Dimensions types

LAB5

Outline

No	Dimensions Types topics
1	Conformed Dimension
2	Degenerate Dimension
3	Junk Dimension (Garbage Dimension)
4	Role-Playing Dimension
5	Outrigger Dimension
6	Shrunken Rollup Dimension
7	Swappable Dimension
8	Slowly changing Dimension
9	Fast Changing Dimension
10	Heterogenous Dimensions
11	Multi-valued dimensions

Conformed Dimension

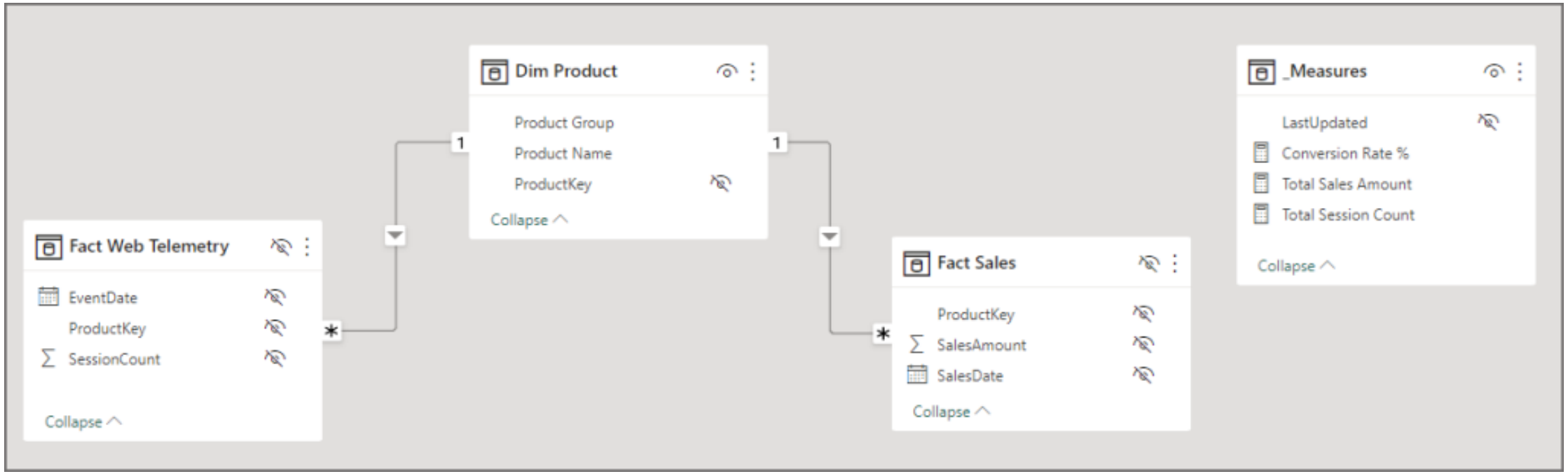
A dimension that can be **shared** across different facts (different business processes).

A dimension that returns a distinct list of values that have been mapped together from source systems.

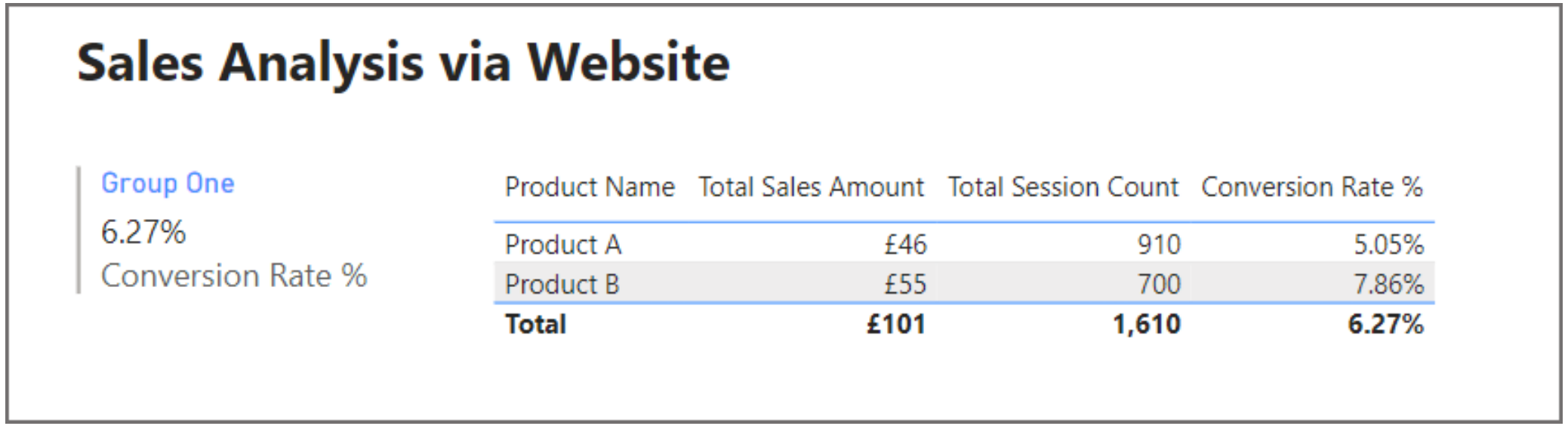
Conformed Dimension

As long as the Dimension means the same thing across these multiple Fact tables (and has the same surrogate keys connecting the fact tables), then we can say that Dimension is conformed.

Conformed Dimension Example



Conformed Dimension Example



Sales Analysis via Website

Group One

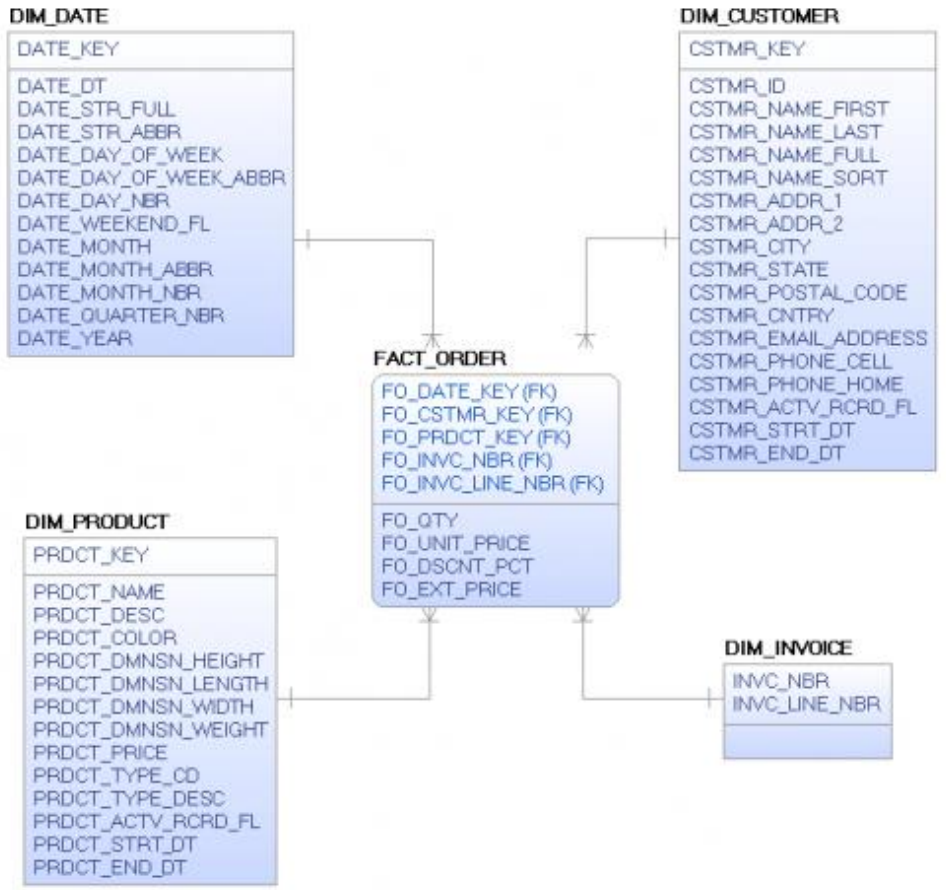
6.27%

Conversion Rate %

Product Name	Total Sales Amount	Total Session Count	Conversion Rate %
Product A	£46	910	5.05%
Product B	£55	700	7.86%
Total	**£101**	**1,610**	**6.27%**

Degenerate Dimension

Sometimes a dimension is defined that has no content except for its primary key.



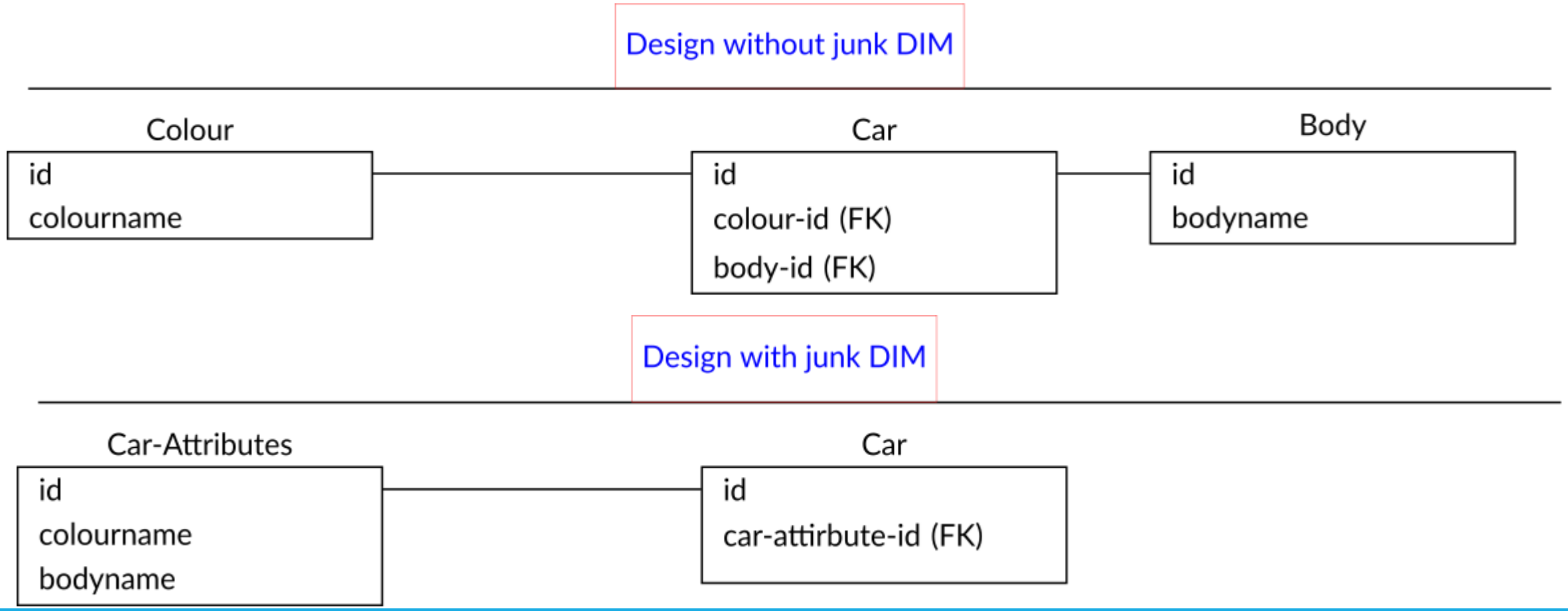
Junk Dimension (Garbage Dimension)

A junk dimension combines two or more related low cardinality flags into a single dimension.

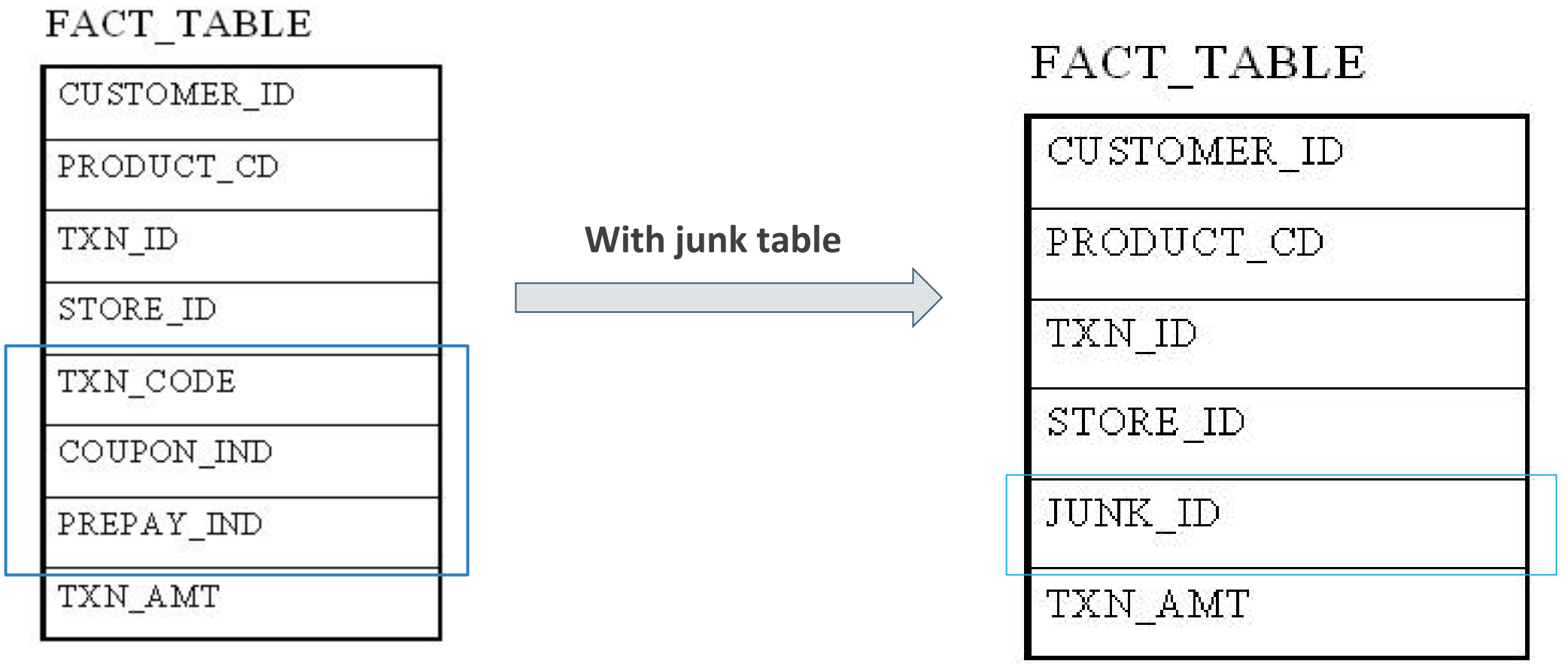
An example of this may be car color (red, black, blue, etc.) and body style (sedan, van, SUV, etc.).

The content in the junk dimension table is the combination of all possible values of the individual indicator fields.

Junk Dimension Example



Junk Dimension Example



Junk Dimension Example

DIM_JUNK

JUNK_ID	TXN_CODE	COUPON_IND	PREPAY_IND
1	1	Y	Y
2	2	Y	Y
3	3	Y	Y
4	1	Y	N
5	2	Y	N
6	3	Y	N
7	1	N	Y
8	2	N	Y
9	3	N	Y
10	1	N	N
11	2	N	N
12	3	N	N

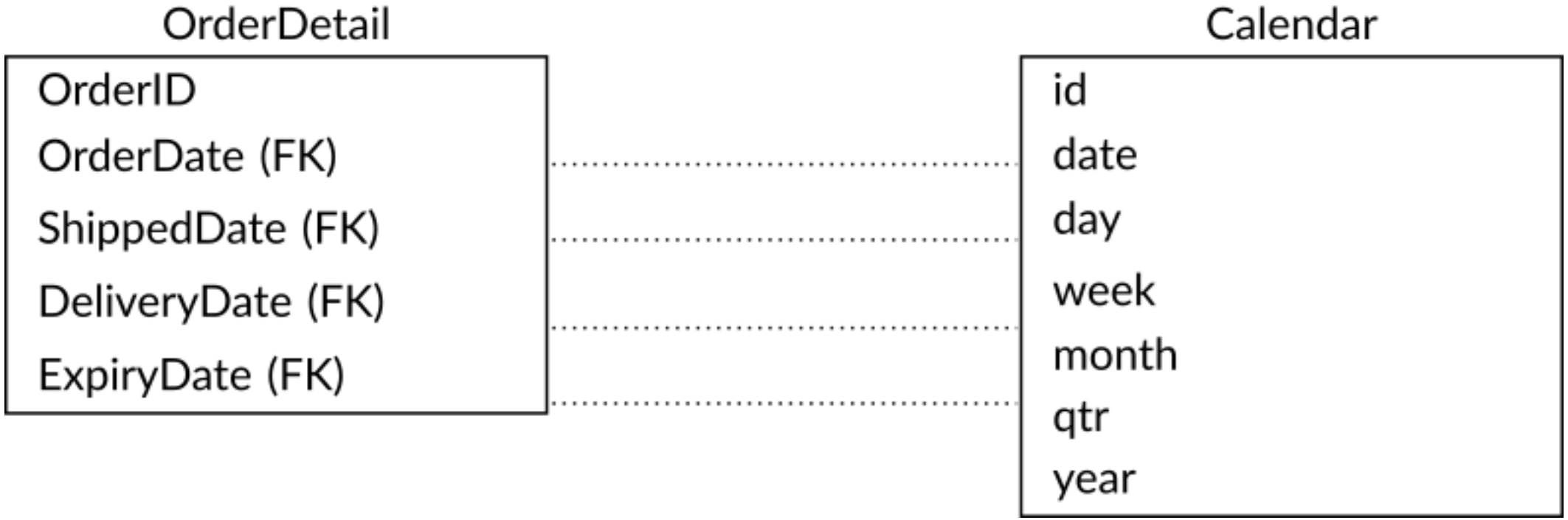
Role-Playing (Re-usable) Dimension

A single physical dimension can be referenced multiple times in a fact table, with each reference linking to a logically **distinct role** for the dimension.

For instance, a fact table can have several dates, each of which is represented by a foreign key to the date dimension.

It is essential that each foreign key refers to a separate view of the date dimension so that the references are independent. These separate dimension views (with unique attribute column names) are called **roles.**

Role-Playing Dimension Example



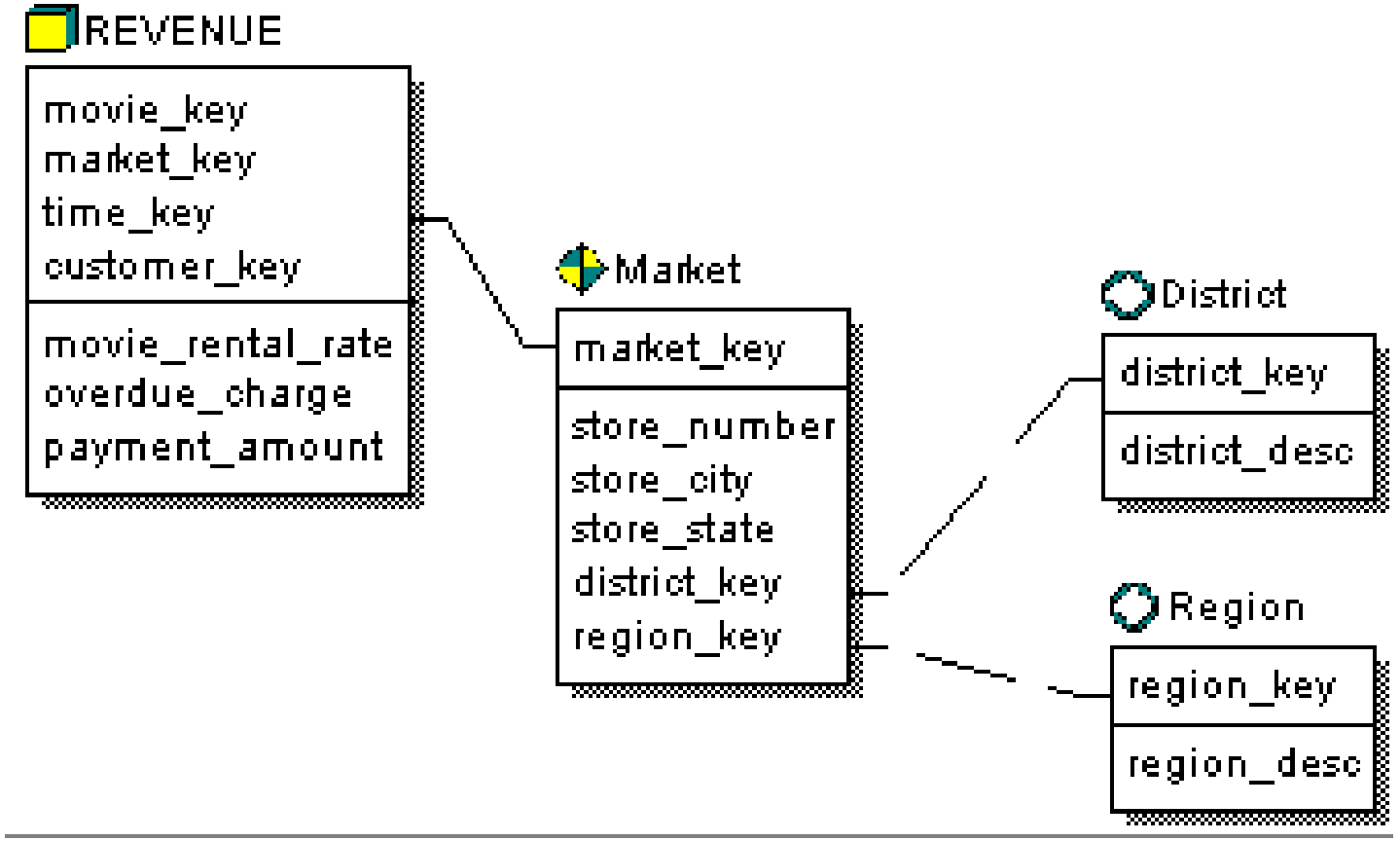
Outrigger Dimension

A dimension can contain a reference to another dimension table.

An outrigger entity is a secondary dimension entity attached to a main conformed dimension entity.

Outrigger entities must be used **sparingly** to avoid snowflaking.

Outrigger Dimension Example



Shrunken Rollup Dimension

Shrunken Rollup dimension is used for developing aggregate (higher level of summary) fact tables.

Shrunken Rollup Dimension Example

City and Area relationship dimension

AreaID	AreaName	CityID
123	Al-Matareya	1
678	Ain shams	1

City dimension

CityID	CityName
1	Cairo

Shrunken Rollup Dimension Example

Order detail per AreaID

OrderDate	AreaID	TotalOrders
123456789	123	20
123456789	123	30
123456789	678	10
123456789	678	12

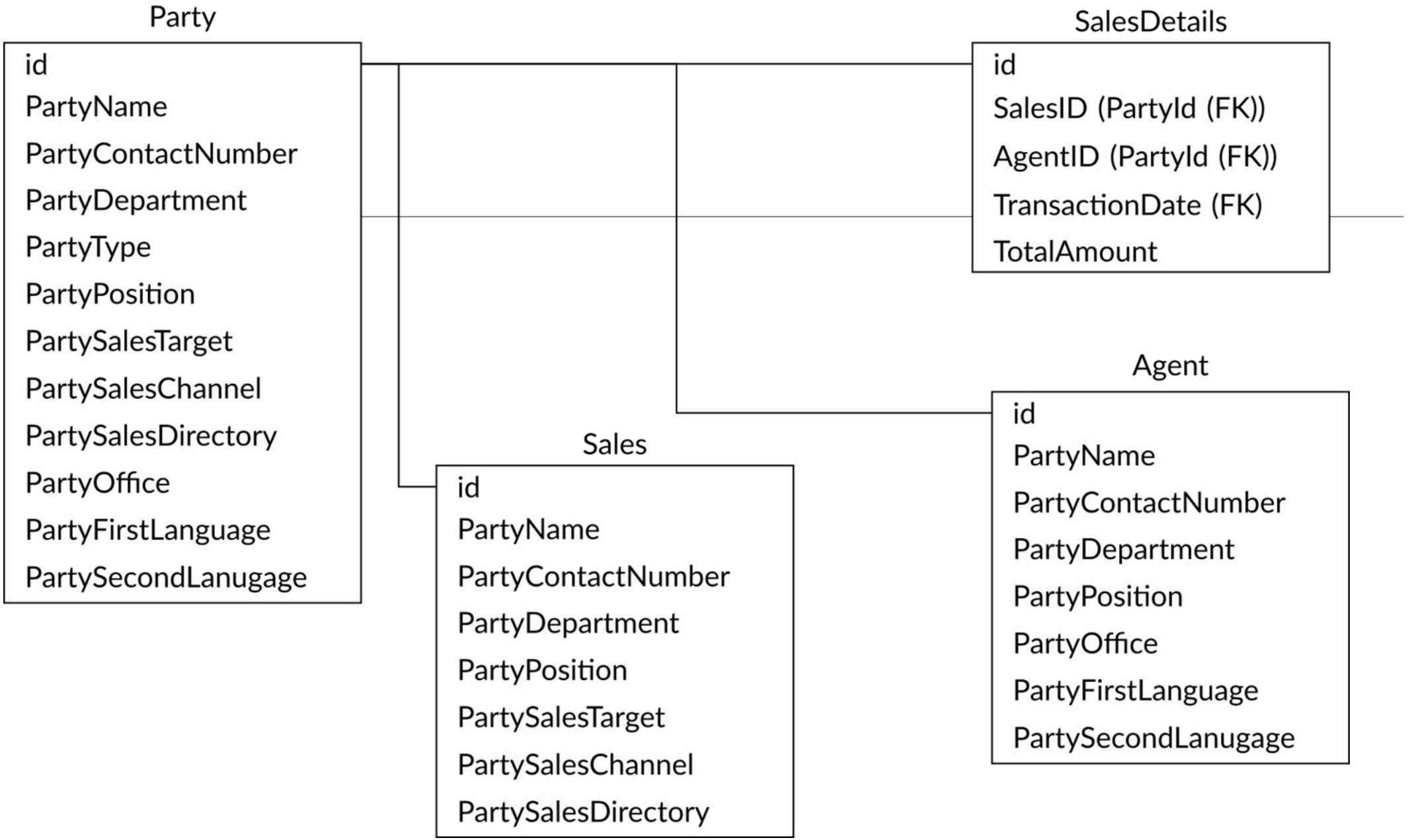
Order detail per CityID

OrderDate	CityID	TotalOrders
123456789	1	72

Swappable Dimension

A dimension that has multiple alternate versions of itself that can be swapped at query time.

- It has a different meaning
- It has a different structure.
- It has fewer data compared to the primary dimension (fewer rows and columns).
- It has a different output based on the input version and its alternatives.
- Multi versions could be used together in the same fact with different types.
- It can act as the primary dimension and join to the same fact table.
- It has different target users and sometimes we restrict the users to access the primary dimension and only access the swapped version to **restrict the data** without needs to show the whole primary attributes.


Party
id
PartyName
PartyContactNumber
PartyDepartment
PartyType
PartyPosition
PartySalesTarget
PartySalesChannel
PartySalesDirectory
PartyOffice
PartyFirstLanguage
PartySecondLanugage
SalesDetails
id
SalesID (PartyId (FK))
AgentID (PartyId (FK))
TransactionDate (FK)
TotalAmount
Agent
id
PartyName
PartyContactNumber
PartyDepartment
PartyPosition
PartyOffice
PartyFirstLanguage
PartySecondLanugage
Sales
id
PartyName
PartyContactNumber
PartyDepartment
PartyPosition
PartySalesTarget
PartySalesChannel
PartySalesDirectory

Swappable Dimensions Implementation

- **Direct join between Fact and Dimension** with filter based on PartyType (run-time). In this case Party includes some empty columns based on the type.
- **Logical views** each view has its own number of columns and rows based on the type details.

  Pros: Easy for (managing, implementation) with consistent views.

  Cons: Performance and manage the authorization per view.

- **Physical tables (Types & Subtypes).**

  Pros: Performance, better design.

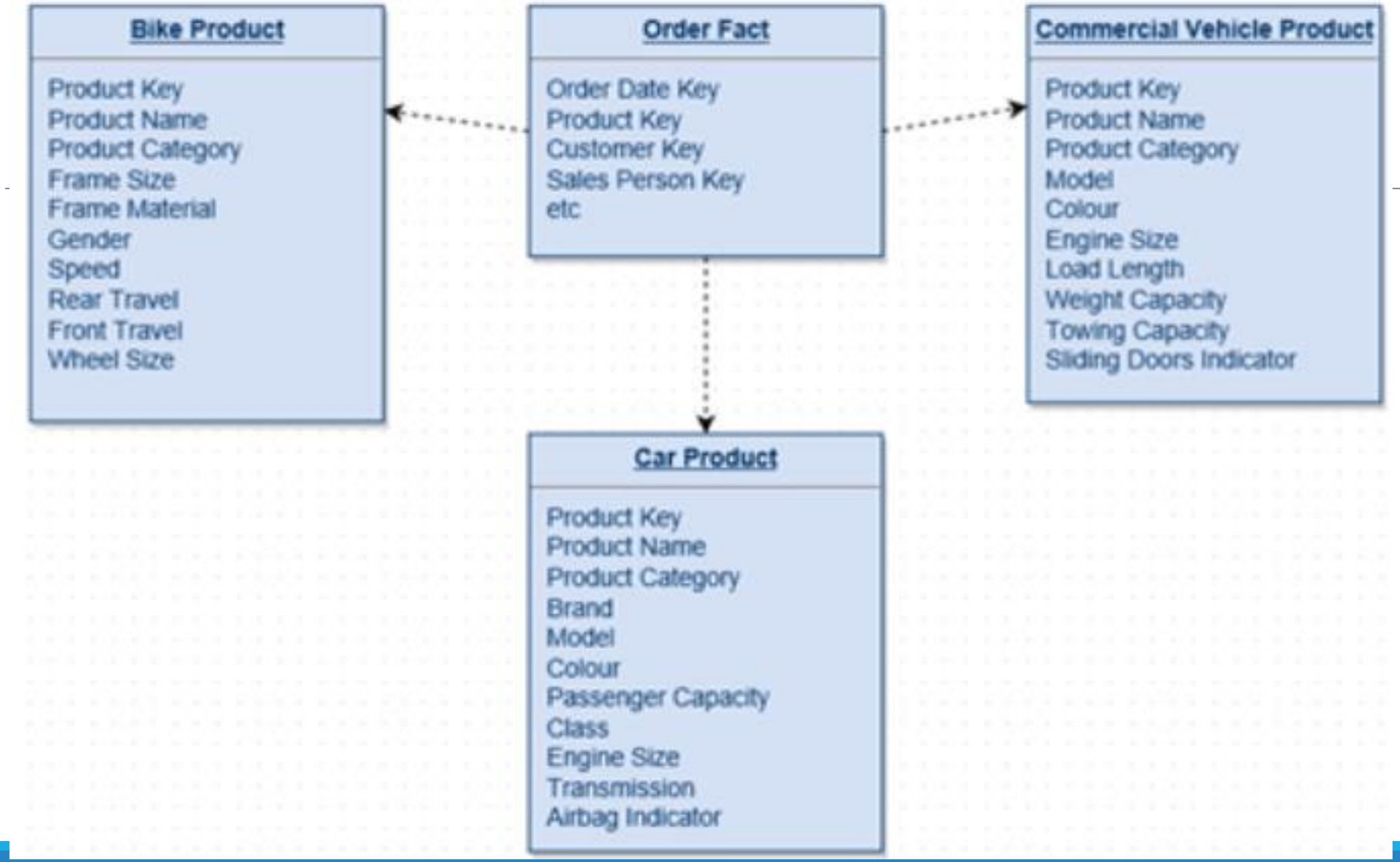
  Cons: Data redundancy, key could be duplicated (when join with fact), increase in data size, and ETL headache.

Heterogeneous Dimension

- This type works when we have a case that a company selling different product to the same base of customer. Every product has it different attributes.
  - One famous example of this type assume an insurance company has two types of product like health and car. In this case Car insurance has different attributes than the health insurance.
  - If we tried to model this two different products this type name Heterogeneous dimensions.

Heterogeneous Dimension

- There are different scenario to implement this type
  a. **Separate Dimensions**: Split each one in separate dimensions and facts. It will be less data and business will do this analysis from two separate facts.
  b. **Merge Attributes:** We will merge all the attributes in one single table and we will add the common attributes and null for un related attributes. Implementing this scenarios when we have less different of attributes. However, this implementation is not recommended because of the table size, performance, and maintenance.
  c. **Generic Design:** In this approach we will create a single fact table and single dimension with the common attributes. The problem of this design we will report or care about the common attributes only.


Bike Product
Product Key
Product Name
Product Category
Frame Size
Frame Material
Gender
Speed
Rear Travel
Front Travel
Wheel Size
Order Fact
Order Date Key
Product Key
Customer Key
Sales Person Key
etc
Commercial Vehicle Product
Product Key
Product Name
Product Category
Model
Colour
Engine Size
Load Length
Weight Capacity
Towing Capacity
Sliding Doors Indicator
Car Product
Product Key
Product Name
Product Category
Brand
Model
Colour
Passenger Capacity
Class
Engine Size
Transmission
Airbag Indicator

Slowly Changing Dimension

The dimension which changes over time. So, for a specific date we have — different value.

Type 0 (Fixed Dimension): We don’t change the current even the source changes.

Type 1 (No History): No history is maintained only the latest replace the current.

Type 2 (History): Series of history of records are maintained.

Type 3 (Hybrid): Only the last Change and the Current new change is stored

Type 4 : We split the data into two tables, first the current record and second is the historical (most common usage).

Slowly changing Dimensions Types

Type 0

No update if the previous dimension updated.

- Ronaldo current address in Madrid
  Ronaldo updated his to Turin

CustomerID	Name	City
123456789	Ronaldo	Madrid

CustomerID	Name	City
123456789	Ronaldo	Turin

Results in type 0 will not updated.

ID	CustomerID	Name	City
1	123456789	Ronaldo	Madrid

Slowly changing Dimensions Types

Type 1

- Ronaldo current address in Madrid Ronaldo updated his to Turin

CustomerID	Name	City
123456789	Ronaldo	Madrid

CustomerID	Name	City
123456789	Ronaldo	Turin

Results in type 1 will update the record without history maintenance.

ID	CustomerID	Name	City
1	123456789	Ronaldo	Turin

Type 2

- Ronaldo addresses history

CustomerID	Name	City	UpdatedDt
123456789	Ronaldo	Madrid	2018-12-12
123456789	Ronaldo	Turin	2019-06-12
123456789	Ronaldo	London	2019-08-12
123456789	Ronaldo	Porto	2019-12-12

Type 2 will keep all the history and keep the current with `terminationDt as null`

ID	CustomerID	Name	City	effectiveDt	terminationDt	isCurrent
1	123456789	Ronaldo	Madrid	2018-12-12	2019-06-12	false
2	123456789	Ronaldo	Turin	2019-06-12	2019-08-12	false
3	123456789	Ronaldo	London	2019-08-12	2019-12-12	false
4	123456789	Ronaldo	Porto	2019-12-12	null	true

Type 3

- Maintain the current and previous only with maintaining the history.

Result:

CustomerID	Name	City	UpdatedDt
123456789	Ronaldo	Madrid	2018-12-12
123456789	Ronaldo	Turin	2019-06-12
123456789	Ronaldo	London	2019-08-12
123456789	Ronaldo	Porto	2019-12-12

ID	CustomerID	Name	City	UpdatedDate	previousCity
1	123456789	Ronaldo	Porto	2019-12-12	London

Type 4

- Split current and historical into two tables.
- This change help to easy join with only current active records without the needs for the filter such as type 2.

ID	CustomerID	Name	City	effectiveDt	TerminationDt
1	123456789	Ronaldo	Madrid	2018-12-12	2019-06-12
2	123456789	Ronaldo	Turin	2019-06-12	2019-08-12
3	123456789	Ronaldo	London	2019-08-12	2019-12-12
4	123456789	Ronaldo	Porto	2019-12-12	null

ID	CustomerID	Name	City	UpdatedDate
1	123456789	Ronaldo	Porto	2019-12-12

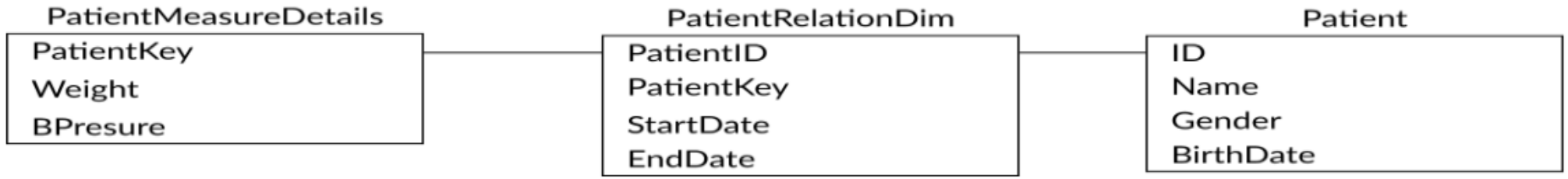
Fast Changing Dimension (Mini Dimension)

- When we have a dimension with one or more of its attributes changing very fast.
- It causes a performance issue if we tried to handle this case similar SCD Type 2 because of the rapidly changing in this dimension and the table will includes a lot of rows for this dimension
- We solve this case by separation the attributes into one or more dimensions. This technique also called mini-dimensions

Fast Changing Dimension Handling

- How to handle FCD?
  1. Identify the fast changing columns in dimension.
  2. Split the fast changing columns to a separate junk dimension.
  3. Map the junk dimension with the main dimension using *mini-dimension*
- In the following example, we have Weight & B_pressure columns are FCD

Patient_id	Name	Gender	BirthDate	Weight	B_Pressure	UpdateDt
123	Anna	F	1968-01-12	50	110.0	2019-01-01
123	Anna	F	1968-01-12	55	130.0	2019-01-07
123	Anna	F	1968-01-12	59	115.0	2019-01-14
123	Anna	F	1968-01-12	65	120.0	2019-01-21



- Patient Dimension (static columns)

Patient_id	Name	Gender	BirthDate
123	Anna	F	1968-01-12

- Junk Dimension

Patient_Key	Weight	B_Pressure
1	50	110.0
2	55	130.0
3	59	115.0
4	65	120.0

- Mini Dimension

Patient_id	Patient_Key	Start_Date	End_Date
123	1	2019-01-01	2019-01-07
123	2	2019-01-07	2019-01-14
123	3	2019-01-14	2019-01-21
123	4	2019-01-21	null

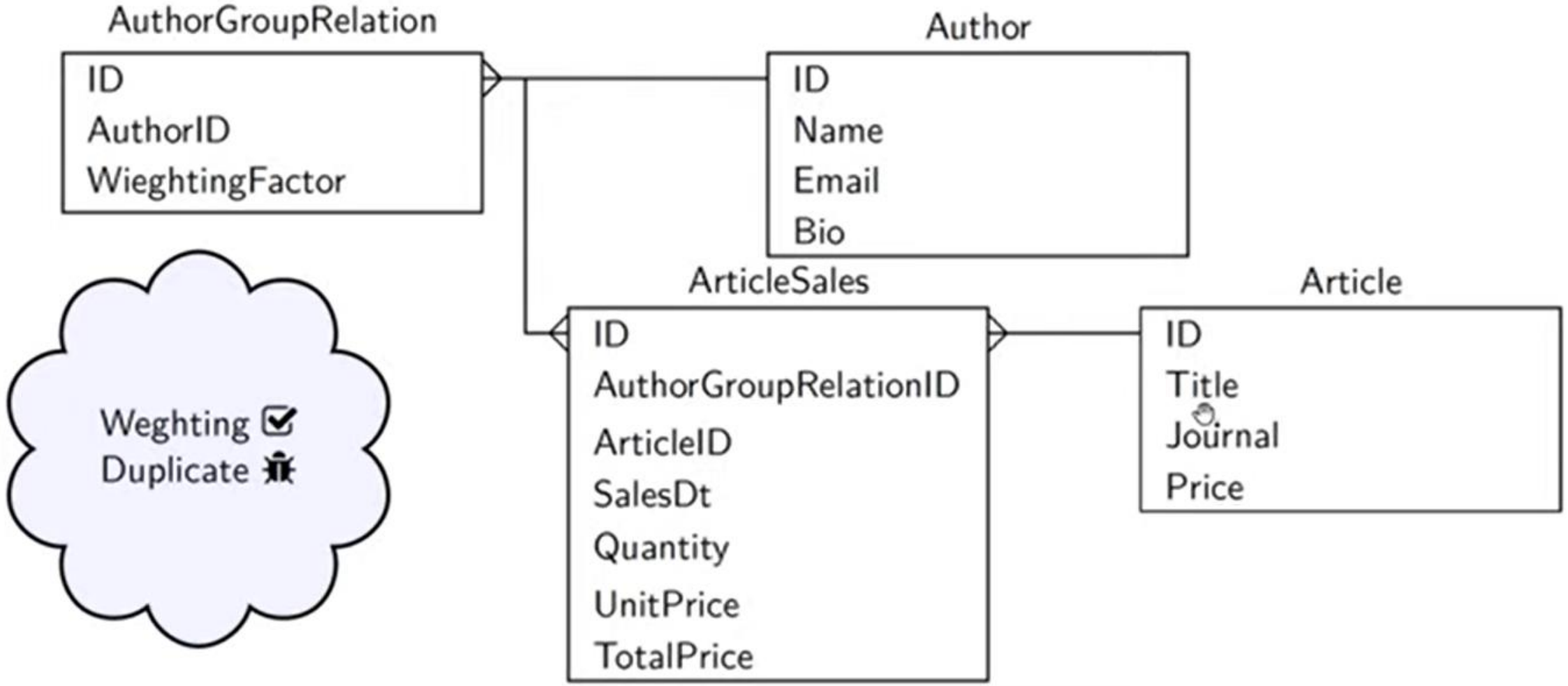
Multi Valued Dimension

- When the relationships between the dimension member and the fact are many to many which means the dimension members are lower granularity than the facts.
- Fact table should contains one-to-one relationship with the dimension. So, we introduce the *Bridge table* when we need to related multiple dimensions values with one record.
- Patients can have multiple diagnoses. Students can have multiple majors. customers can have multiple account. Authors can have multiple publications.

- Assume we need to report the sales of article and we have some articles has more than one author.
- Each author has weighting factor for each article.
- According to the report we need to check each author and associate with the articles they have authored. How can we model this case?
- Assume the first article has only one author *Moustafa*, and the second article has two authors *Ahmed & Amr*.

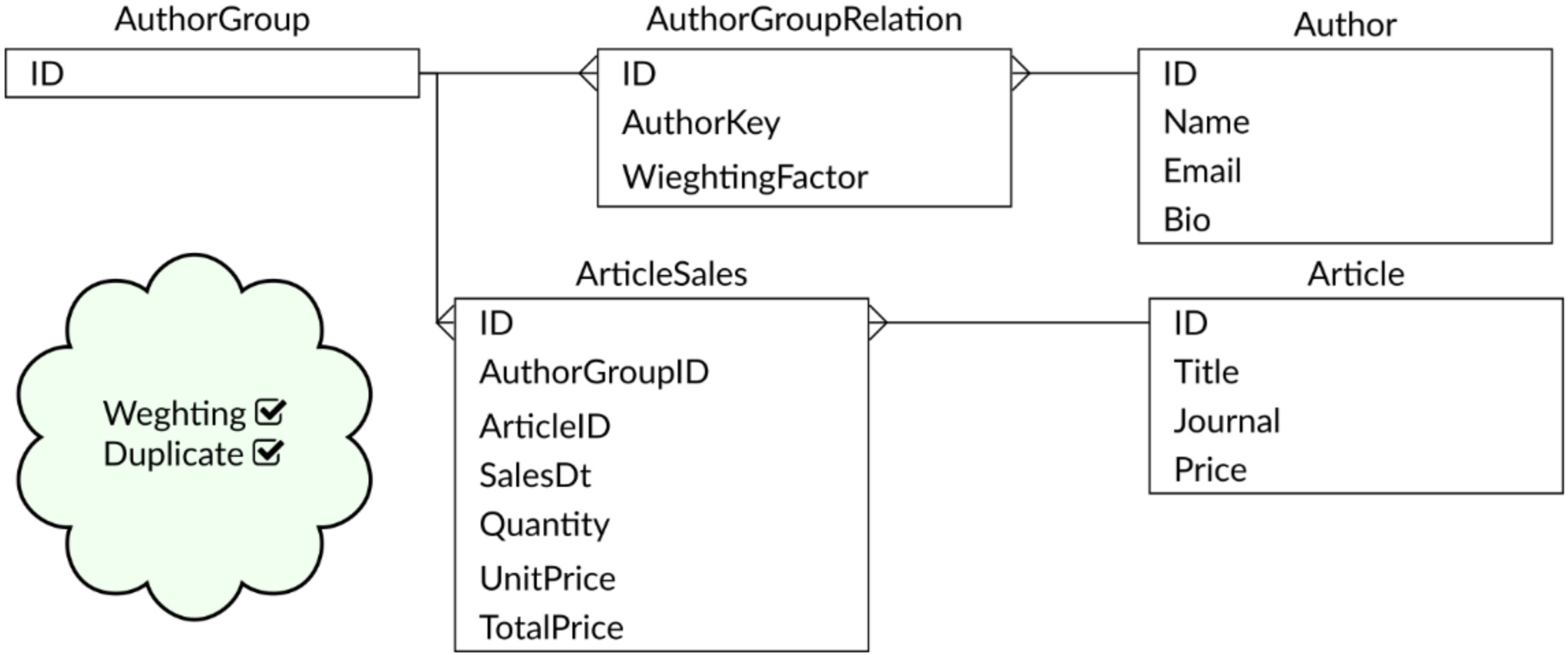
ID	Name	Email	Bio
123	Moustafa	abc@gability.com	S-Engineer
234	Ahmed	def@gability.com	L-Engineer
345	Amr	geh@gability.com	S-Manager

ID	Title	Journal	Price
11	50	IEEE	110.0
22	55	ACM	130.0



ID	AuthorGroupRelationID	ArticleID	SalesDt	Quantity	UnitPrice	TotalOrder
1	321	11	20200303	3	10	30
2	432	22	20200304	1	20	20
3	432	22	20200304	1	20	20

Table: Output of wrong implementation of ArticleSales



ID	AuthorGroupID	ArticleID	SalesDt	Quantity	UnitPrice	TotalOrder
1	321	11	20200303	3	10	30
2	432	22	20200304	1	20	20