



Advanced Databases

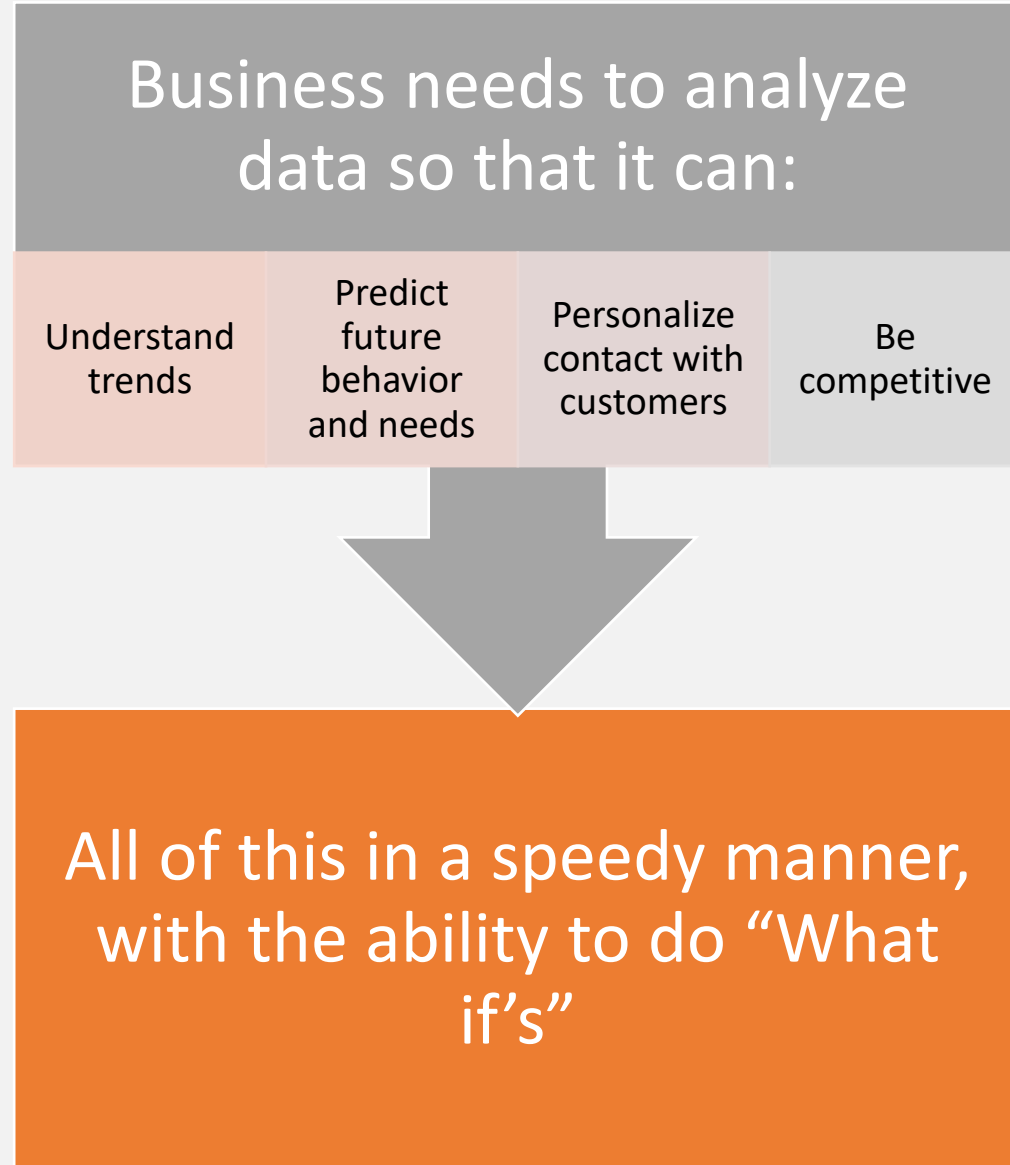
Data Warehouse

What is a Data Warehouse?

- A system where organisations keep their data in order to create reports and visualizations to support data-driven decision makings.
- Data from relevant relational databases are joined and transformed into a clearly-defined structure called star schema before being put into a single place in order to make the reading as efficient and seamless as possible.

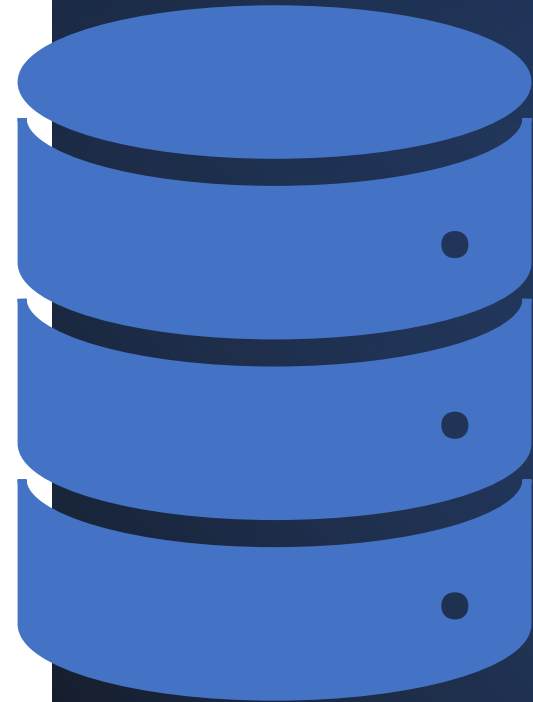


Why Data Warehouses ?

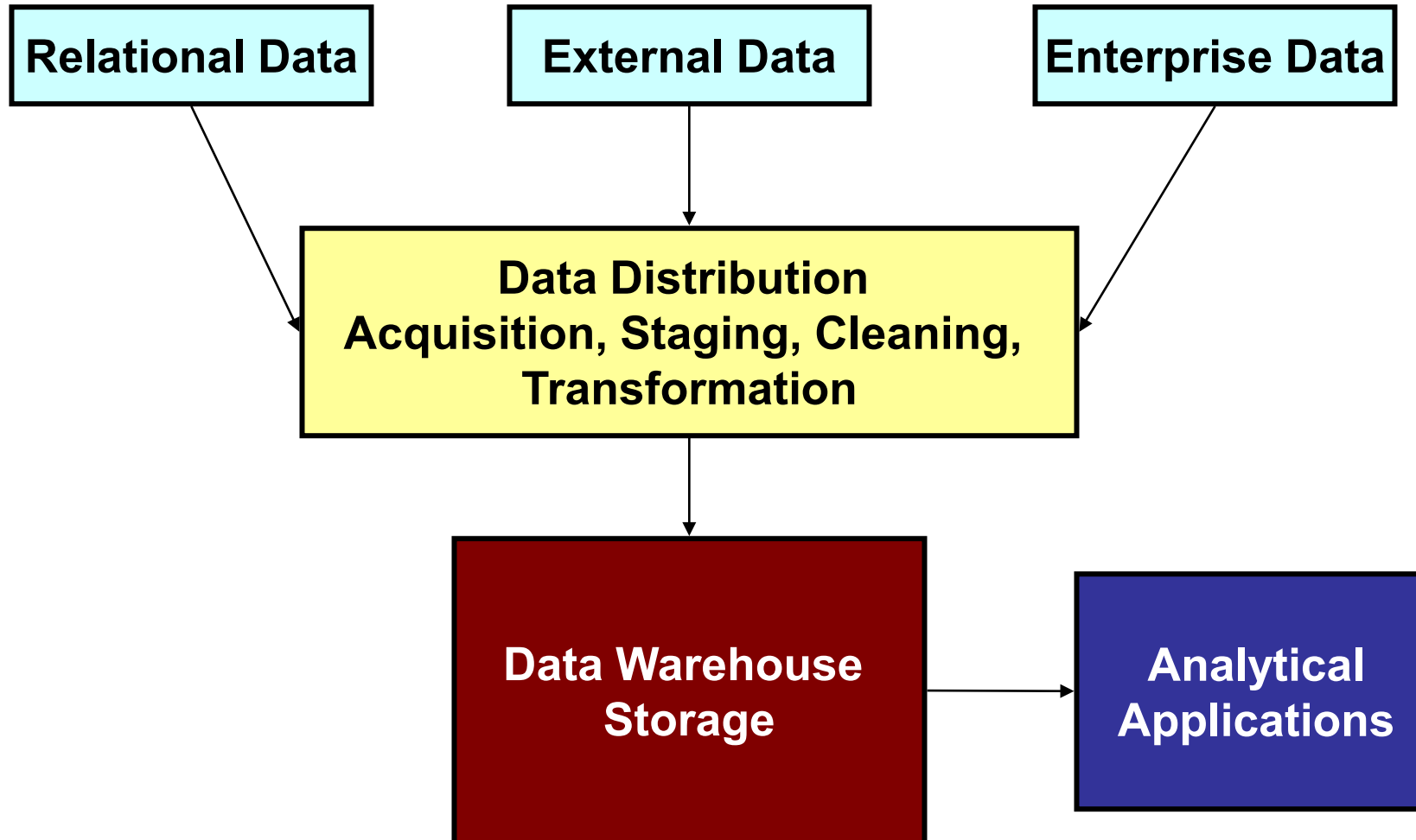


What is a Data Warehouse?

- A database (and it can be built using a relational DB like Oracle)
- It is not a live / day-by-day database
- Day-by day transactions go to another database(s), usually fully relational databases fully normalized

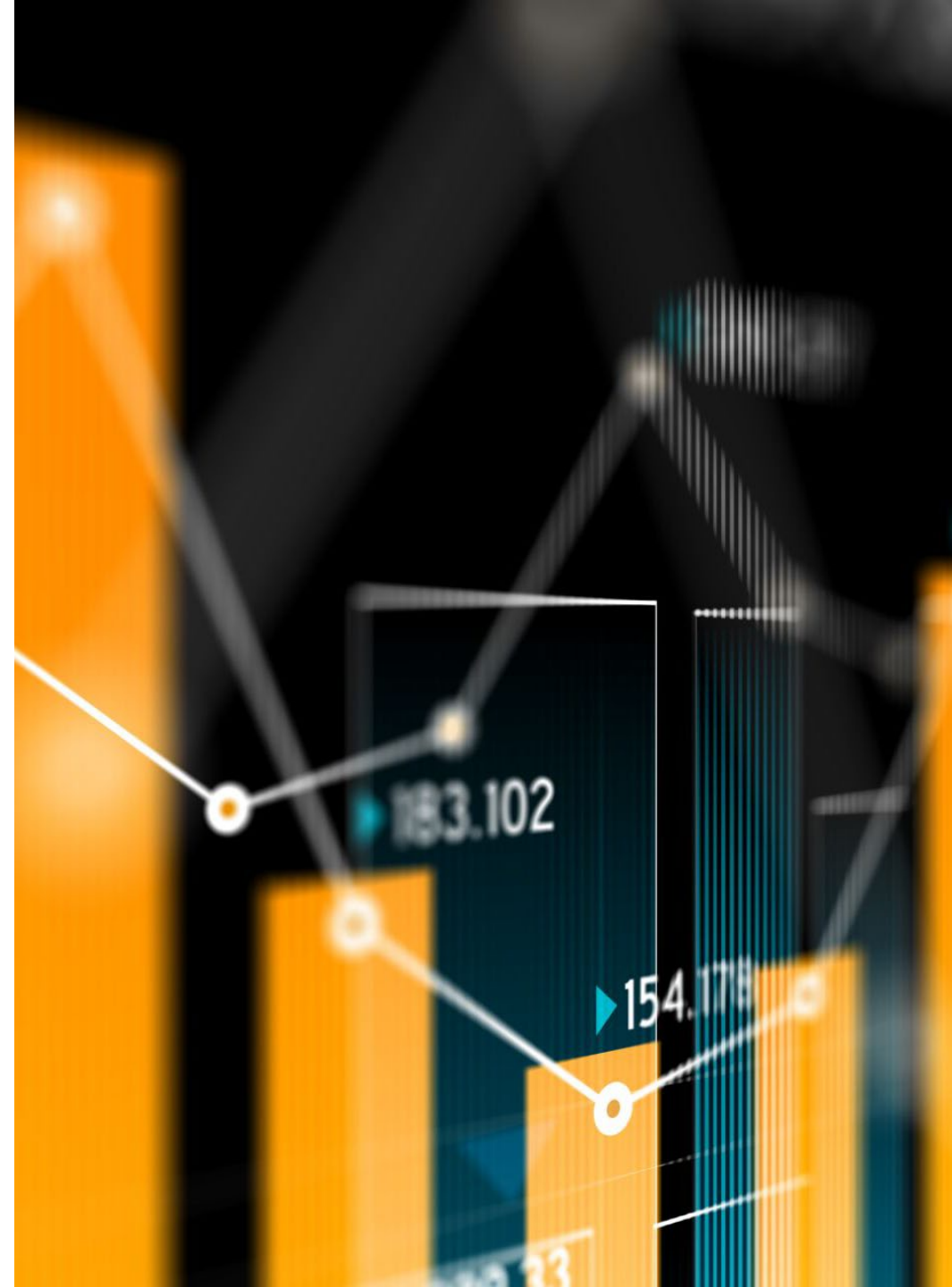


Dimensional Data Warehouse Architecture



What is a Data Warehouse?

- A DW:
 - is updated at specific points in time
 - is mainly read-only (analytics)
 - is optimized for (read) performances
 - is a collection (integration) of different sources
- In our diagram on the previous slide
 - The “yellow” box (= the staging area) is permanent and it is where data are clean and integrated



Evolution of IT Systems (60's to 80's)

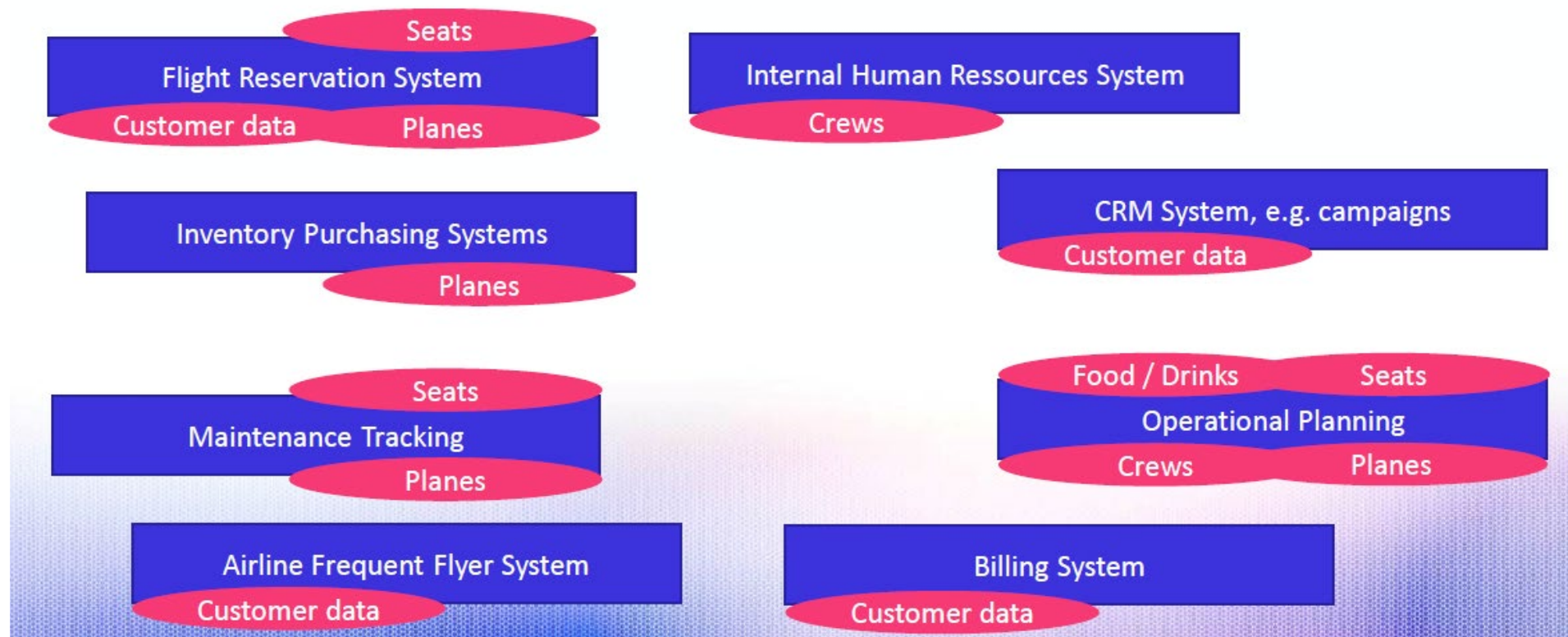
Many systems throughout the enterprises for dedicated purposes

- To support daily transactions / day to day business
- Target: replace manual and time-consuming activities

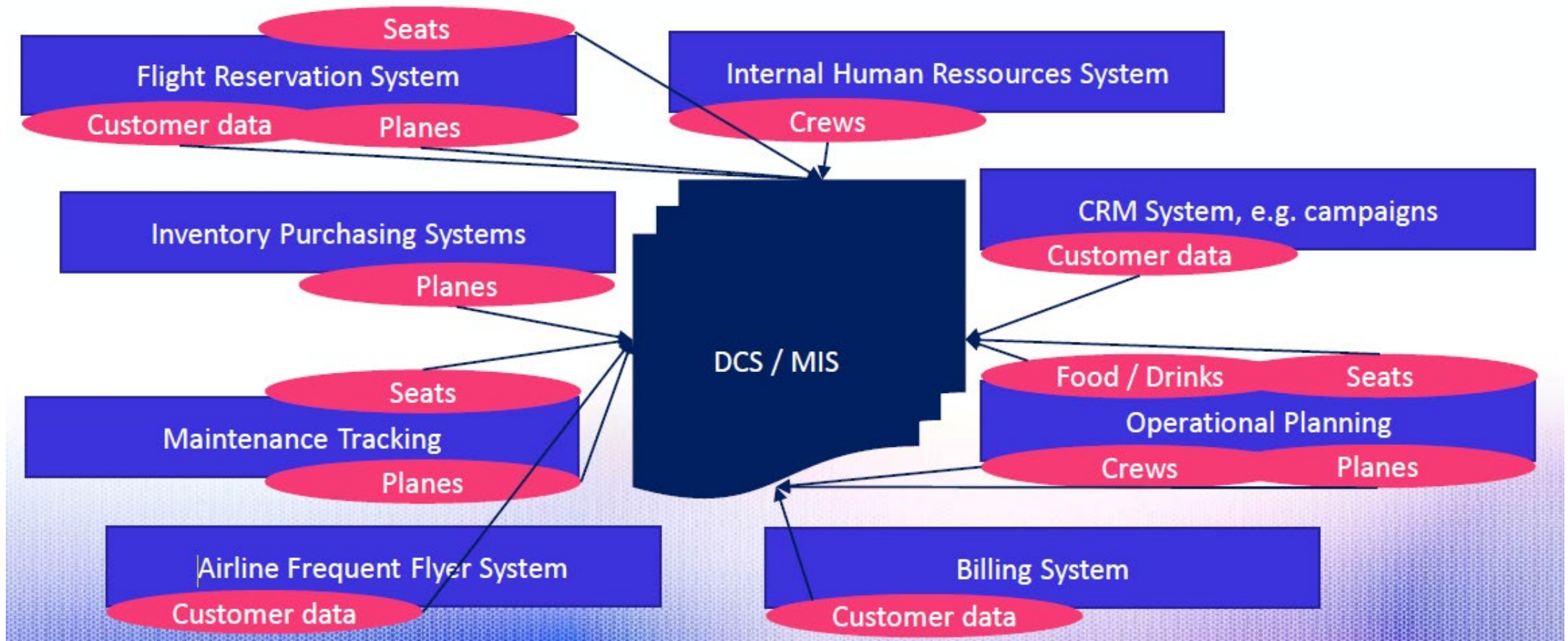
Data embedded in process specific application

- Process orientation + dedicated purpose
- Customer data, order data, etc. spread over many systems in many variations and with contradictions

Example Applications for an Airline



Need emerged for Management Information Systems/Decision Support Systems



Evolution of IT Systems (60's to 80's)

- Pretty much unplanned
 - Management needs reports / combined data from different systems to make decisions for company
- Reports were manually written by IT people
 - Extract, combine, accumulate data
 - Could take several days to write report and to get the data
 - Error prone and labour-intensive
- Relevant information could be forgotten or combined in a wrong way
- Not a huge success but need was identified for improved MIS/DSS

IT Systems Today



Data still spread across many applications, but additional requirements



Data as Asset, getting more and more important also in all industries

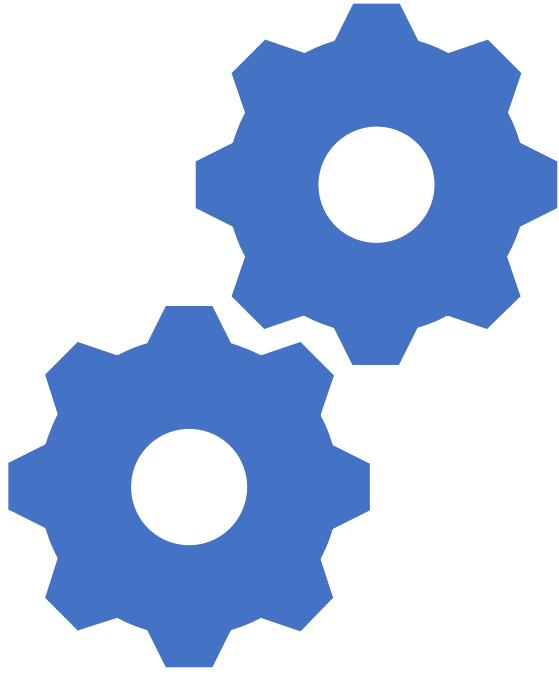


Hype technologies

New databases technologies like NoSQL and Big Data



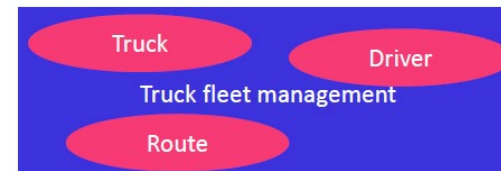
DWH still booming with increased interest generated from Big Data, Digitization, Internet Of Things IOT, Industry 4.0....



Exercise OLTP


- Outline at least 5 operational systems for a vehicle manufacturer which also rents vehicles, maintains and repairs vehicles and offers loans to customers
 - which data is stored by these systems
 - characterize which operations are performed by them
 - which questions can be answered by these systems
 - and which questions can not be answered - major problems for decision support

Some OLTP



Problems for DSS Support

- Distributed data
 - Data resides on different systems / storages, in different applications implemented using different technologies
- Challenges to overcome
 - Data has to be accumulated on one system for further analysis
 - But data is heterogeneous, e.g. each system has their own customer number or order number etc
 - How do you combine the data?
 - How to you keep this data up to date?



```
mirror_mod = modifier_ob.  
#set mirror object to mirror  
mirror_mod.mirror_object =  
operation == "MIRROR_X":  
mirror_mod.use_x = True  
mirror_mod.use_y = False  
mirror_mod.use_z = False  
operation == "MIRROR_Y":  
mirror_mod.use_x = False  
mirror_mod.use_y = True  
mirror_mod.use_z = False  
operation == "MIRROR_Z":  
mirror_mod.use_x = False  
mirror_mod.use_y = False  
mirror_mod.use_z = True  
  
#selection at the end -add  
mirror_ob.select= 1  
modifier_ob.select=1  
context.scene.objects.active  
("Selected" + str(modifier  
mirror_ob.select = 0  
= bpy.context.selected_object  
data.objects[one.name].select  
  
print("please select exactly  
  
--- OPERATOR CLASSES ---  
  
types.Operator):  
X mirror to the selected  
object.mirror_mirror_x"  
mirror X"  
  
context):  
context.active_object is not
```

```
mirror_mod = modifier_ob.  
set mirror object to mirror.  
mirror_mod.mirror_object =  
operation == "MIRROR_X":  
mirror_mod.use_x = True  
mirror_mod.use_y = False  
mirror_mod.use_z = False  
operation == "MIRROR_Y":  
mirror_mod.use_x = False  
mirror_mod.use_y = True  
mirror_mod.use_z = False  
operation == "MIRROR_Z":  
mirror_mod.use_x = False  
mirror_mod.use_y = False  
mirror_mod.use_z = True  
  
selection at the end -add  
mirror_ob.select= 1  
modifier_ob.select=1  
context.scene.objects.active  
("Selected" + str(modifier_ob.  
mirror_ob.select = 0  
= bpy.context.selected_object  
data.objects[one.name].select  
  
print("please select exactly  
  
-- OPERATOR CLASSES ----  
  
types.Operator):  
on X mirror to the selected  
object.mirror_mirror_x"  
mirror X"  
  
context):  
context.active_object is not
```

Problems for DSS Support

- Different data structures
 - Different data types E.g.: zip code as integer or character string
 - Different encodings E.g.: kilometre or miles
 - Different data modelling E.g.: last name / first name in different fields vs last name / first name in one single field (not good modelling)
- To overcome
 - Need a dedicated system required that harmonizes / standardizes the data


```
mirror_mod = modifier_ob.  
set mirror object to mirror.  
mirror_mod.mirror_object =  
operation == "MIRROR_X":  
mirror_mod.use_x = True  
mirror_mod.use_y = False  
mirror_mod.use_z = False  
operation == "MIRROR_Y":  
mirror_mod.use_x = False  
mirror_mod.use_y = True  
mirror_mod.use_z = False  
operation == "MIRROR_Z":  
mirror_mod.use_x = False  
mirror_mod.use_y = False  
mirror_mod.use_z = True  
  
#selection at the end -add  
mirror_ob.select= 1  
modifier_ob.select=1  
context.scene.objects.active  
("Selected" + str(modifier_ob.  
mirror_ob.select = 0  
= bpy.context.selected_object  
data.objects[one.name].select  
  
print("please select exactly  
  
-- OPERATOR CLASSES ----  
  
types.Operator):  
on X mirror to the selected  
object.mirror_mirror_x"  
mirror X"  
  
context):  
context.active_object is not
```

Problems for DSS Support

- Different data structures
 - Different data types E.g.: zip code as integer or character string
 - Different encodings E.g.: kilometere or miles
 - Different data modelling E.g.: last name / first name in different fields vs last name / first name (badly modelled) in one single field
- To overcome
 - Need a dedicated system required that harmonizes / standardizes the data

Problems for DSS Support

- Historical data
 - This is Usually deleted periodically
 - But it is needed for decision making
- Need a system which is capable of storing large volumes of data

```
mirror_mod = modifier_ob.  
#set mirror object to mirror_  
mirror_mod.mirror_object =  
operation == "MIRROR_X":  
mirror_mod.use_x = True  
mirror_mod.use_y = False  
mirror_mod.use_z = False  
operation == "MIRROR_Y":  
mirror_mod.use_x = False  
mirror_mod.use_y = True  
mirror_mod.use_z = False  
operation == "MIRROR_Z":  
mirror_mod.use_x = False  
mirror_mod.use_y = False  
mirror_mod.use_z = True  
  
#selection at the end -add  
mirror_ob.select= 1  
modifier_ob.select=1  
context.scene.objects.active  
("Selected" + str(modifier_ob.  
mirror_ob.select = 0  
= bpy.context.selected_object  
data.objects[one.name].select  
  
print("please select exactly  
  
-- OPERATOR CLASSES ----  
  
types.Operator):  
on X mirror to the selected  
object.mirror_mirror_x"  
mirror X"  
  
context):  
context.active_object is not
```

Problems for DSS Support

- System Workload
- DSS queries should not add to operational system workload
- Need a dedicated system that handles complex (arithmetic) queries on huge amounts of data and that can handle level of workload required.

```
mirror_mod = modifier_ob.  
#set mirror object to mirror_  
mirror_mod.mirror_object =  
operation == "MIRROR_X":  
mirror_mod.use_x = True  
mirror_mod.use_y = False  
mirror_mod.use_z = False  
operation == "MIRROR_Y":  
mirror_mod.use_x = False  
mirror_mod.use_y = True  
mirror_mod.use_z = False  
operation == "MIRROR_Z":  
mirror_mod.use_x = False  
mirror_mod.use_y = False  
mirror_mod.use_z = True  
  
#selection at the end -add  
mirror_ob.select= 1  
modifier_ob.select=1  
context.scene.objects.active  
("Selected" + str(modifier_ob.  
mirror_ob.select = 0  
= bpy.context.selected_object  
data.objects[one.name].select  
  
print("please select exactly  
  
-- OPERATOR CLASSES ----  
  
types.Operator):  
on X mirror to the selected  
object.mirror_mirror_x"  
mirror X"  
  
context):  
context.active_object is not
```


Problems for DSS Support

- Inadequate technology
- Technology used for OLTP isn't usually capable of supporting OLAP
- Need to use appropriate tools and technology

```
mirror_mod = modifier_ob.  
#set mirror object to mirror  
mirror_mod.mirror_object =  
operation == "MIRROR_X":  
mirror_mod.use_x = True  
mirror_mod.use_y = False  
mirror_mod.use_z = False  
operation == "MIRROR_Y":  
mirror_mod.use_x = False  
mirror_mod.use_y = True  
mirror_mod.use_z = False  
operation == "MIRROR_Z":  
mirror_mod.use_x = False  
mirror_mod.use_y = False  
mirror_mod.use_z = True  
  
#selection at the end -add  
mirror_ob.select= 1  
modifier_ob.select=1  
context.scene.objects.active  
("Selected" + str(modifier_ob.  
mirror_ob.select = 0  
= bpy.context.selected_object  
data.objects[one.name].select  
  
print("please select exactly  
  
-- OPERATOR CLASSES ----  
  
types.Operator):  
on X mirror to the selected  
object.mirror_mirror_x"  
mirror X"  
  
context):  
context.active_object is not
```

```

mirror_mod = modifier_ob.
set mirror object to mirror
mirror_mod.mirror_object

operation == "MIRROR_X":
mirror_mod.use_x = True
mirror_mod.use_y = False
mirror_mod.use_z = False
operation == "MIRROR_Y":
mirror_mod.use_x = False
mirror_mod.use_y = True
mirror_mod.use_z = False
operation == "MIRROR_Z":
mirror_mod.use_x = False
mirror_mod.use_y = False
mirror_mod.use_z = True

#selection at the end -add
mirror_ob.select= 1
modifier_ob.select=1
context.scene.objects.active
("Selected" + str(modifier_ob
mirror_ob.select = 0
= bpy.context.selected_object
data.objects[one.name].select

print("please select exactly

-- OPERATOR CLASSES -----

types.Operator):
on X mirror to the selected
object.mirror_mirror_x"
mirror X"

context):
context.active_object is not

```

DSS Support

- Using operative systems for OLAP poses problems
- Need for a new, separated system that can answer the right questions, quickly
 - And supports ad hoc queries
- Without interfering with transactional systems
- with daily transaction business

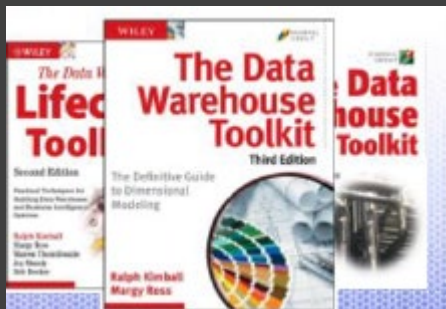
Data Warehouse

- Contains data from different systems
- Imports data from different systems on a regular basis
 - detailed data and summarized data
 - provide historic data
 - generate metadata
- OLTP applications remain, DWH is a completely new system
- Overcomes difficulties when using existing transaction systems for those tasks

Data Warehouse Definition

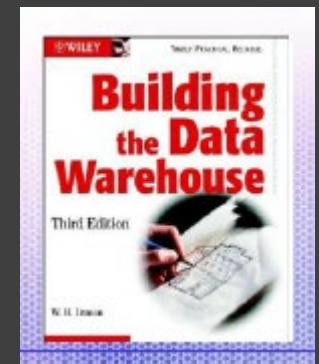
Ralph Kimball

A data warehouse is a copy of transaction data specifically structured for querying and reporting



William Marshall, Bill Inmon

A data warehouse is a subject oriented, integrated, time variant, nonvolatile collection of data in support of management's decision making process

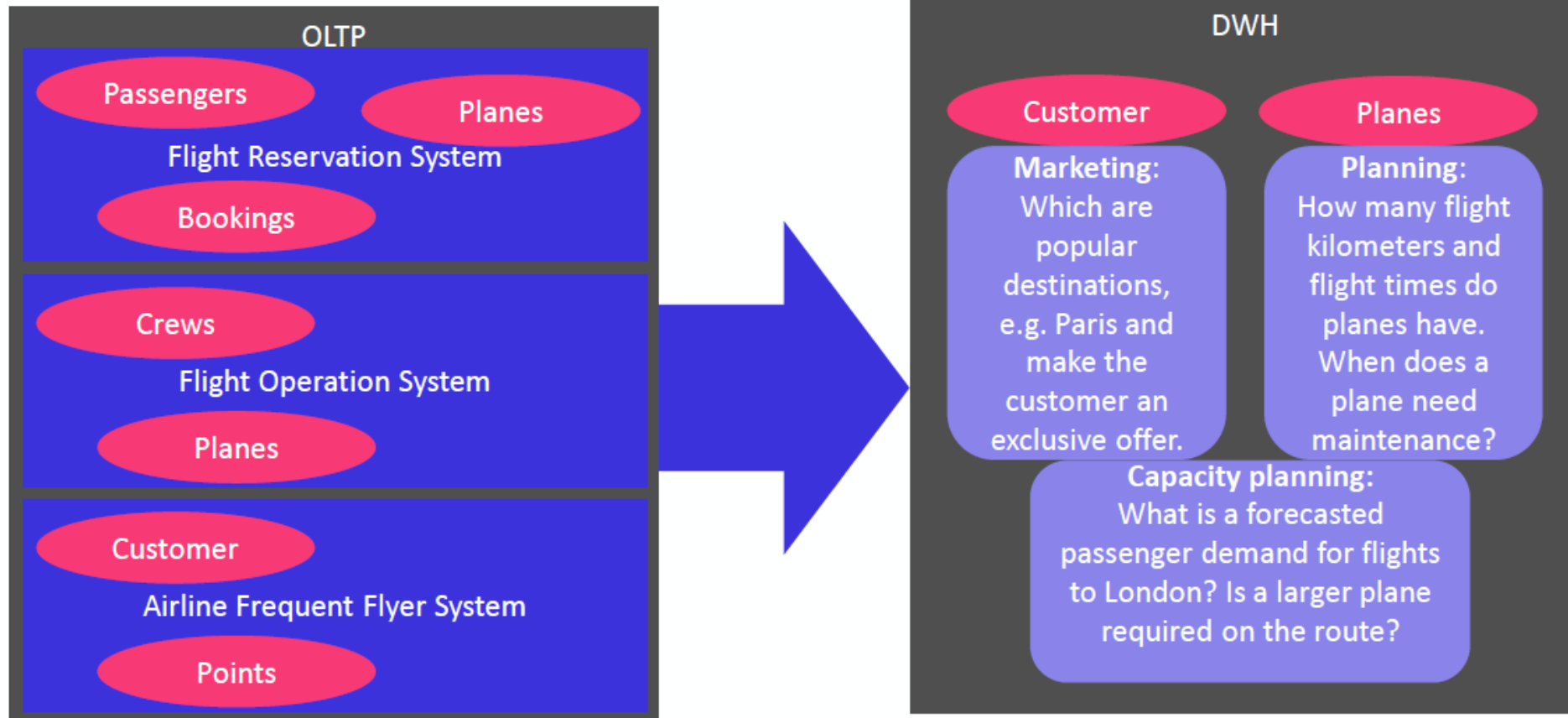




Data Warehouse - Definition

- Subject-Oriented:
 - Data is organised around the major subjects of the enterprise (e.g. customers, sales, products) rather than application areas (e.g. invoicing, stock control etc.)

Subject-Oriented

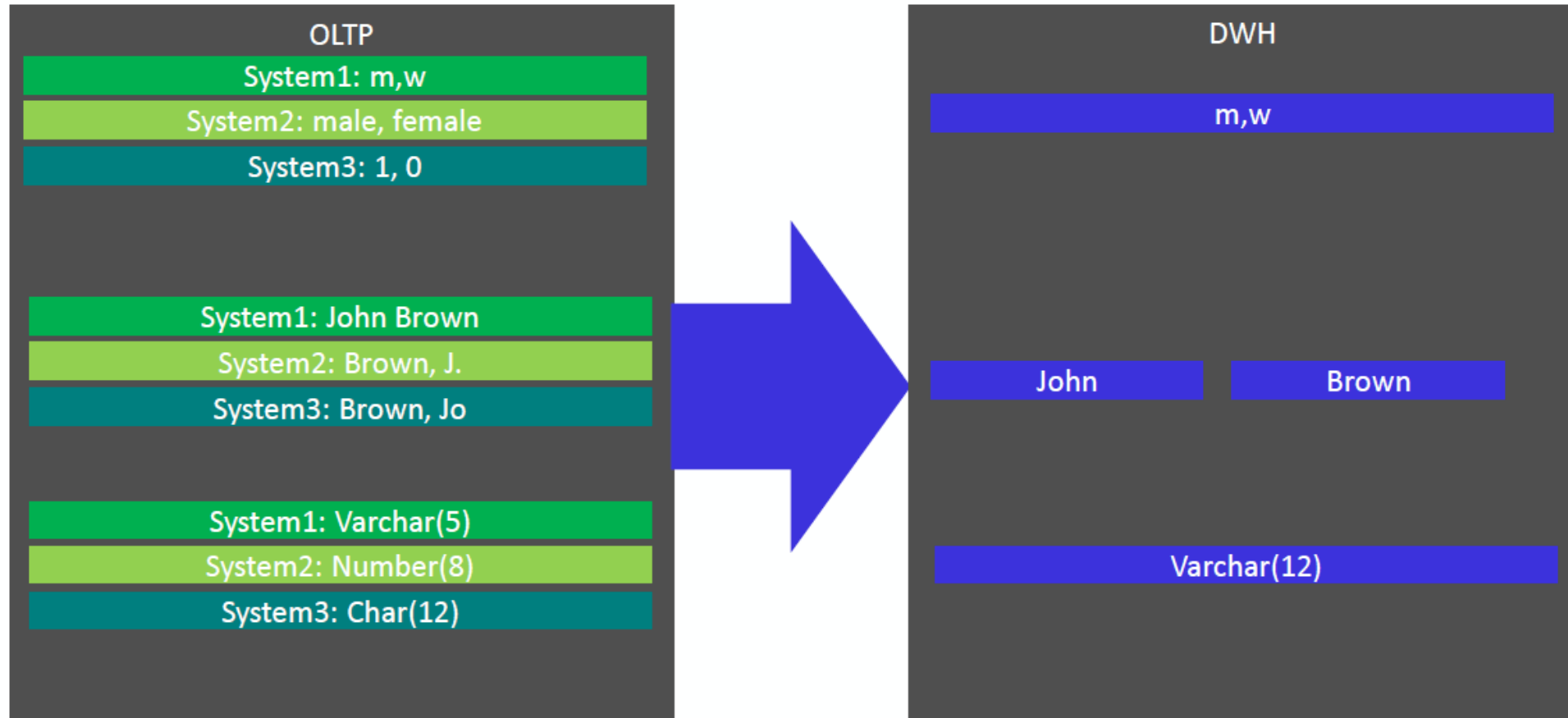




Data Warehouse - Definition

- Integrated:
 - Data from different sources are combined.
 - These sources may be inconsistent and formatted differently.
 - The DW establishes a consistent combined data source.
 - consistent naming conventions
 - consistent measurement of variables
 - consistent encoding structures
 - consistent physical attributes of data

Integrated



Data Warehouse - Definition

- Time Variant:
 - Data is only accurate at a particular point of time or over some time interval.
 - Time variances is shown in the extended time that data is held.
 - The implicit or explicit association of time with all the data and the fact that the data represents a series of snapshots.

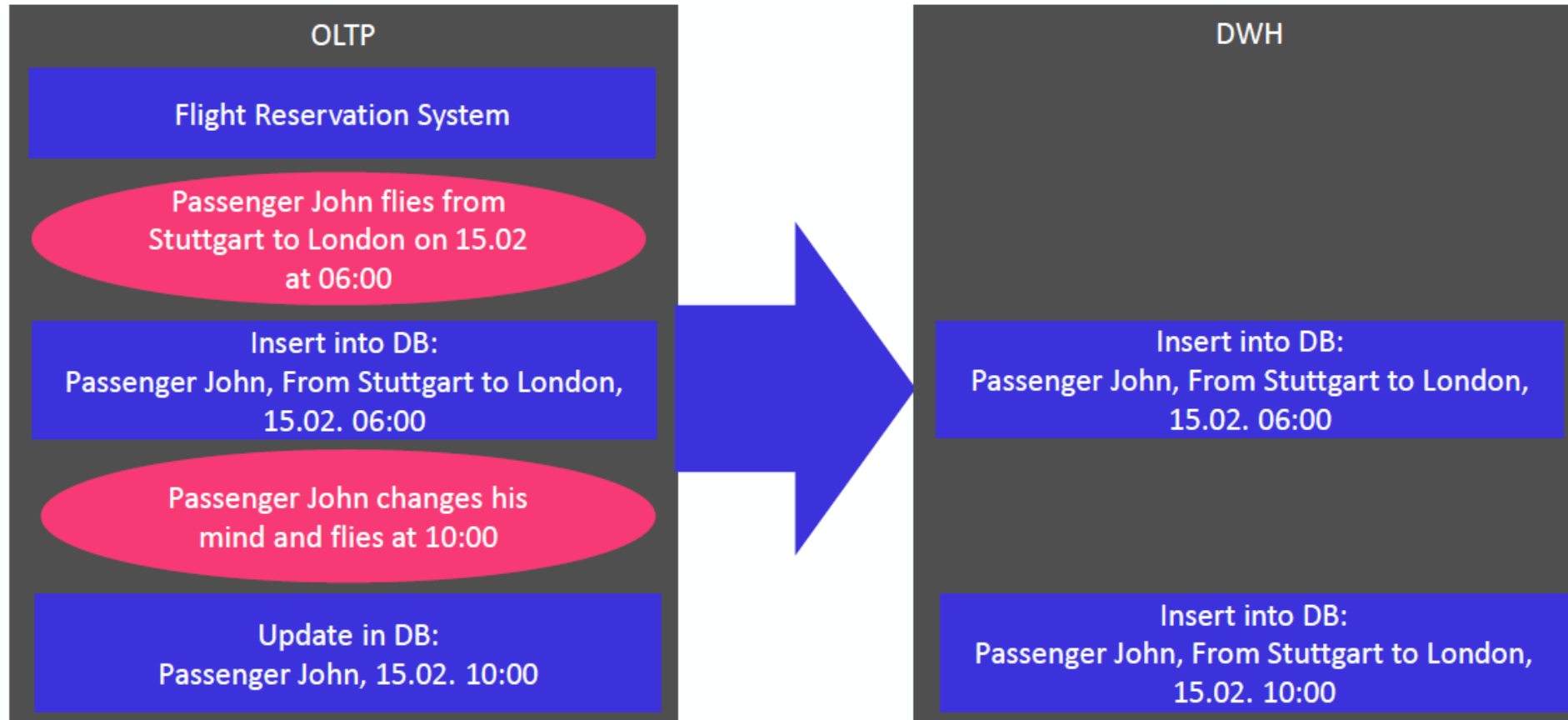
Time-Variant

DWH	
Insert into DB: Passenger John, From Stuttgart to London, 15.02. 06:00	DB insert timestamp: 02.02. 15:03:21
Insert into DB: Passenger Jim, From Hamburg to Munich, 18.02. 15:00	DB insert timestamp: 02.02. 15:04:29
Insert into DB: Passenger John, From Stuttgart to London, 15.02. 10:00	DB insert timestamp: 05.02. 12:15:03
Insert into DB: Passenger Mike, From Hamburg to Munich, 15.02. 10:00	DB insert timestamp: 05.02. 12:15:11
Insert into DB: Passenger John, From Stuttgart to London, 15.02. 10:00, Cancel Flag	DB insert timestamp: 08.02. 09:52:33

Data Warehouse - Definition

- Non-Volatile:
 - Data is not updated in real-time but refreshed from operational data at regular intervals.
 - New data is already added as a supplemental to the database rather than as a replacement.
 - The database is constantly absorbs new data, incrementally integrating it with the previous data.

Non-Volatile



Data Warehouse - Requirements

- Accessibility:
 - Understandable- legible, meaningfully labelled
 - Intuitive and obvious to the business user – not just developers
 - Requires well-designed tools that are simple and easy to use in accessing data
 - Tractable – minimal wait time on data operations
- Consistency
 - Credible data – data must be clean and quality assured
 - Cross Business Process Compatible – a customer is always a customer, otherwise it should be labelled differently
 - Common definitions should be available for end users
 - Consistent information is high quality information that is accounted for and complete

Data Warehouse - Requirements

- Adaptive and Resilient
 - Tolerant to business changes (which are inevitable)
 - Warehouse changes should be graceful and should not invalidate existing data or applications
 - New case or business cases should not disrupt existing applications
 - If changes to descriptive data cannot be avoided, appropriate measures must be in place to account for these changes
- Secure
 - A warehouse contains business critical, sensitive, confidential and valuable information that may be harmful in the wrong hands
 - Requirements include:
 - Access control
 - Data distribution
 - Encryption
 - Redundancy
 - Etc

Data Warehouse - Requirements

- Supports improved decision making
 - Need the right data, visualization and analytical tools
 - There is only one true output of a DW – the decision made after viewing the evidence from the DW
 - Evidence should support decisions that deliver business impact and value
 - Decision support systems



Data Warehouse - Requirements

- Acceptable
 - User acceptance = success
 - Senior management must also buy in and support the increased use of this approach and technology
 - Requires that users trust the data
 - Tools must be intuitive





Exercise

- You identified OLTP systems for a vehicle manufacturer in an earlier exercise.
- Now start designing a Data Warehouse:
 - Describe what data can be stored in it.
 - Define at least 5 subject areas
 - Which questions can/should be answered with this information

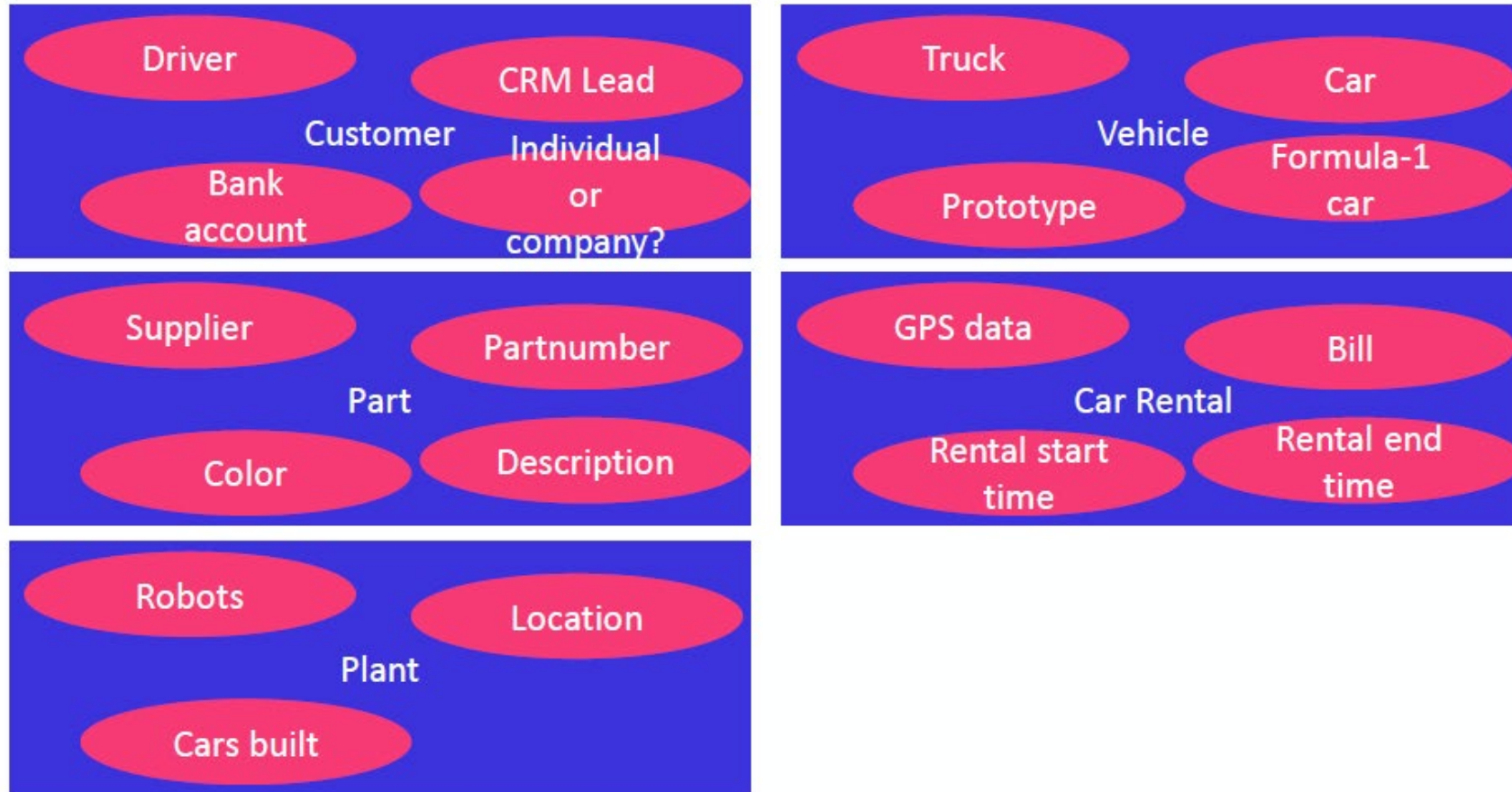
Which customers own a car and use car rental regularly?

Which parts have the most defects?

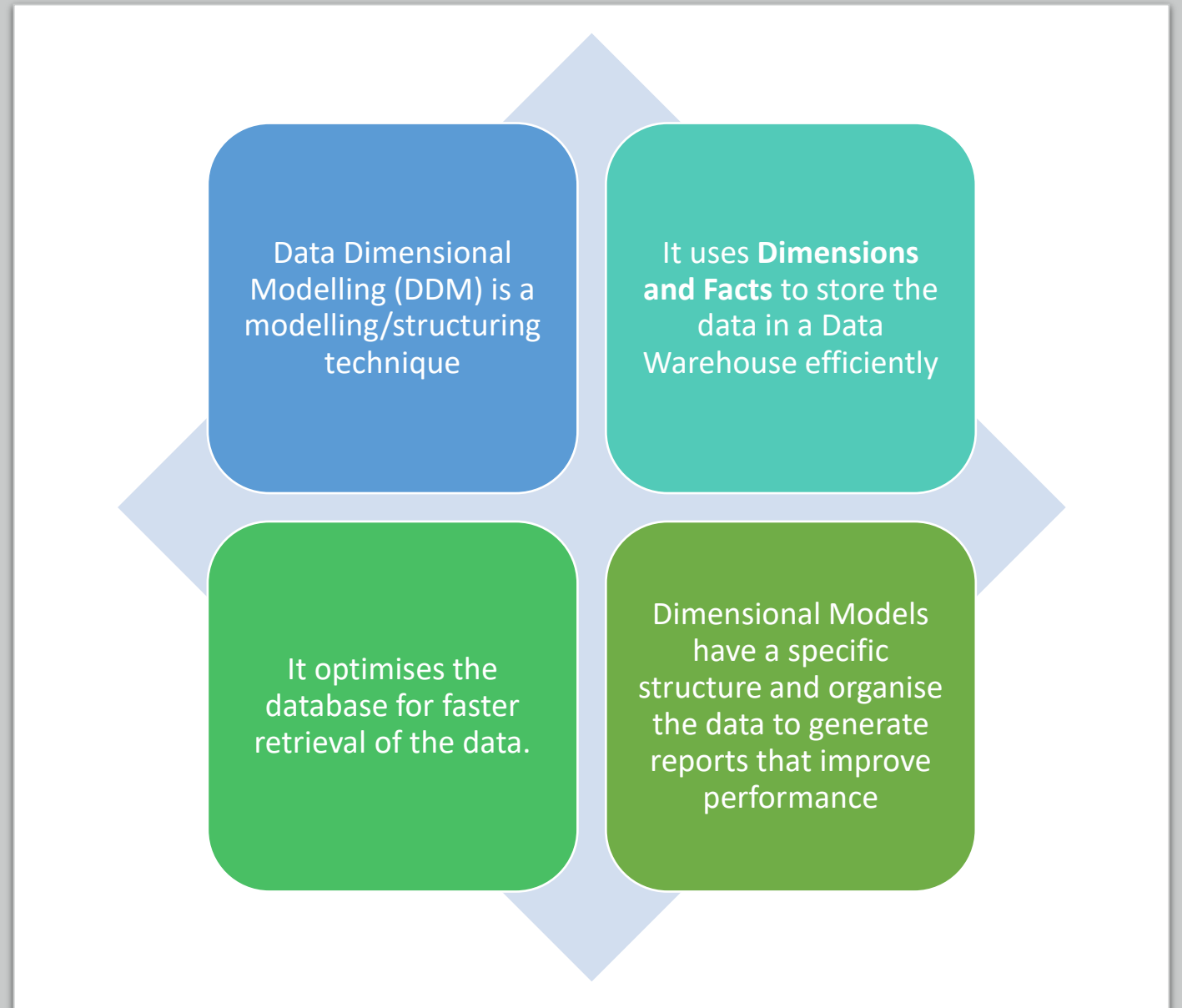
Can diagnostic data be used to predict potential defects and warn customers?

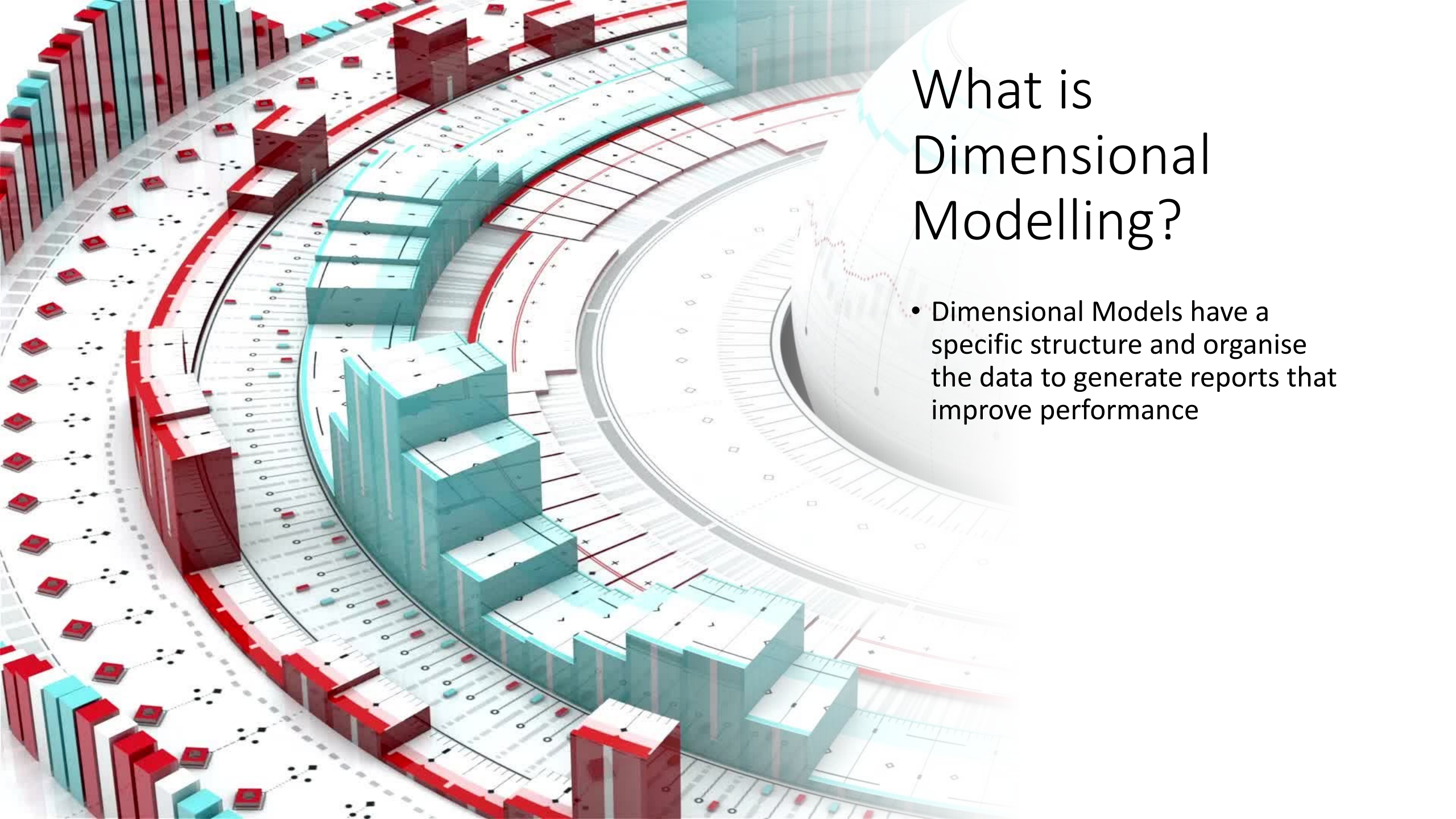
Which areas and times are popular for car rentals?

Does it make sense to relocate cars to these areas? (e.g. cinema in the evening/night)



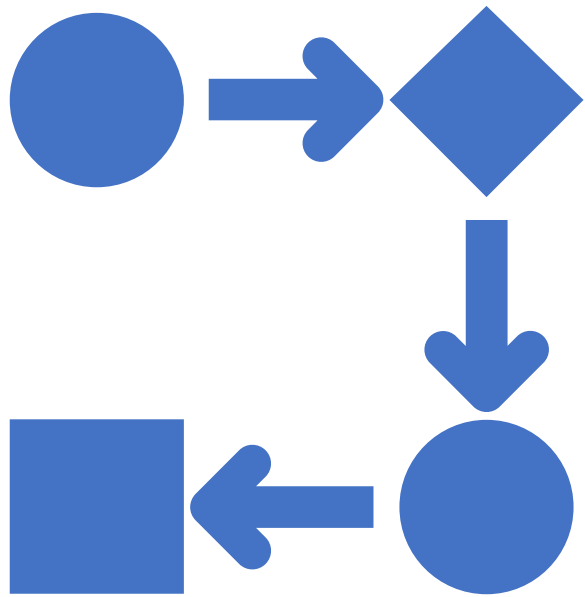
What is Dimensional Modelling?





What is Dimensional Modelling?

- Dimensional Models have a specific structure and organise the data to generate reports that improve performance



Dimension Modelling

- Part of the Business Dimensional Lifecycle methodology developed by Ralph Kimball
- Includes a set of methods, techniques and concepts for use in data warehouse design.
- Focuses on identifying the key business processes within a business and modelling and implementing these first before adding additional business processes, a bottom-up approach.



DM Objectives

- Make information easily accessible
- Present information consistently
- Adaptable and receptive to change
- Present information in a timely way
- Protect information assets
- Serve as an authoritative and trustworthy foundation for improved decision making (single source of truth in data engineering language)
- Keep key stakeholders (VIPs) happy

What is Dimensional Modeling?

- Every dimensional model is composed of one table with a multipart key
 - called the **fact** table,
- and a set of smaller tables
 - called **dimension** tables.



Dimension and Fact



A fact:

A quantitative piece of information - such as a sale or a download.

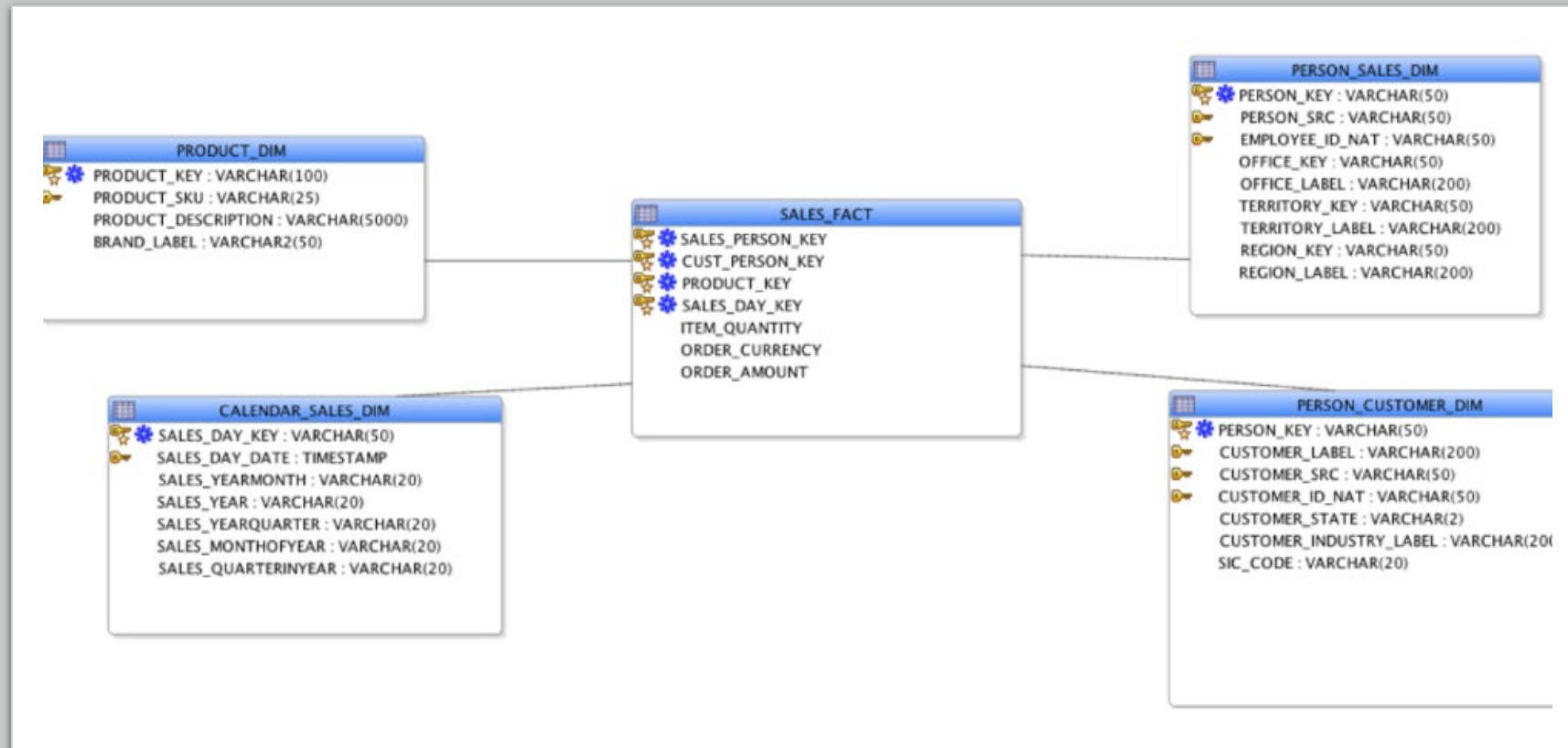
Facts are stored in fact tables and have a foreign key relationship with a number of dimension tables.



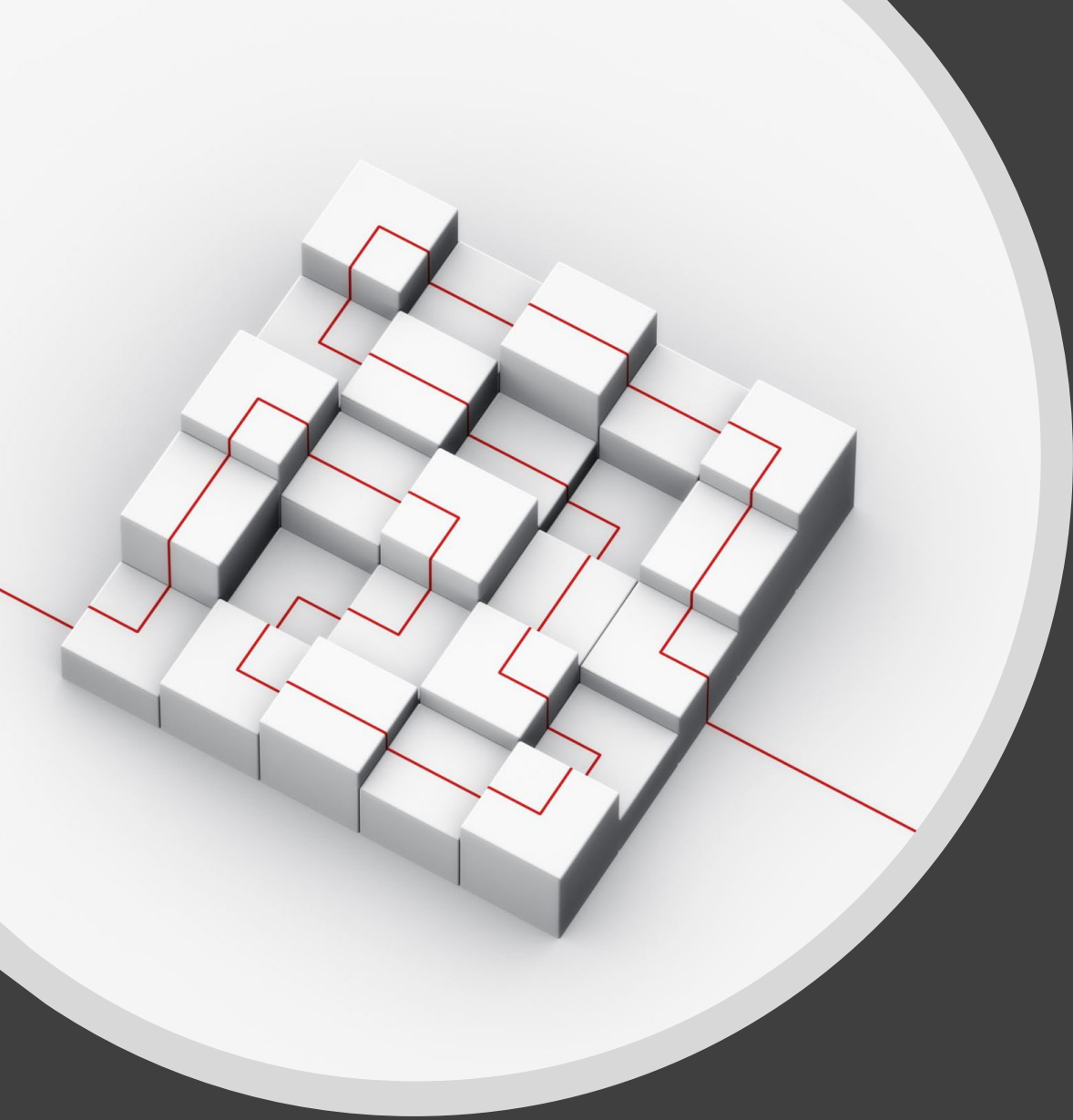
Dimensions:

Are companions to facts and describe the objects in a fact table.

Facts can be linked to multiple dimensions.



Dimension and Fact



What is Dimensional Modeling?

- A logical design technique that seeks to present the data in a standard, intuitive framework that allows for high-performance access.

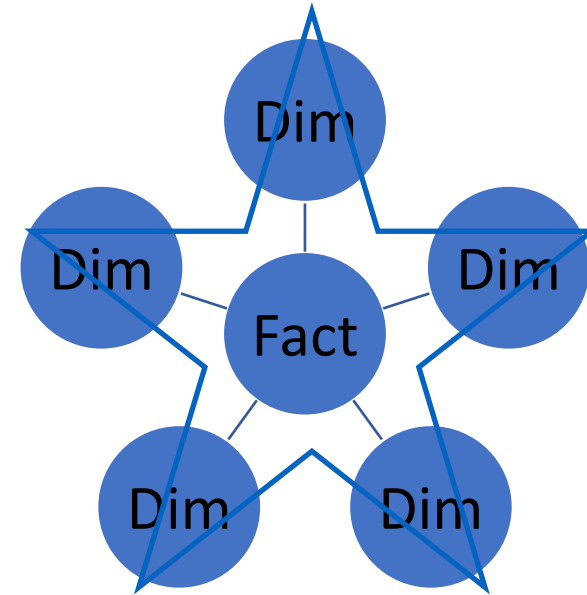


What is Dimensional Modeling?

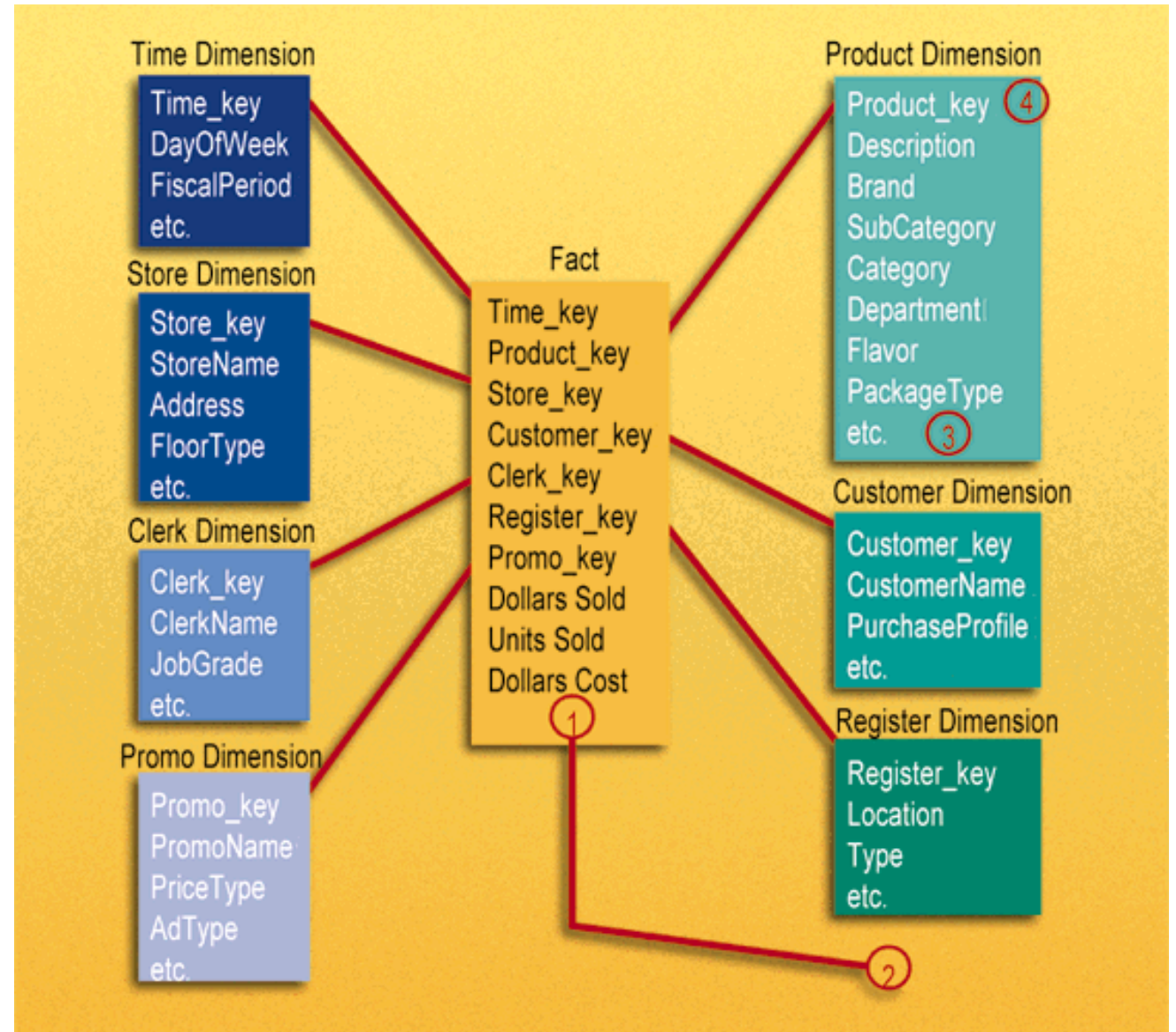
- This characteristic "**star-like**" structure is often called a star join.
 - The term star join dates back to the earliest days of relational databases.

Star Schema

- Single data (fact) table surrounded by multiple descriptive (dimension) tables



Dimensional Model Example



Dimensional Schema

- Fact Tables
 - Contain related **measures**
 - Store quantifiable business data (such as sales, expenses, and inventory). "
 - Usually, the largest tables
 - Usually appended to
 - Can contain detail or summary data
 - Measures are usually additive

Dimensional Schema

- Dimension Tables
 - Contain descriptors
 - Utilize business terminology
 - Textual and discrete data
 - Attributes through which the table measures are analyzed

Dimensional Schema – Fact Table

- A fact table contains information about things that an organization wants to measure.
- A fact table's key is made up from the keys of two or more parents.
- A fact always 'resolves' a many-to-many relationship between the parent, or dimension tables.
- The most useful fact tables also contain one or more numerical measures, or facts, that occur for the combination of keys that define each record.
- Example: the facts are Dollars Sold, Units Sold, and Dollars Cost.

Dimensional Schema – Fact Table

- The most useful facts in a fact table are numeric and additive.
- Additivity is crucial because data warehouse applications almost never retrieve a single fact table record;
 - Rather, they fetch back hundreds, thousands, or even millions of these records at a time, and often the most useful thing to do with so many records is to add them up.

Dimensional Schema – Dimension Table

- Dimension tables contain information about how an organization wants to analyze facts:
 - “Show me sales revenue (fact) for last week (time) for blue cups (product) in the western region (geography)”
- Dimension tables most often contain descriptive textual information ‘Blue cups’, ‘Western Region’
- Dimension attributes are used as the source of most of the interesting ‘constraints’ in data warehouse queries., and they are virtually always the source of the row headers in the SQL answer set

Dimensions vs Facts

Dimensions

- The time independent, textual and descriptive attributes by which users describe objects.
- Combining all the attributes including hierarchies, rollups and sub-references into a single dimension is denormalization.
- Often the “by” word in a query or report
- Not time dependent

Facts

- Business Measurements
- Most Facts are Numeric
- Additive, Semi-Additive, Non-Additive
- Built from the lowest level of detail (grain)
- Very Efficient
- Time dependent



Example

- Retail Organisation
- We want to store the information of how many of productX and how many of productY are sold daily in our shops.
- Step 1: Chose Business Objective
 - The first step in data modeling is, identify the business objective. In our example, the business objective is to store the information of how many products are sold by the online store every day.
- Step2: Identify Granularity
 - Granularity is the lowest level of information stored in the table. E.g. if the table contains daily sales data then granularity is “Daily”.

Product				Shop	
PK	NAME			PK	NAME
1	Product X			1	Shop1
2	Product Y			2	Shop2
				3	Shop3

Day	
PK	NAME
1	01/01/2022
2	01/01/2022
3	01/01/2022



Example

- Retail Organisation
- Step 3: Identify Dimensions
 - Dimensions are objects or things.
 - In our example, we are dealing with 3 things, a “Shop”, “Product”, and “Day”.
 - We have 3 dimension tables here “Shop”, “ProductType” – productX and productY, and “Day”.



Example

- Retail Organisation
- Step 4: Identify the fact
 - The Fact table holds something that is measurable.
 - In our example, number of products sold is a measure.
 - We create separate table called Fact to store the measures.
- We have 3 dimensions and 1 fact table
- STAR model

