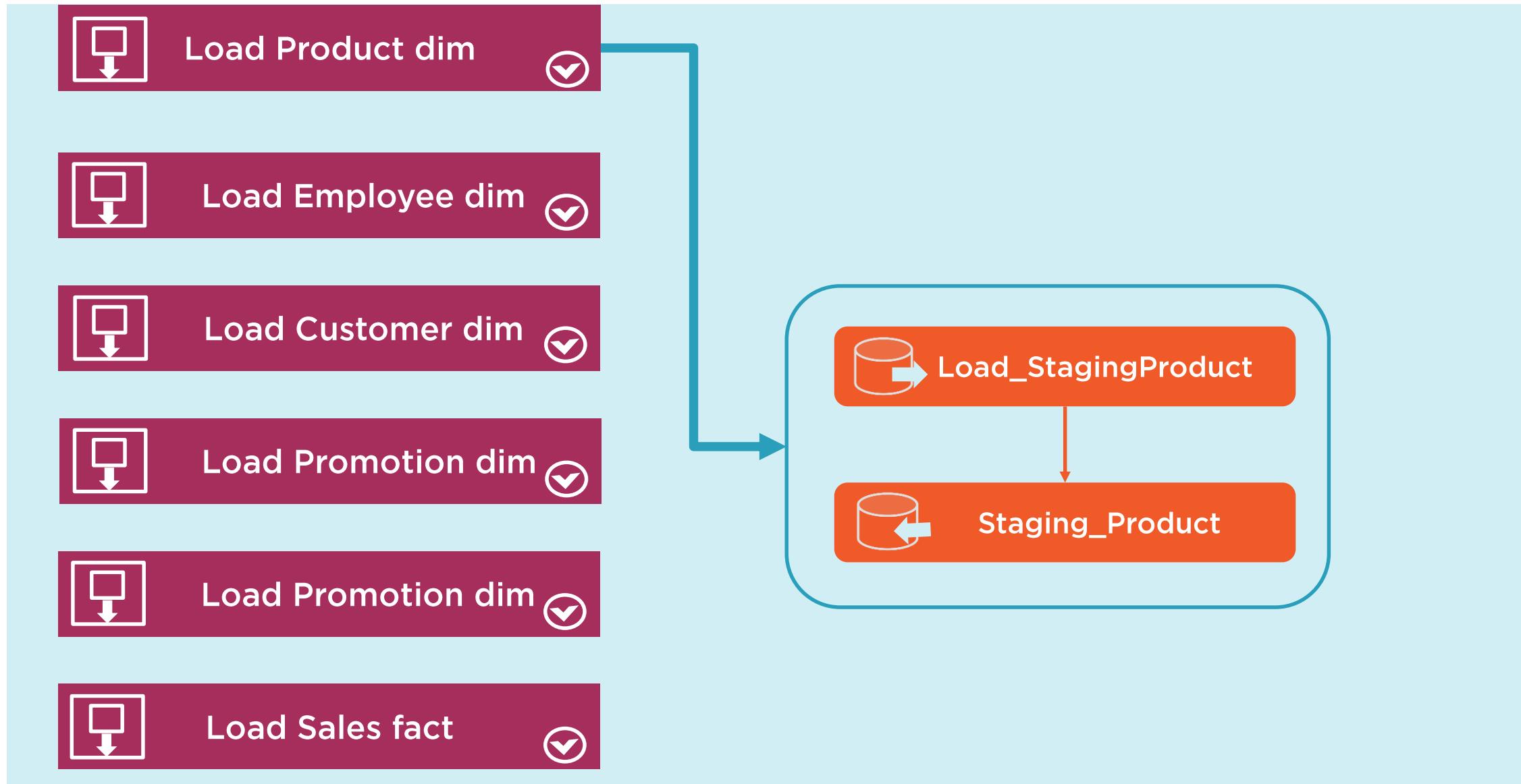
# Overview of a SSIS Package



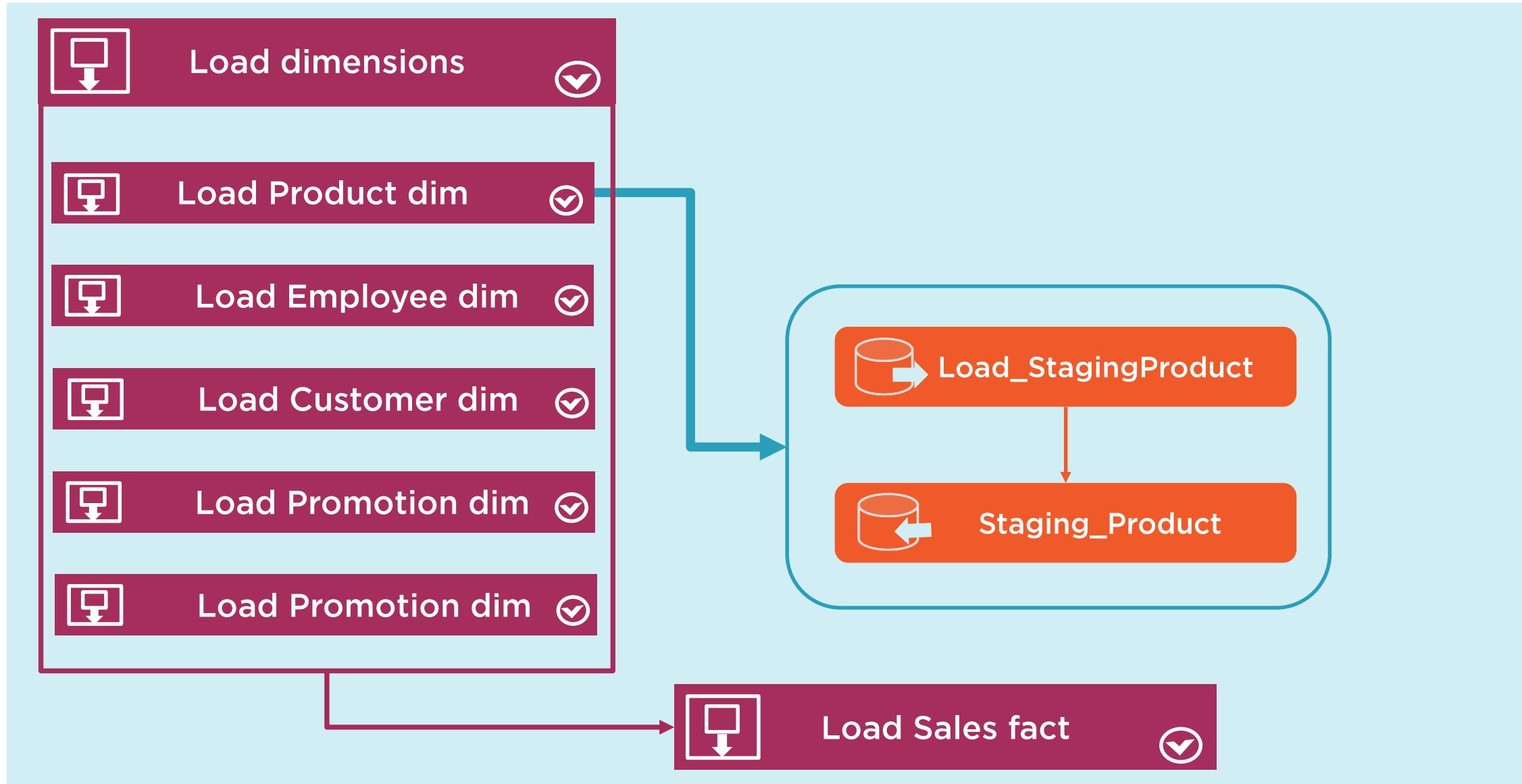
# Overview of a SSIS Package



# Overview of a SSIS Package



# Overview of a SSIS Package



# SSIS Terminology




# SSIS Terminology

<b>SSIS package</b>	



# SSIS Terminology

<b>SSIS package</b>	
<b>Task</b>	



# SSIS Terminology

<b>SSIS package</b>	
<b>Task</b> <b>Control flow task</b>	



# SSIS Terminology

<b>SSIS package</b>	
<b>Task</b> <b>Control flow task</b> <b>Data flow task</b>	



# SSIS Terminology

<b>SSIS package</b>	
<b>Task</b> <b>Control flow task</b> <b>Data flow task</b>	
<b>Control flow</b>	



# SSIS Terminology

<b>SSIS package</b>	
<b>Task</b> <b>Control flow task</b> <b>Data flow task</b>	
<b>Control flow</b>	
<b>Data flow</b>	



# SSIS Terminology

<b>SSIS package</b>	
<b>Task</b> <b>Control flow task</b> <b>Data flow task</b>	
<b>Control flow</b>	
<b>Data flow</b>	
<b>Connection manager</b>	



# SSIS Terminology

SSIS package	Container
Task Control flow task Data flow task	
Control flow	
Data flow	
Connection manager	



# SSIS Terminology

SSIS package	Container
Task Control flow task Data flow task	Precedence constraint
Control flow	
Data flow	
Connection manager	



# SSIS Terminology

<b>SSIS package</b>	<b>Container</b>
<b>Task</b> <b>Control flow task</b> <b>Data flow task</b>	<b>Precedence constraint</b>
<b>Control flow</b>	<b>Variable</b>
<b>Data flow</b>	
<b>Connection manager</b>	

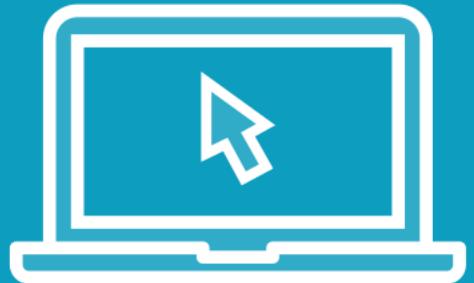


# SSIS Terminology

<b>SSIS package</b>	<b>Container</b>
<b>Task</b> <b>Control flow task</b> <b>Data flow task</b>	<b>Precedence constraint</b>
<b>Control flow</b>	<b>Variable</b>
<b>Data flow</b>	<b>Parameter</b>
<b>Connection manager</b>	



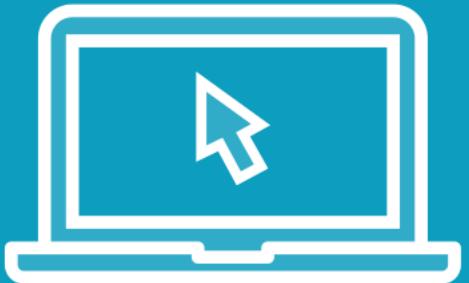
Demo



**Creating a new SSIS project**



# Demo



## Creating connection managers

- This is one of the first tasks to do in a new project
- Almost all components in SSIS need a connection



Demo



## Creating SSIS components for loading Product dimension

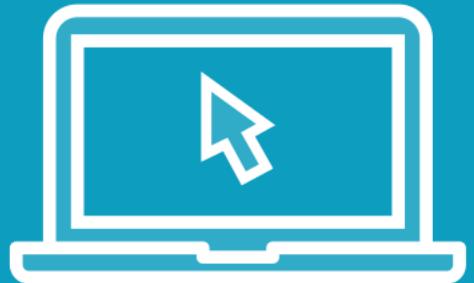
- Connection managers
- Tasks
- Precedence constraints
- Containers

### The package will do the following:

- Update Lineage and Incremental loads tables
- Transfer data into the staging table
- Update the dimension table



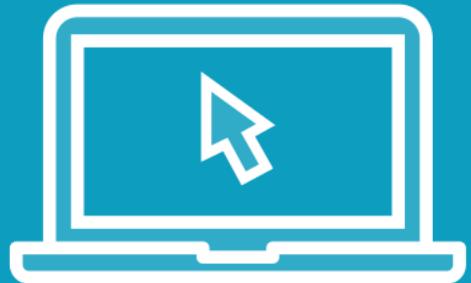
Demo



**Creating and working with variables**



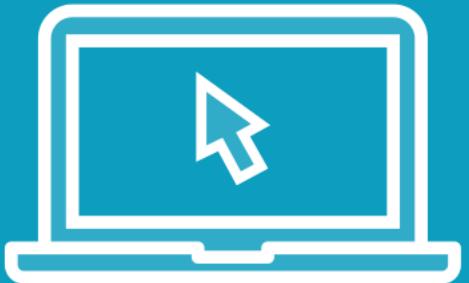
Demo



**Creating precedence constraints**



Demo



**Executing a SQL task with parameters**



# Practical Exercise

Retrieve the date of the previous load

- Create a SQL task that executes the stored procedure **[int].[Get\_LastLoadedDate]**
- Save the result in the variable: **[User::PreviousLoadDate]**
- Execute the task after the staging table is truncated
- Rename the procedure “SQL Get date of the previous load”



# Summary



**Advantages of using ETL tools**

**High-level overview of SSIS**

**Creating an SSIS project**

**Configuring connections to the data sources**

**Creating control flow tasks**

**Setting up the order of execution with precedence constraints**

**Creating and working with package variables**



# Loading a Data Warehouse with SSIS

---



**Ana Voicu**  
@ana\_voicu



# Overview



**Design data flow elements**

**Organize control flow tasks in containers**

**Add package parameters**

**Change the properties of a task using expressions**

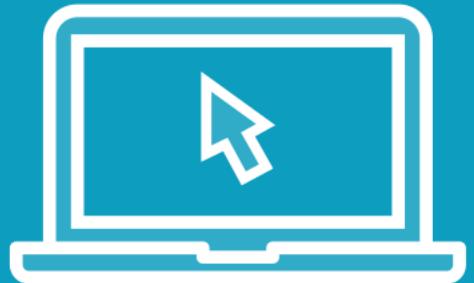
**Deploy and execute the SSIS project in SQL Server**

**The result is a project that:**

- Loads the data warehouse (full or incremental load)
- Can be customized for different customers with parameters
- Can be executed automatically, from a SQL Agent job



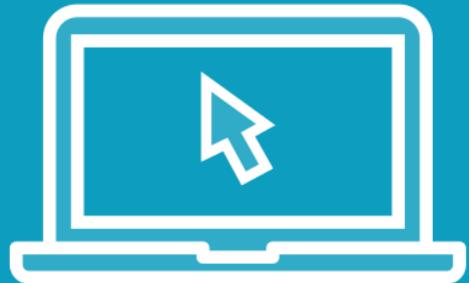
Demo



**Designing data flow elements**



Demo



Organizing data with containers



# What Are Containers?



**SSIS components that organize tasks into units**

**Three types of containers:**

- For loop
- Foreach loop
- Sequence container



# Description of Each Container Type



## For loop

- The tasks are executed repeatedly, based on an expression

## Foreach loop

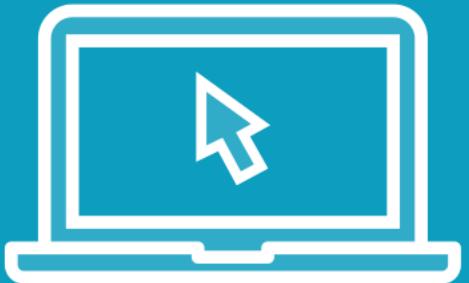
- Iterate through a collection of objects

## Sequence container

- Treats the elements as a whole
- If a task fails, the entire container fails
- You cannot add a precedence constraint between a task from the container and a task outside of it
- “All for one and one for all” behavior



# Demo



## Working with package parameters

- What are parameters?
- Types of parameters
- How to create parameters?
- The difference between parameters and variables



# SSIS Parameters



**Assign values to properties at package execution**

**Can be set:**

- At package level
- At project level

**Choosing what type to use depends on the deployment strategy for the project**

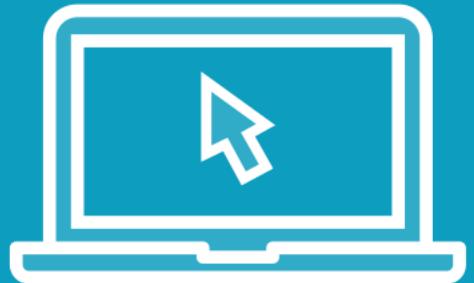
**Create the link between the outside world and the content within SSIS**

**Examples:**

- Database server name
- Type of load (full or incremental)



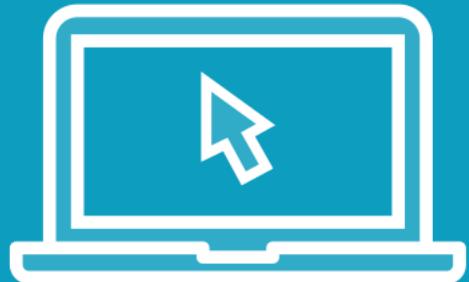
Demo



**Adding expressions to tasks**



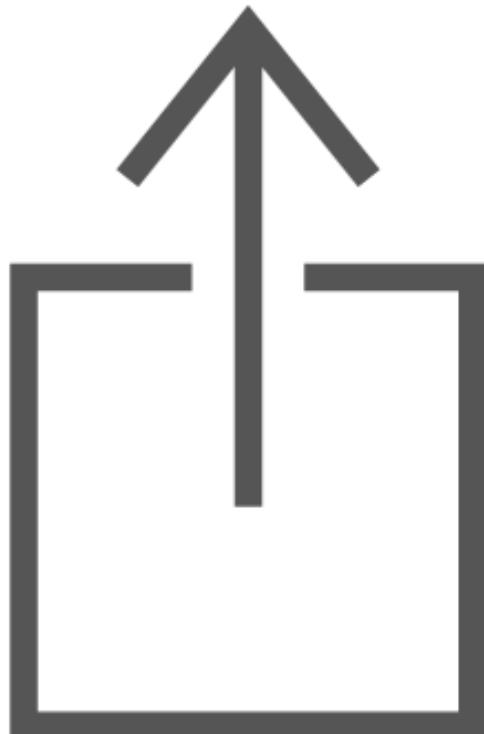
Demo



**Deploying and configuring a SSIS project**



# SSIS Deployment Model



**The current deployment model is available starting with SQL Server 2012**

**Everything SSIS-related is centralized in one place**

- Project and package parameters
- Execution runtime values
- Execution status and other troubleshooting information
- General information about SSIS

**The centralized location is called SSISDB catalog**



# Summary



**Understand the difference between control flow and data flow**

**Created data flow components**

**Created tasks to load all the dimensions and fact tables**

**Organized tasks in containers**

**Added package parameters**

**Used expressions to change the properties of a task at package execution**

**Deployed an SSIS project in SQL Server**



# Visualizing Data from the Data Warehouse

---



**Ana Voicu**  
@ana\_voicu



# Project Status Update

Task	Status
------	--------



# Project Status Update

Task	Status
Create data warehouse	Done



# Project Status Update

Task	Status
Create data warehouse	Done
Create load process in SSIS	Done



# Project Status Update

Task	Status
Create data warehouse	Done
Create load process in SSIS	Done
Expose data and reports to end-users	Not done



# Overview



## Adding a semantic layer on top of the data warehouse

- A semantic layer is a structure that:
  - Reduces the complexity of the data warehouse
  - Shows only what is relevant
  - Creates pre-calculated measures, to improve performance
- SQL Server Analysis Services is used
  - Multidimensional model
  - Tabular model

## Microsoft reporting tools:

- SQL Server Reporting Services
- Power BI
- Power BI Report Server



# Creating a Semantic Layer

---



# Stages of Creating a BI Solution



## **Summary of the previous steps**

- Gathered business requirements
- Created the dimensional model
- Implemented the data warehouse in SQL Server
- Extended the DW with auxiliary tables
- Created the load process

## **Options to move forward**

- a) Create the visualization layer directly from the data warehouse
- b) Create a semantic layer



# Semantic layer

“It is a business representation of corporate data that helps end-users access data autonomously using common business terms.

A semantic layer maps complex data into familiar business terms such as product, customer, or revenue to offer a unified, consolidated view of data across the organization.” (Wikipedia)



# Clarifying the Confusion



**Isn't that what the dimensional design is doing?**

- Yes, it is

**The dimensional design**

- Is an abstraction layer on top of different data sources
- Can be extended and simplified even more with a semantic model on top of it
- There are some advantages for creating the semantic model

# Advantages of Using a Semantic Layer



# Advantages of Using a Semantic Layer



# Advantages of Using a Semantic Layer

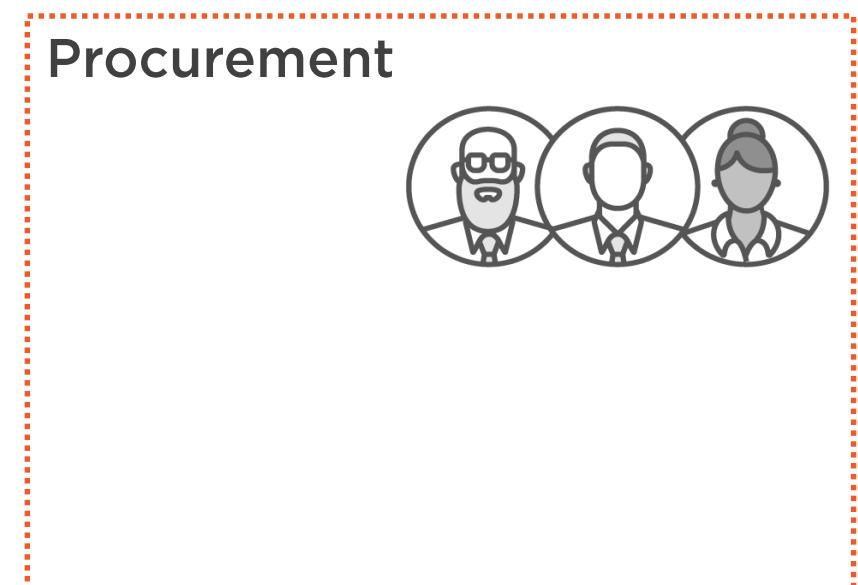


**Sales**

**Procurement**



# Advantages of Using a Semantic Layer



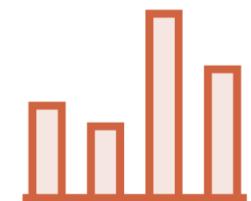
# Advantages of Using a Semantic Layer



Sales



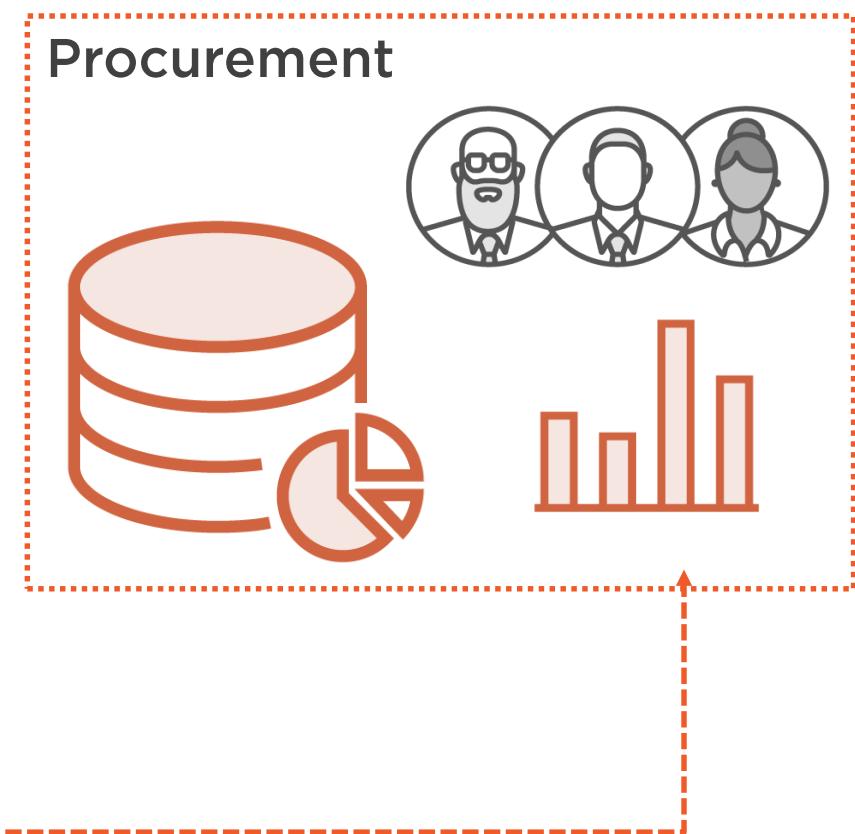
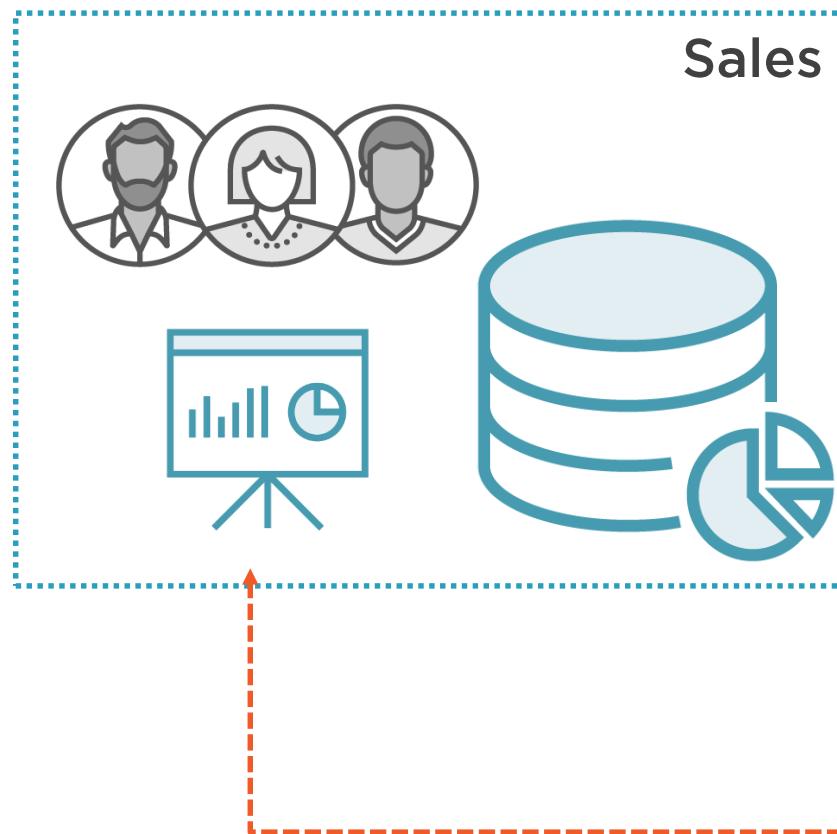
Procurement



# Advantages of Using a Semantic Layer



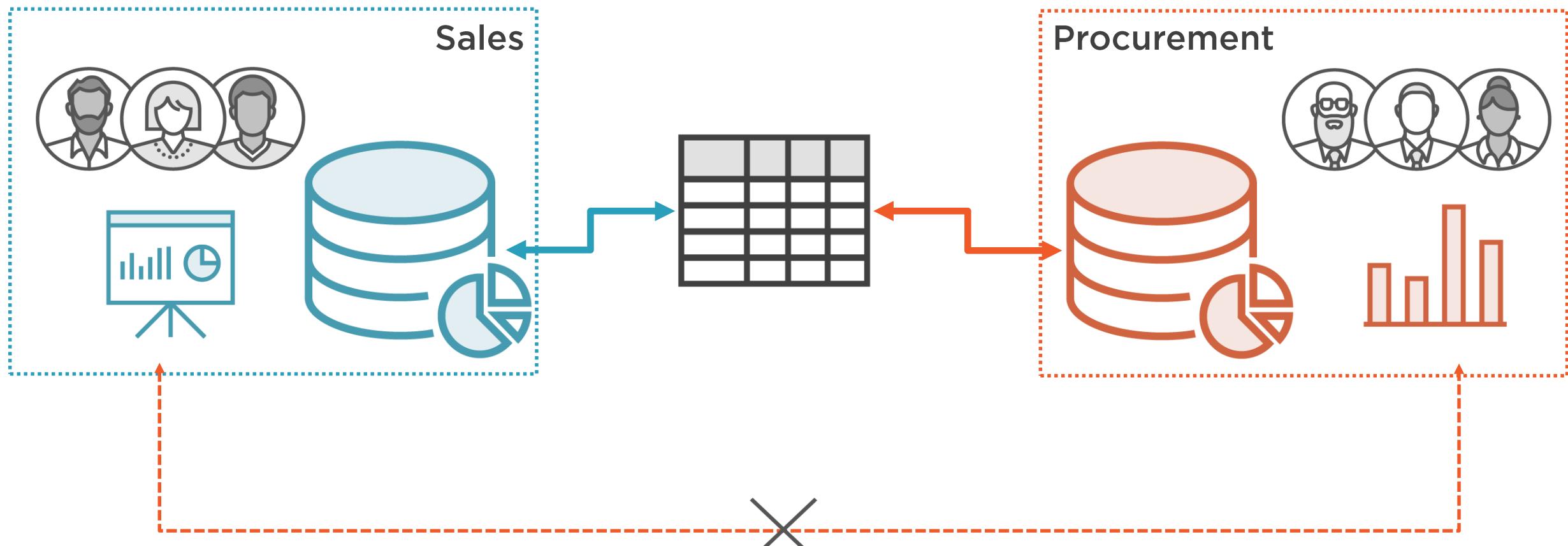
# Advantages of Using a Semantic Layer



Sensitive data should not be shared  
between departments



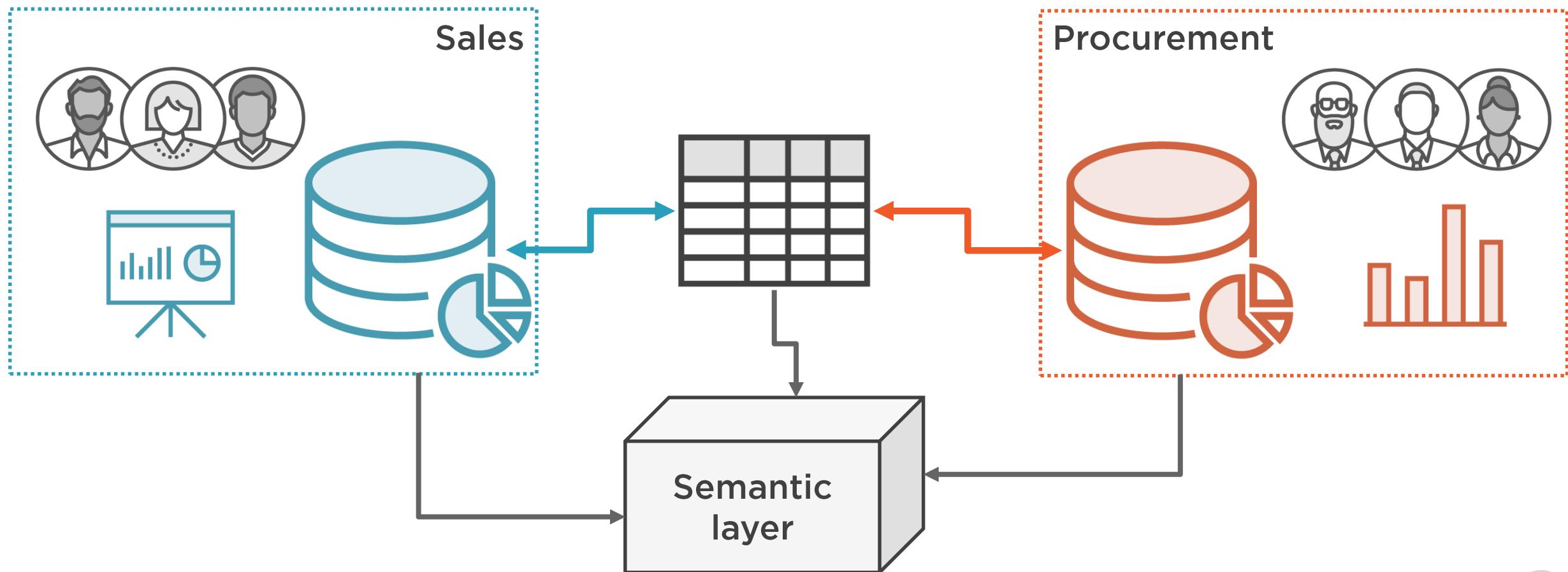
# Advantages of Using a Semantic Layer



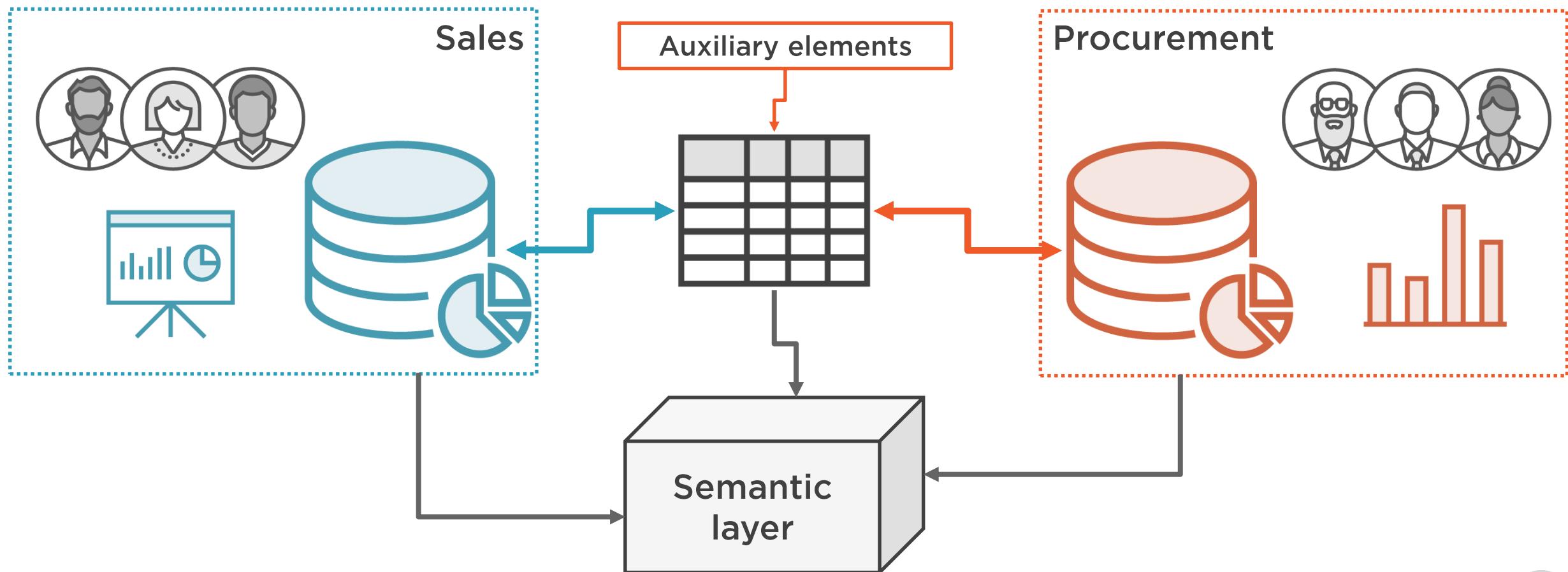
Sensitive data should not be shared  
between departments



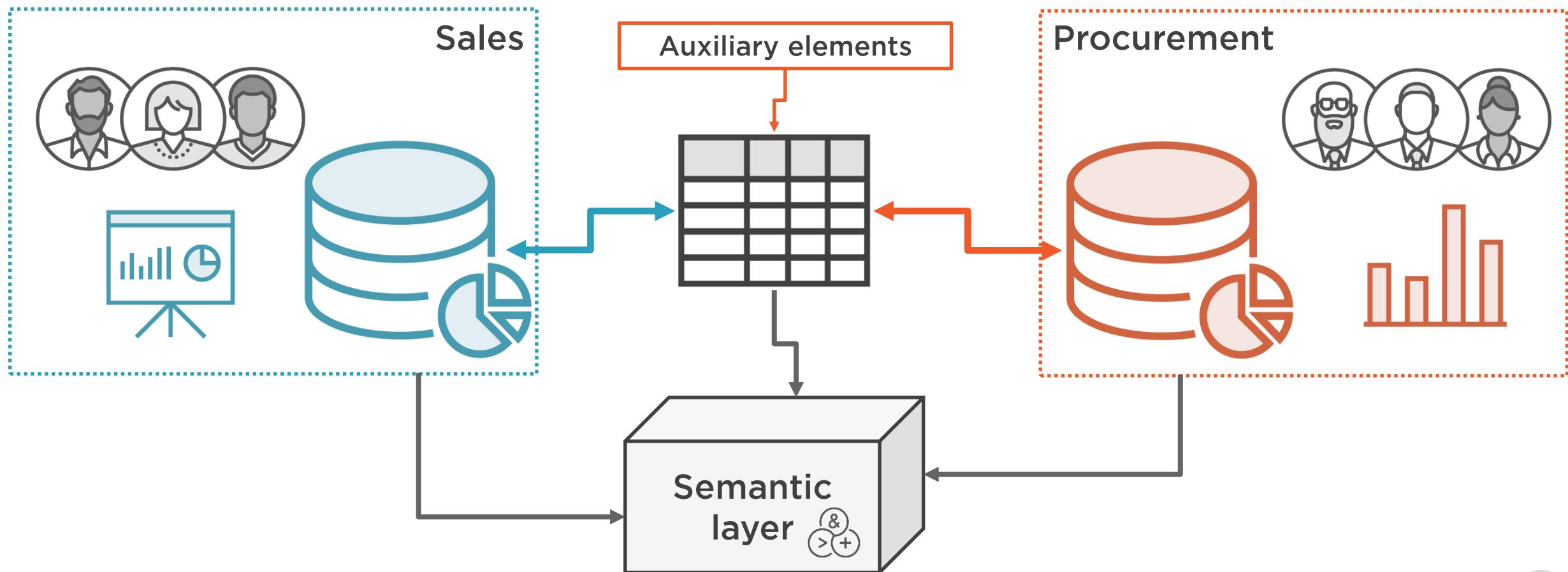
# Advantages of Using a Semantic Layer



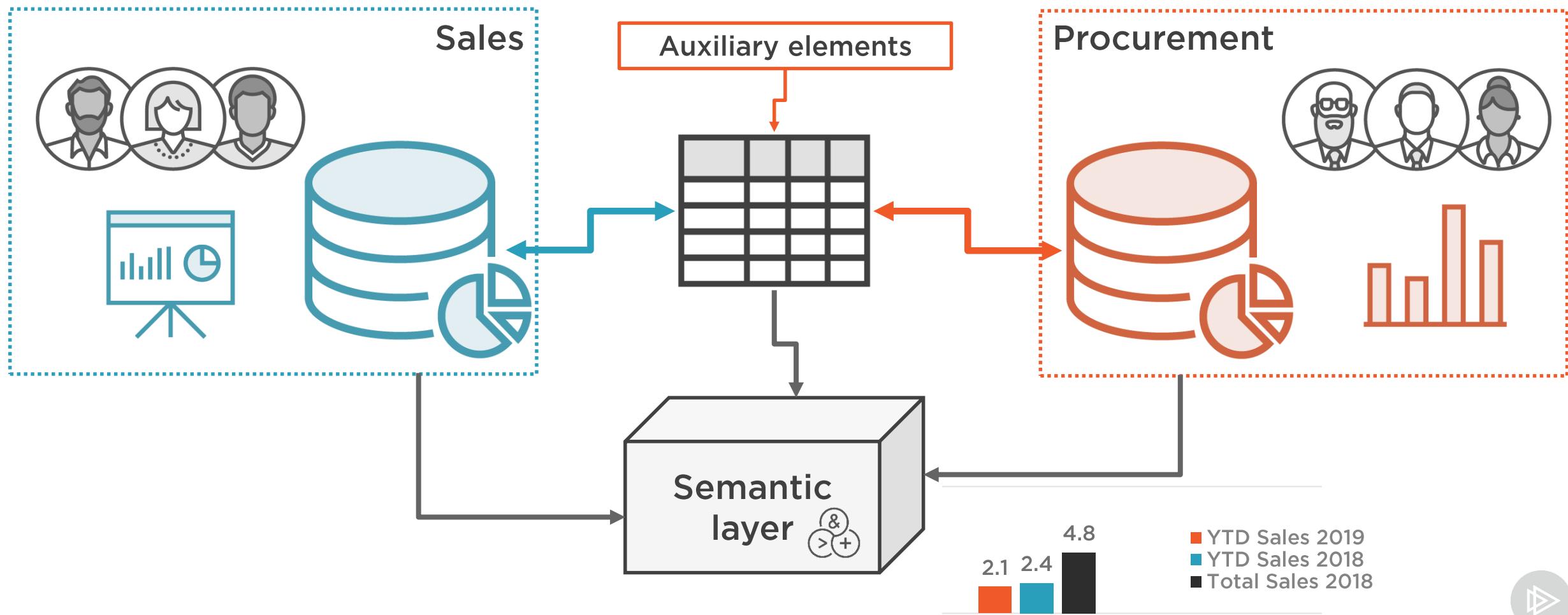
# Advantages of Using a Semantic Layer



# Advantages of Using a Semantic Layer



# Advantages of Using a Semantic Layer



# Advantages of Using a Semantic Layer

Show only  
relevant data

Faster query  
performance

Complex  
aggregations

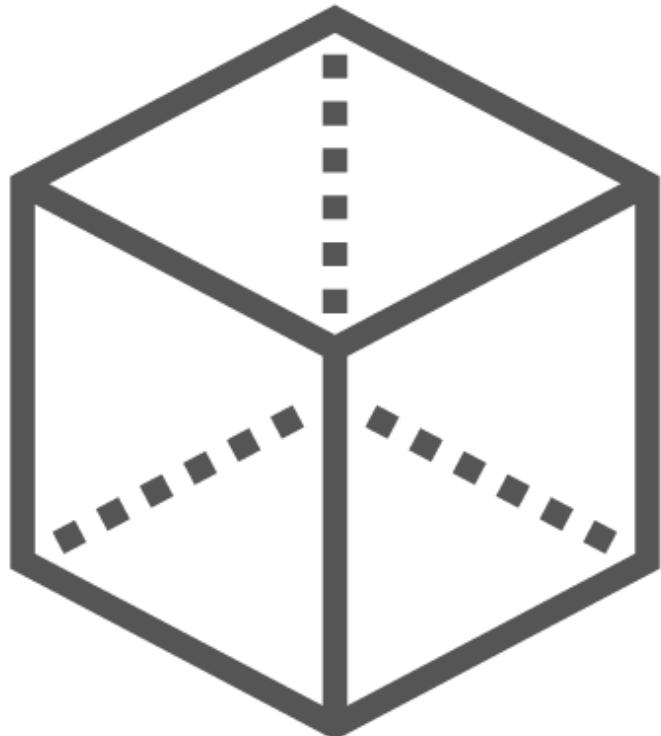
Readable names  
for objects

Consolidate data  
from multiple  
source

Integration with  
many BI tools



# Types of Models in SSAS



**Multidimensional model**  
**Tabular model**

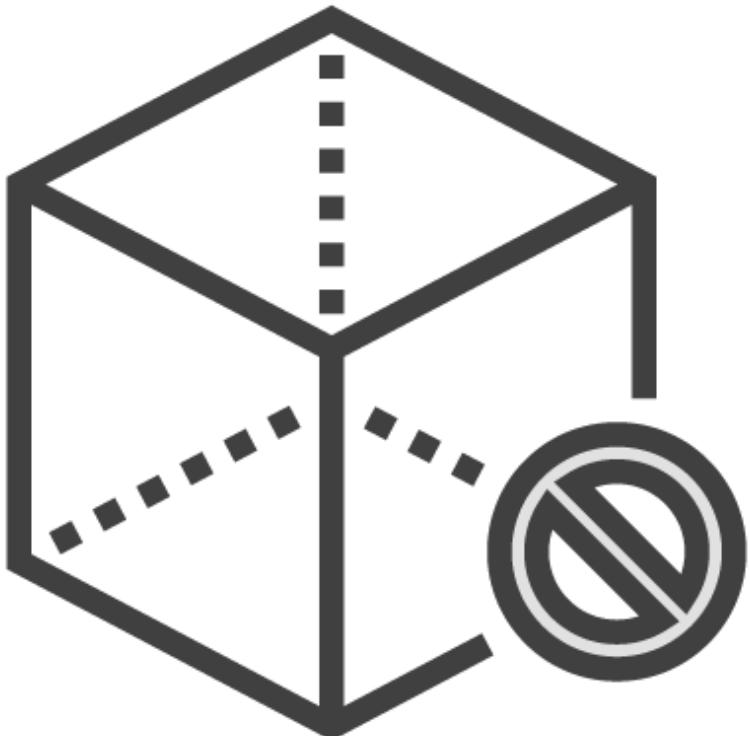


# The Multidimensional and Tabular Models

---



# The Multidimensional Model



**Similar to the star schema design**

**The main structure used for data analysis is the cube**

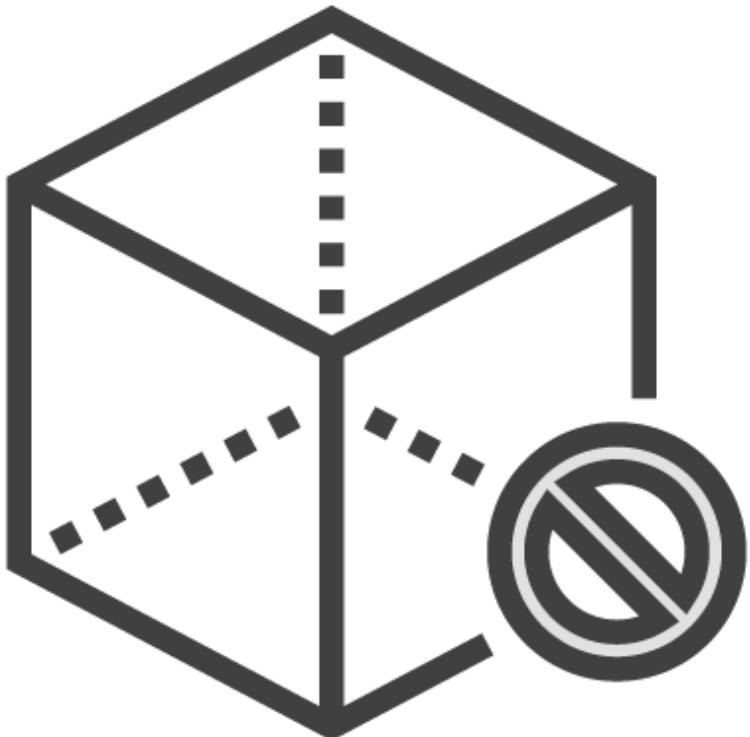
**A cube consists of**

- Dimensions
- Measure groups

**Data sources must be defined before creating other elements**



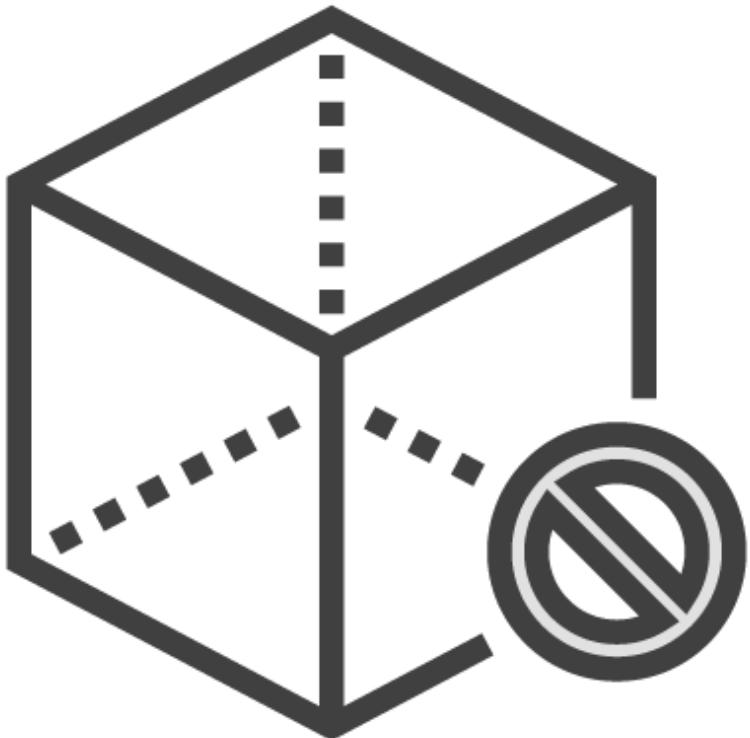
# The Multidimensional Model



<b>Multidimensional object</b>	<b>Data warehouse object</b>
<b>Dimension</b>	<b>Dimension table</b>



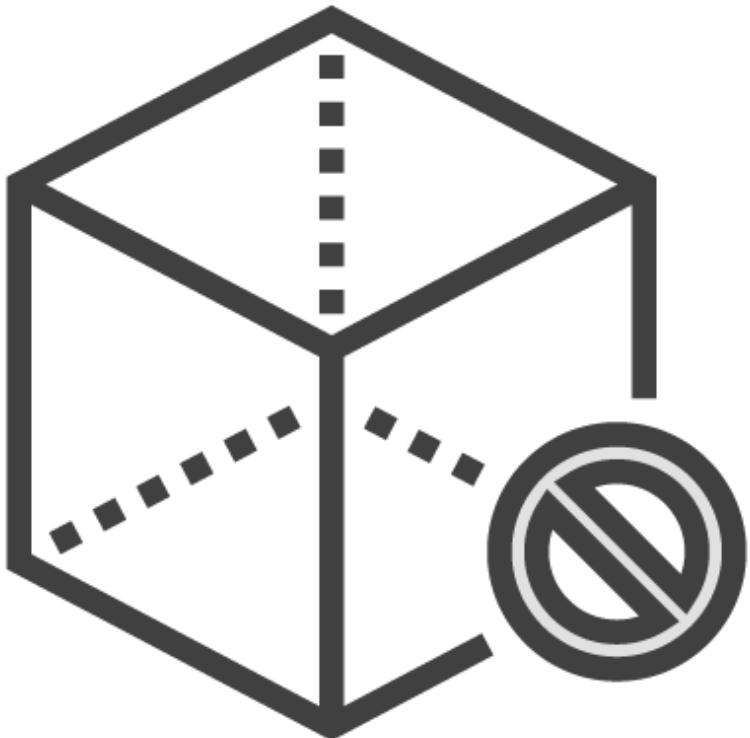
# The Multidimensional Model



Multidimensional object	Data warehouse object
Dimension	Dimension table
Measure group	Fact table



# The Multidimensional Model

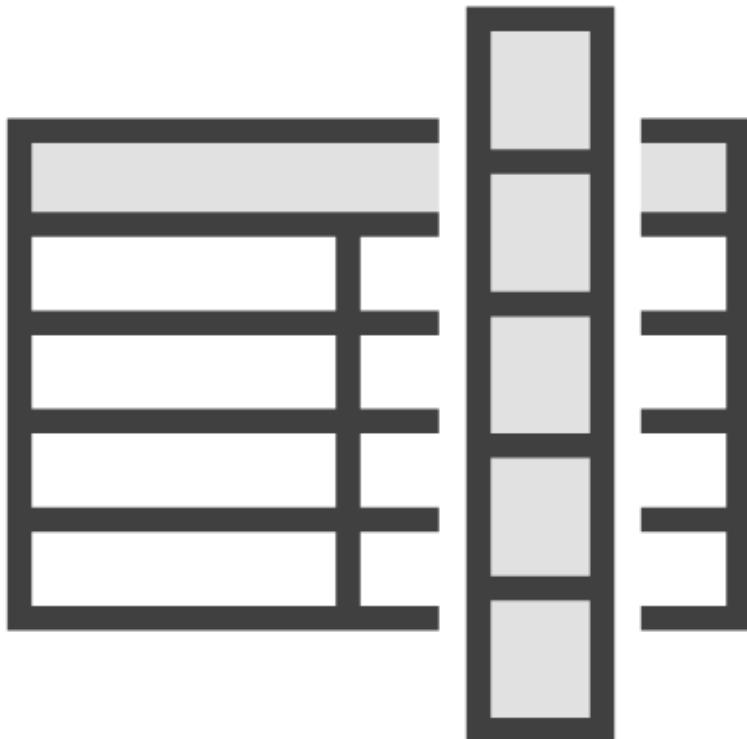


Multidimensional object	Data warehouse object
Dimension	Dimension table
Measure group	Fact table
Measure	Column in the fact table

New measures can be calculated in this database



# The Tabular Model



**In-memory database**

**Connects directly to a relational source of data**

**Similar to both SQL Server relational databases and a SSAS cube**

- Consists of tables and relationships between tables
- You can also create measures and KPIs

**The database can be used in Excel, Reporting Services, Power BI, etc.**



# Multidimensional vs. Tabular Model

## Multidimensional

Better option when working with large amounts of data

Better performance in terms of scalability

Easily create and work with parent-child hierarchies

Writeback capabilities

## Tabular

Easier for developers to understand and implement the model

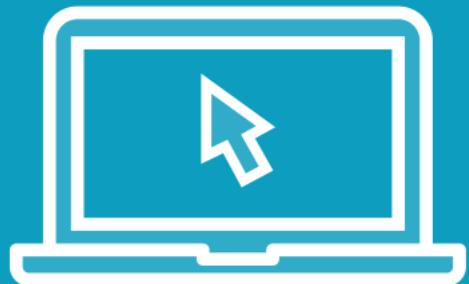
DAX language

Doesn't require strong hardware (disk) capabilities. It is only memory-dependent

The database is much smaller than the original source of data



# Demo



## **Creating a multidimensional database in SSAS**

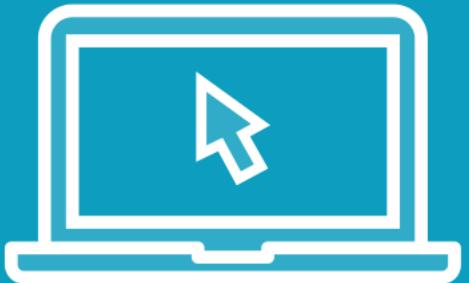
- This is a separate type of database

## **Prerequisite for this demo**

- Install SQL Server Analysis Services in multidimensional mode



Demo



Analyzing data from a SSAS database



# Creating Interactive Reports

---



# SQL Server Reporting Services

## Paginated reports

- The “traditional” reports
- Data is organized in tables, spread on multiple pages
- Optimized for printing or saving as Word or PDF

## Mobile reports

- Compatible with several mobile devices
- Connect to different data sources (including SQL Server and SSAS)
- The visualizations are not very diverse, but they still have a modern look and feel

## Web portal

- The portal can be accessed from any browser
- Types of reports: Power BI, mobile, paginated, KPIs, Excel files
- The data sets used in reports can be accessed from the portal



# Power BI



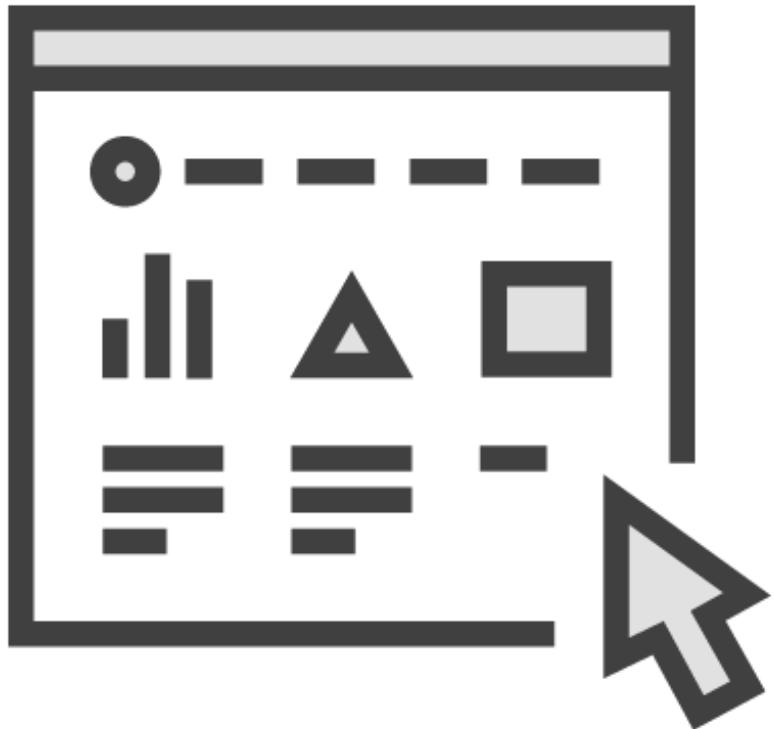
Tool used to create rich and interactive reports

Components:

- **Power BI Desktop**
  - Download and use for free, to create reports
  - Collaborative editing is not supported
- **Power BI Service**
  - Cloud service
  - Share reports with people within/external to the organization
- **Power BI mobile app**



# Power BI Report Server



**Reporting capabilities of Power BI**

**Data remains on-premises**

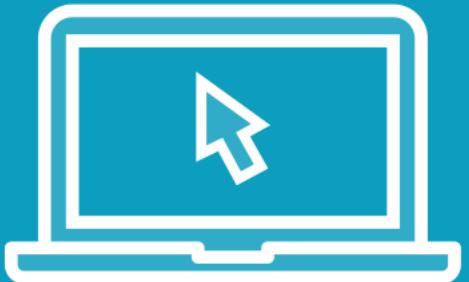
**Users can visualize the reports**

- In the web portal
- On mobile devices
- Shared by email

**Available in SQL Server Enterprise Edition,  
or with Power BI Premium license**



Demo



## Creating and publishing a report with Power BI Report Server

You need to install two components

- Power BI Report Server
- Power BI Desktop for Report Server
- Download them from the Microsoft Download Center

Power BI Report Server is available in

- SQL Server Enterprise Edition
- SQL Server Developer Edition



# Course Summary



## Fundamental concepts of a data warehouse

- Purpose of a data warehouse
- Facts and fact tables
- Dimensions and dimension tables
- Steps of the dimensional design process

## Dimension table techniques

## Hierarchies

## Fact tables

- Difference between fact and dimension tables
- Common types of fact tables



# Course Summary



## Creating the SQL scripts for loading a dimension table

- Create the Lineage and Incremental loads tables
- Create the stored procedures for loading data

## Implementing a load process in SSIS

- Create a new SSIS project in Visual Studio
- Work with control flow and data flow tasks
- Add parameters
- Deploy the project in SQL Server
- Create a SQL Agent job to run the project automatically

## Multidimensional and tabular models

## Creating reports on top of the data warehouse



Thank you!

