Fundamentals of Data Engineering

Scenario

Every stakeholder interacts with the data ecosystem differently: Business executives want results and trends. Data scientists need structured data to build models. Developers need usable interfaces (APIs).Customers want value and privacy. You, the data engineer, ensure the whole data flow works efficiently and ethically.

Data Engineer

Wasting time & resources!

System Requirements

Functional Requirements	Nonfunctional Requirements

Technology Choices

- Ingestion tool
- Storage system
- Transformation tool

Module-5

1. Data Modelling, Dimensional Modelling
2. Star Schema, Data Vault, One Big Table,
3. Data Modelling and Transformation for Machine Learning,
4. Batch Transformation Patterns and Use Cases,
5. Distributed Processing Framework Spark,
6. Working with Spark Data Frames Using Python,
7. Serving Data for Analytics and Machine Learning,
8. Capstone Project

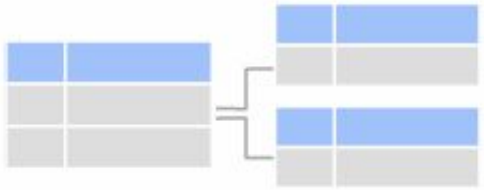
What is a Data Model?

Data Model

A data model organizes and standardizes data in a precise structured representation to enable and guide human and machine behavior, inform decision-making, and facilitate actions.

Define the structure, relationships and meaning of data.

Modeling tabular data



- What tables make up the model?
- How the tables relate to one another?
- What columns to choose for each table?

Structure the data in a way that connects back to the organization

Data is understandable & valuable

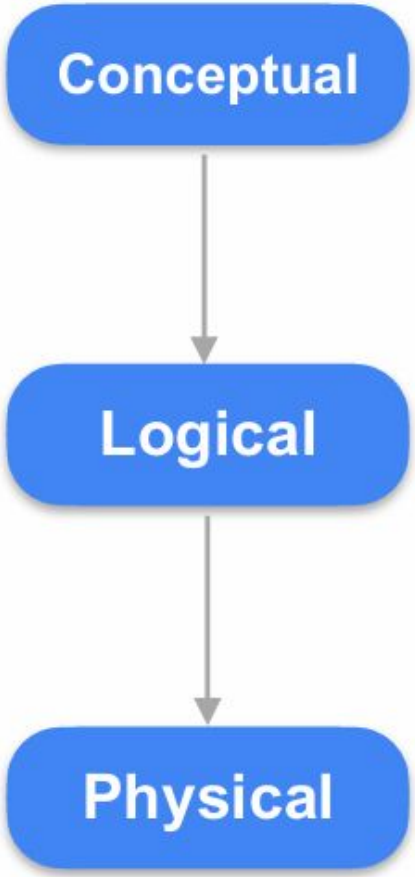
Analytics

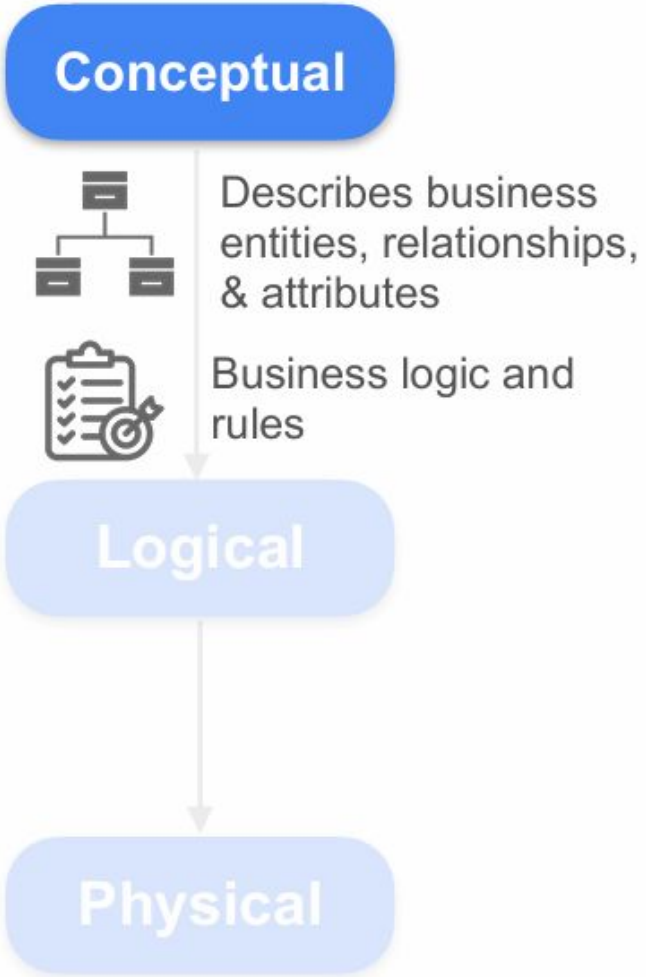
Data is meaningful

Machine

Machine Learning

Data Models





Entity-Relationship (ER) Diagram

A standard tool for visualizing the relationships among various business entities.

Conceptual

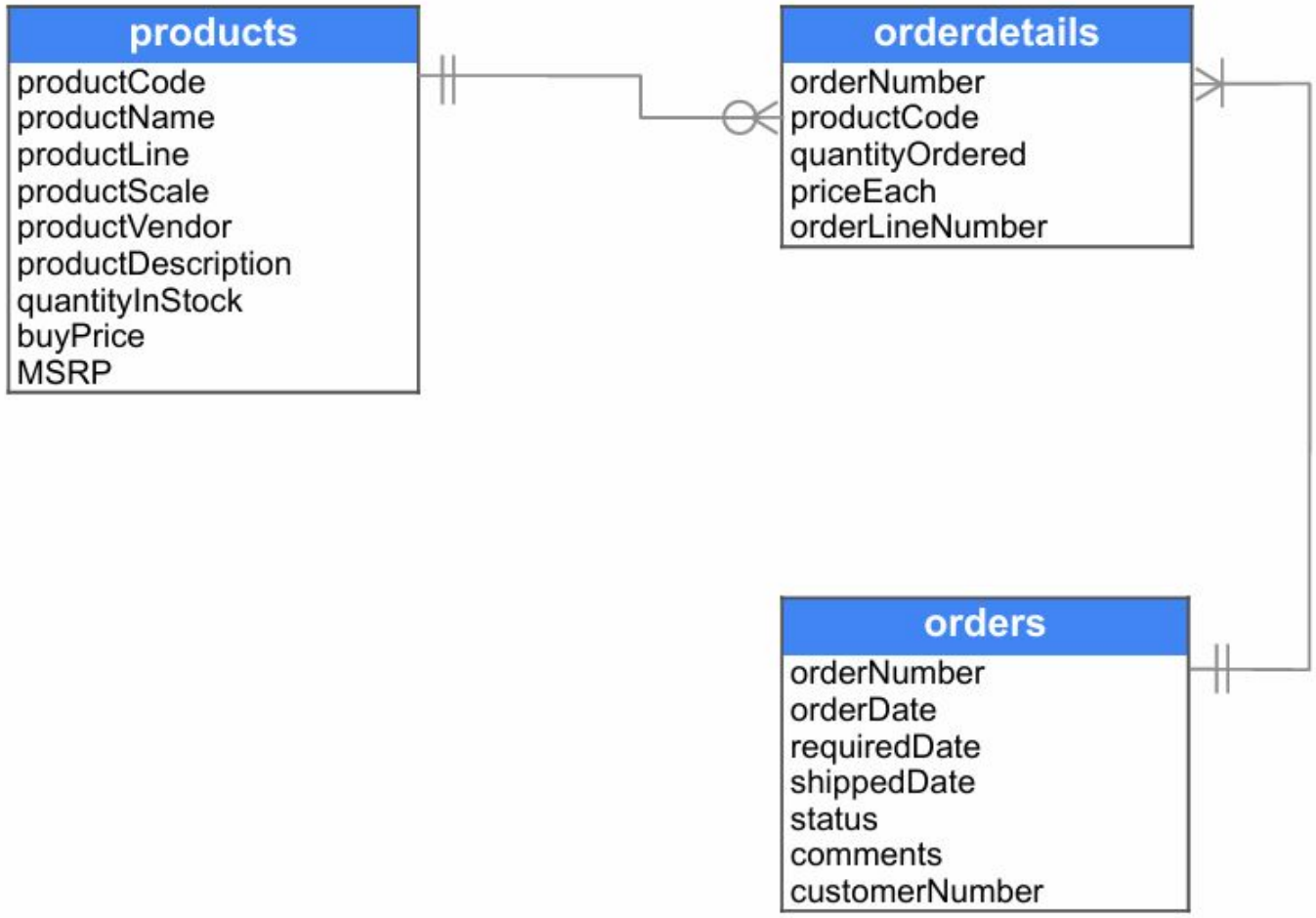
Describes business entities, relationships, & attributes

Business logic and rules

Logical

Physical

Entity-Relationship (ER) Diagram



Conceptual

Describes business entities, relationships, & attributes

Business logic and rules

Logical

Physical

Entity-Relationship (ER) Diagram

Conceptual

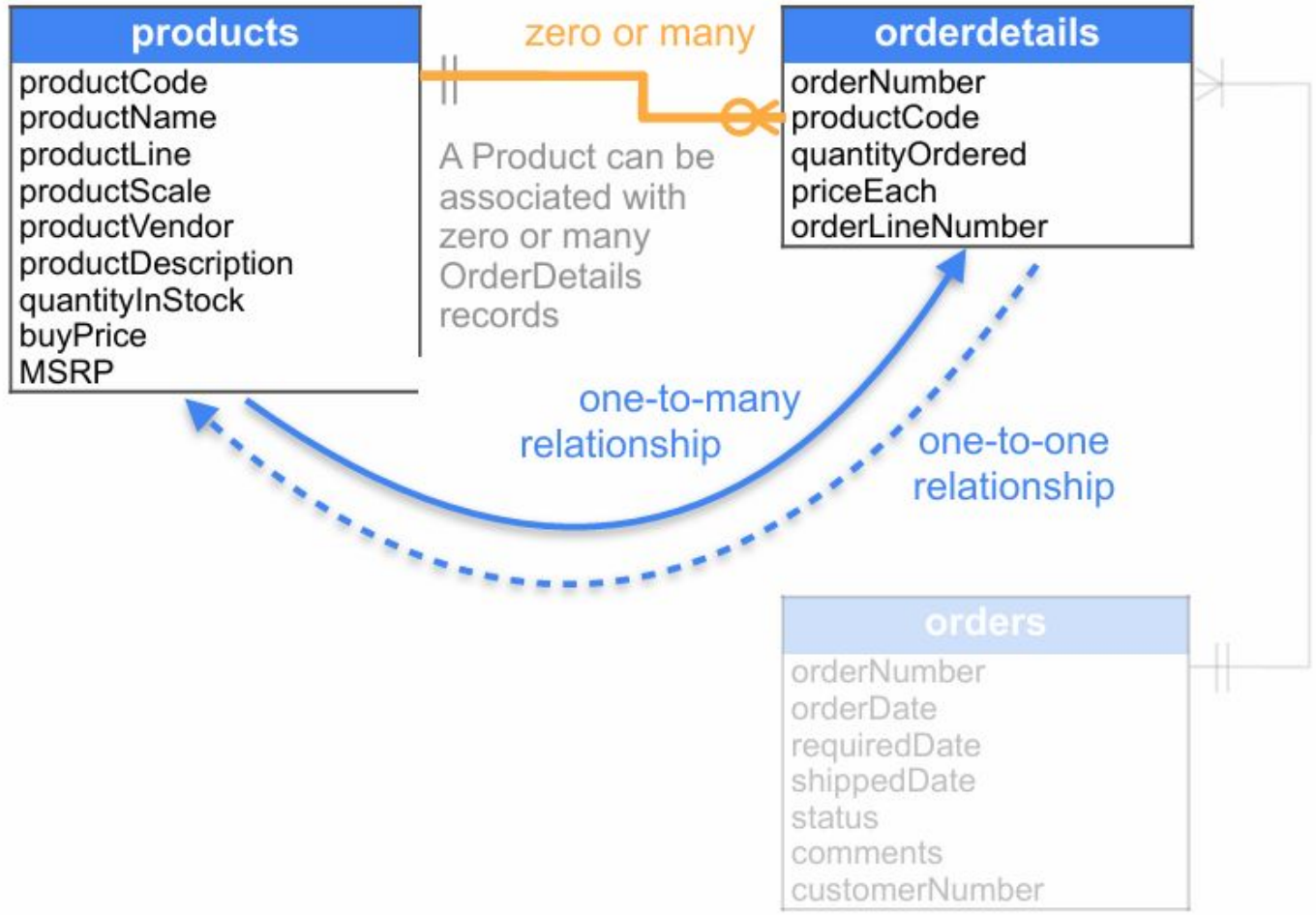
Describes business entities, relationships, & attributes

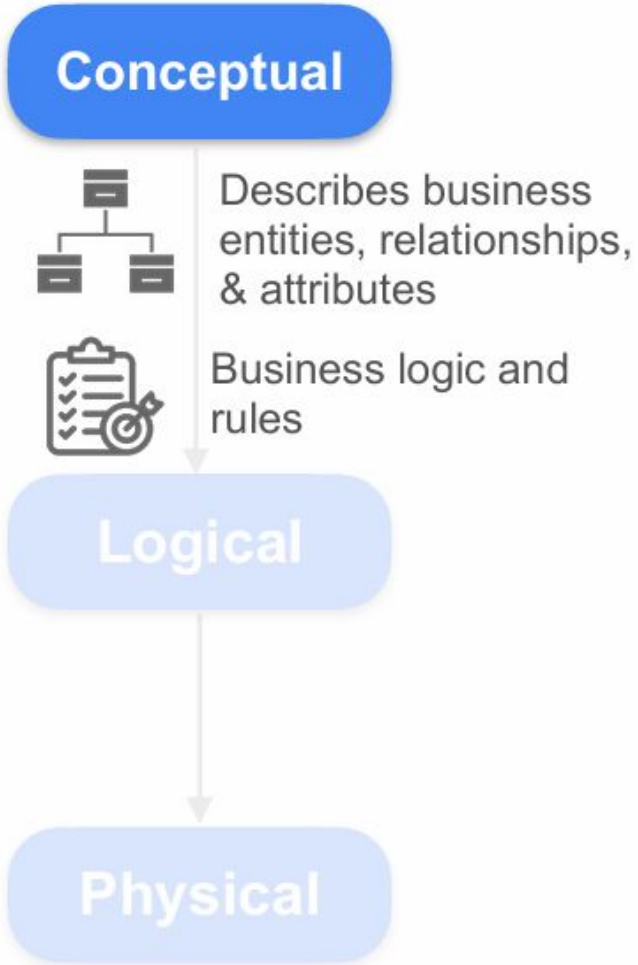
Business logic and rules

Logical

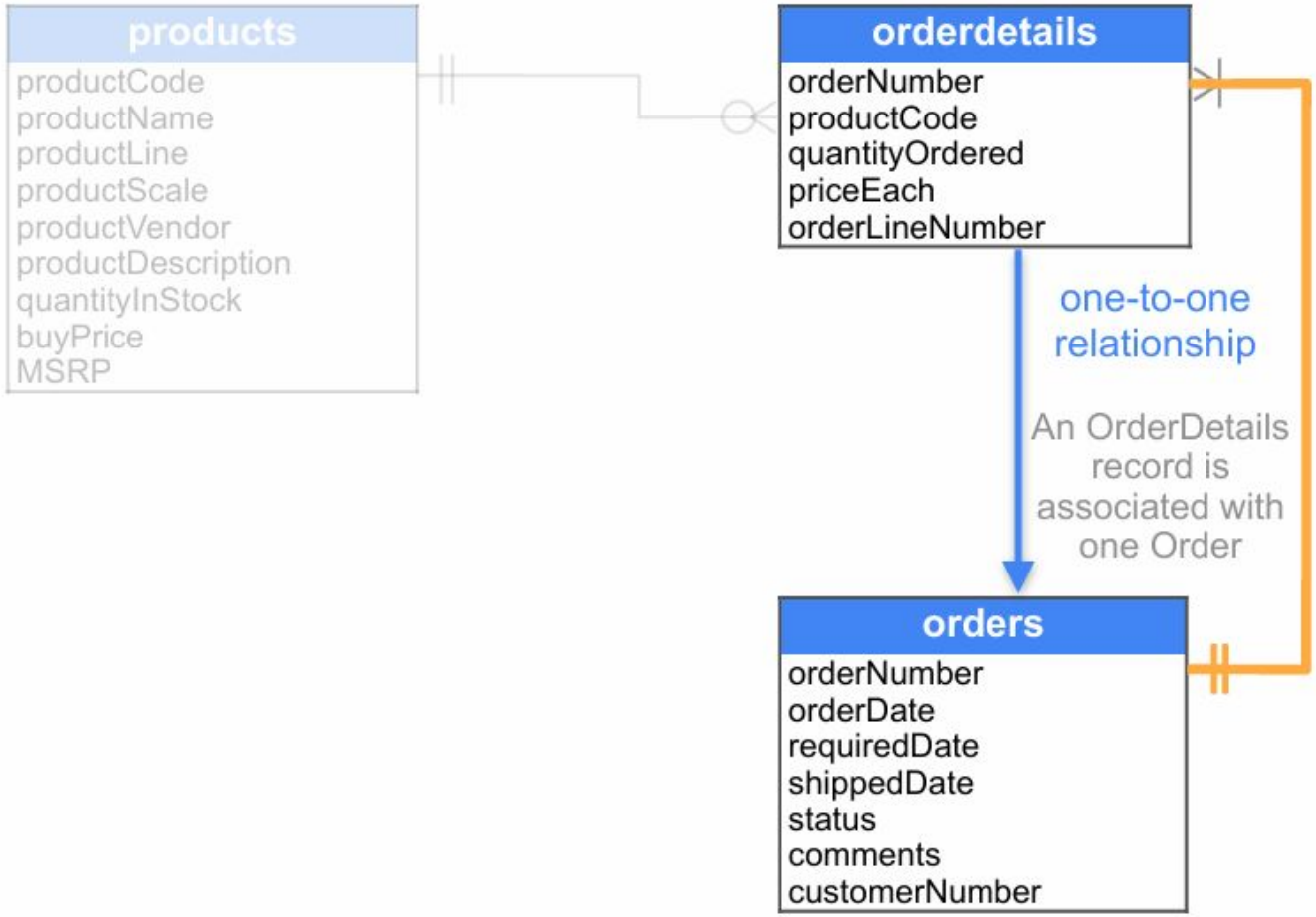
Physical

Entity-Relationship (ER) Diagram





Entity-Relationship (ER) Diagram



Entity-Relationship (ER) Diagram

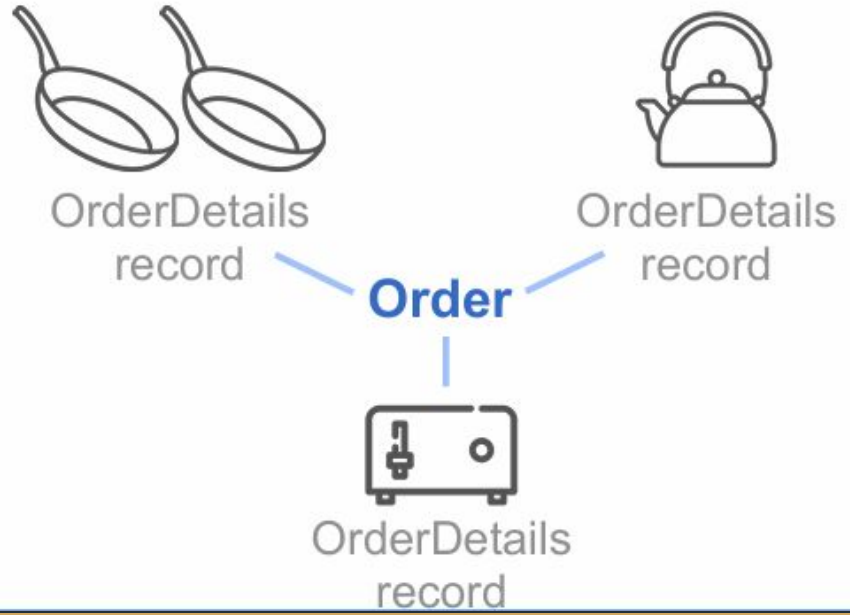
Conceptual

Describes business entities, relationships, & attributes

Business logic and rules

Logical

Physical



products

productCode
productName
productLine
productScale
productVendor
productDescription
quantityInStock
buyPrice
MSRP

orderdetails

orderNumber
productCode
quantityOrdered
priceEach
orderLineNumber

one-to-many relationship

An Order can be associated with many OrderDetails

one-to-one relationship

An OrderDetails record is associated with one Order

orders

orderNumber
orderDate
requiredDate
shippedDate
status
comments
customerNumber

Conceptual

Describes business entities, relationships, & attributes

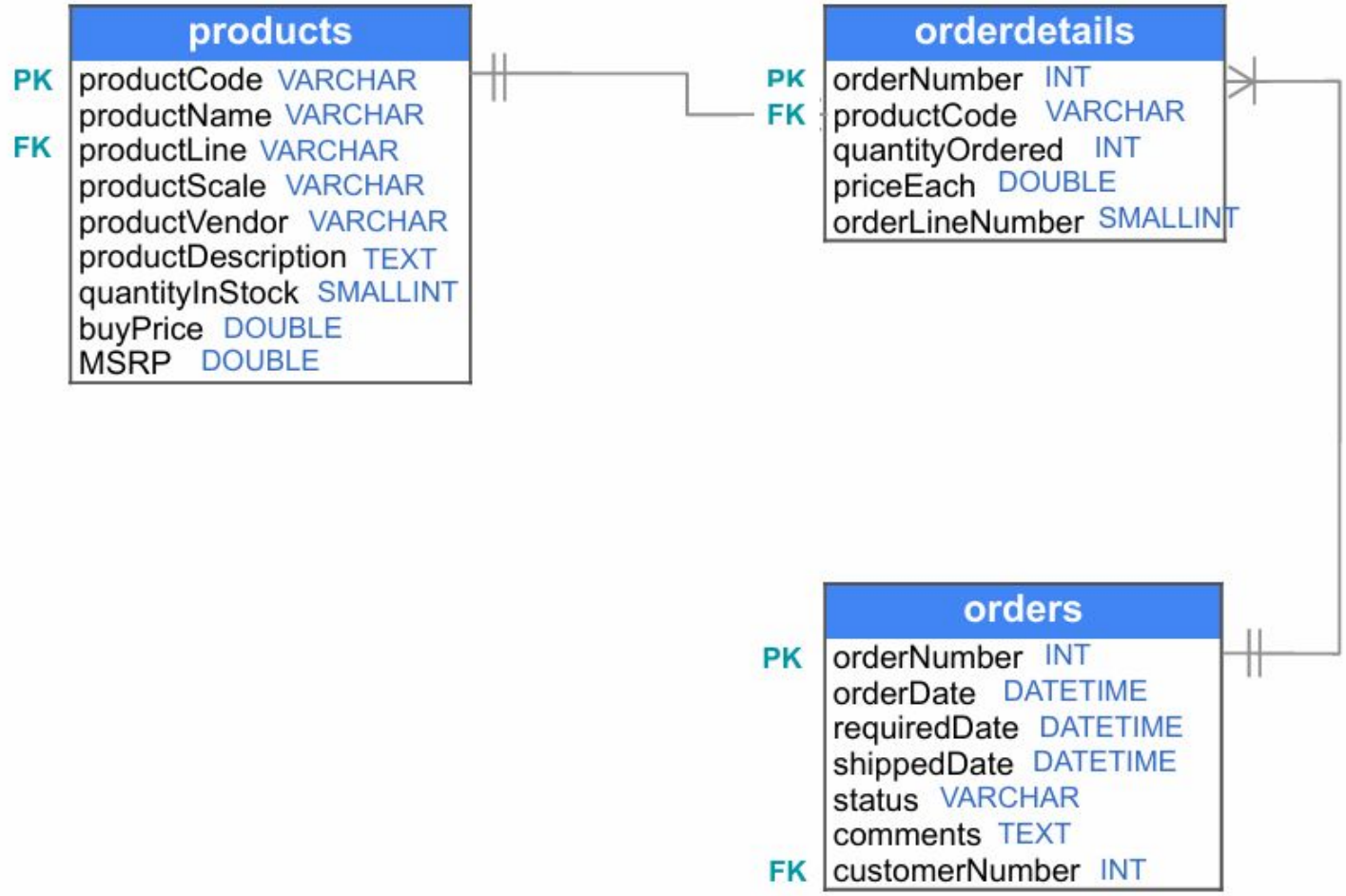
Business logic and rules

Logical

Details about the implementation of the conceptual model

Physical

Entity-Relationship (ER) Diagram



Conceptual

Describes business entities, relationships, & attributes

Business logic and rules

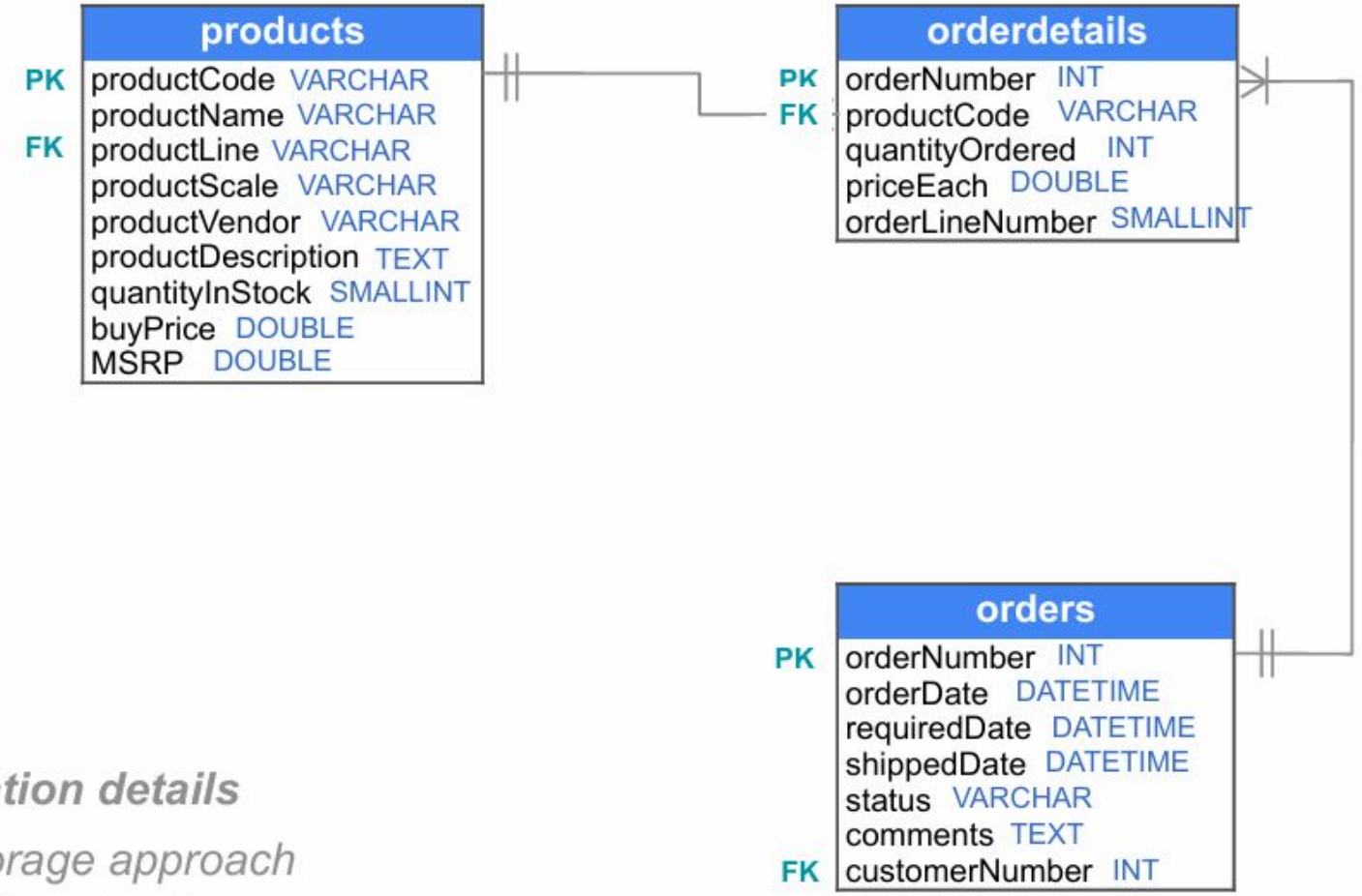
Logical

Details about the implementation of the conceptual model

Physical

Details about the implementation of the logical model in a specific DBMS

Entity-Relationship (ER) Diagram



Configuration details

- *Data storage approach*
- *Partitioning details*
- *Replication details*

Conceptual

Describes business entities, relationships, & attributes

Business logic and rules

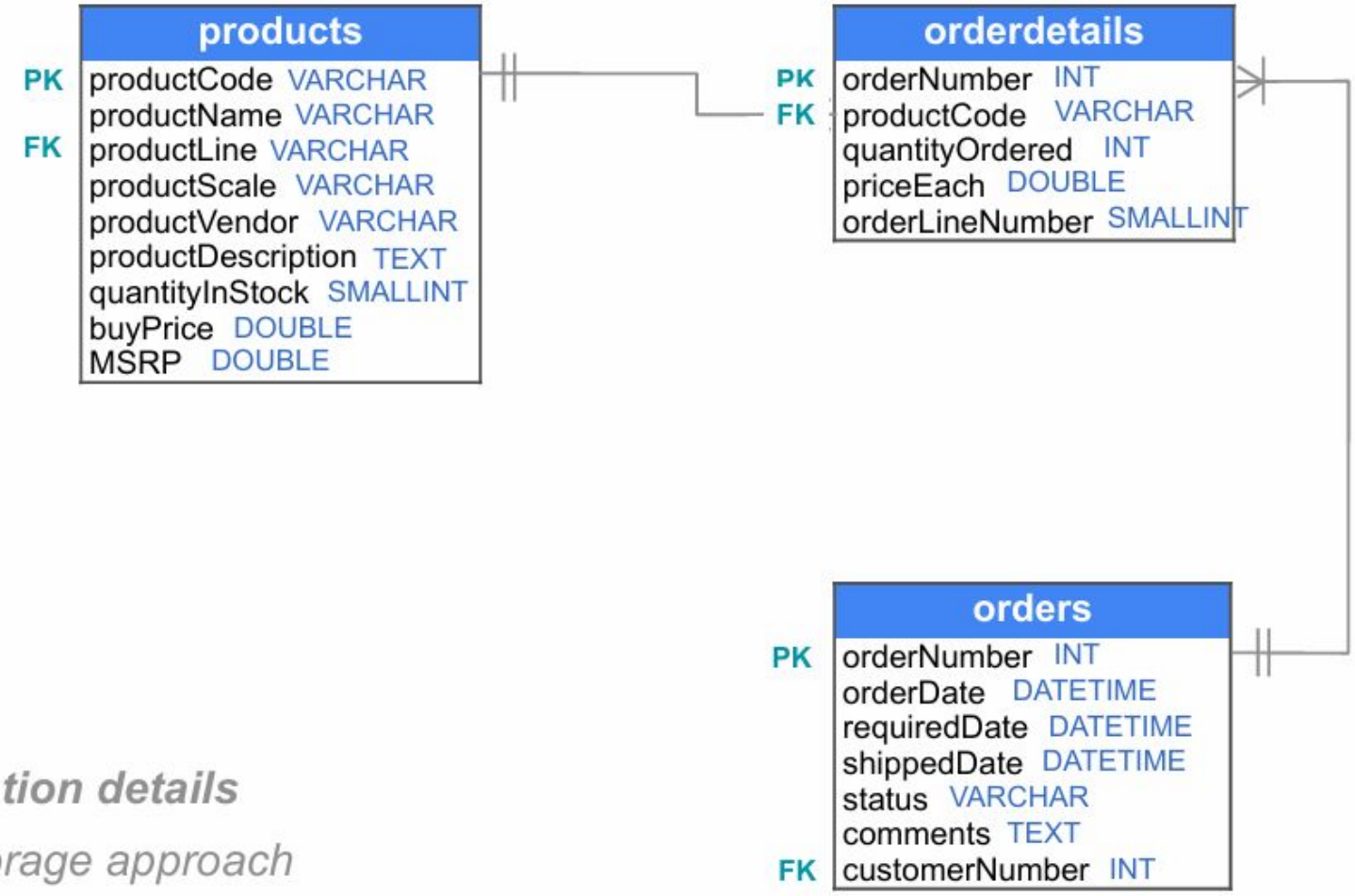
Logical

Details about the implementation of the conceptual model

Physical

Details about the implementation of the logical model in a specific DBMS

Entity-Relationship (ER) Diagram



Configuration details

- *Data storage approach*
- *Partitioning details*
- *Replication details*

Intro to Data Modeling for Analytics

Normalization

Normalization

Normalization

A data modeling practice typically applied to relational databases to remove the redundancy of data within a database and ensure referential integrity between tables.

Edgar Codd

Codd's Objectives of Normalization:

- To free the collection of relations from undesirable insertion, update, and deletion dependencies
- To reduce the need for restructuring the collection of relations, as new types of data are introduced

SalesOrders

OrderID	ItemNumber	sku	price	quantity	name	CustomerID	CustomerName	address	OrderDate
100	1	1	50	1	Thingamajig	5	Joe Reis	1st. St	1/08/2024
100	2	2	25	2	Whatchamacallit	5	Joe Reis	1st. St	1/08/2024
101	1	3	75	1	Whoozeewhatzit	7	Matt Housely	2nd Ave.	1/08/2024
101	2	2	25	3	Whatchamacallit	7	Matt Housely	2nd Ave.	1/08/2024
102	1	1	50	1	Thingamjig	9	Colleen Fotsch	3rd Lk	1/08/2024

Customers

CustomerID	CustomerName	address
5	Joe Reis	1st. St
7	Matt Housely	2nd Ave.
9	Colleen Fotsch	3rd Lk

Items

sku	price	name
1	50	Thingamajig
2	25	Whatchamacallit
3	75	Whoozeewhatzit

Order Items

OrderID	ItemNumber	sku	quantity
100	1	1	1
100	2	2	2
101	1	3	1
101	2	2	3
102	1	1	1

Orders

OrderID	CustomerID	Date
100	5	1/08/2024
101	7	1/08/2024
102	9	1/08/2024

SalesOrders

OrderID	ItemNumber	sku	price	quantity	name	CustomerID	CustomerName	address	OrderDate
100	1	1	50	1	Thingamajig	5	Joe Reis	1st. St	1/08/2024
100	2	2	25	2	Whatchamacallit	5	Joe Reis	1st. St	1/08/2024
101	1	3	75	1	Whoozeewhatzit	7	Matt Housely	2nd Ave.	1/08/2024
101	2	2	25	3	Whatchamacallit	7	Matt Housely	2nd Ave.	1/08/2024
102	1	1	50	1	Thingamjig	9	Colleen Fotsch	3rd Lk	1/08/2024

Customers

CustomerID	CustomerName	address
5	Joe Reis	1st. St
7	Matt Housely	2nd Ave.
9	Colleen Fotsch	3rd Lk

Items

sku	price	name
1	50	Thingamajig
2	25	Whatchamacallit
3	75	Whoozeewhatzit

Less normalized:
Contains more redundant data

Order Items

OrderID	ItemNumber	sku	quantity
100	1	1	1
100	2	2	2
101	1	3	1
101	2	2	3
102	1	1	1

Orders

OrderID	CustomerID	Date
100	5	1/08/2024
101	7	1/08/2024
102	9	1/08/2024

SalesOrders

OrderID	ItemNumber	sku	price	quantity	name	CustomerID	CustomerName	address	OrderDate
100	1	1	50	1	Thingamajig	5	Joe Reis	1st. St	1/08/2024
100	2	2	25	2	Whatchamacallit	5	Joe Reis	1st. St	1/08/2024
101	1	3	75	1	Whoozeewhatzit	7	Matt Housely	2nd Ave.	1/08/2024
101	2	2	25	3	Whatchamacallit	7	Matt Housely	2nd Ave.	1/08/2024
102	1	1	50	1	Thingamjig	9	Colleen Fotsch	3rd Lk	1/08/2024

Customers

CustomerID	CustomerName	address
5	Joe Reis	1st. St
7	Matt Housely	2nd Ave.
9	Colleen Fotsch	3rd Lk

Items

sku	price	name
1	50	Thingamajig
2	25	Whatchamacallit
3	75	Whoozeewhatzit

Order Items

OrderID	ItemNumber	sku	quantity
100	1	1	1
100	2	2	2
101	1	3	1
101	2	2	3
102	1	1	1

Orders

OrderID	CustomerID	Date
100	5	1/08/2024
101	7	1/08/2024
102	9	1/08/2024

SalesOrders

OrderID	ItemNumber	sku	price	quantity	name	CustomerID	CustomerName	address	
100	1	1	50	1	Thingamajig	5	Joe Reis	1st. St	
100	2	2	25	2	Whatchamacallit	5	Joe Reis	1st. St	1/08/2024
101	1	3	75	1	Whoozeewhatzit	7	Matt Housely	2nd Ave.	1/08/2024
101	2	2	25	3	Whatchamacallit	7	Matt Housely	2nd Ave.	1/08/2024
102	1	1	50	1	Thingamjig	9	Colleen Fotsch	3rd Lk	1/08/2024

First Normal Form

Customers

CustomerID	CustomerName	address
5	Joe Reis	1st. St
7	Matt Housely	2nd Ave.
9	Colleen Fotsch	3rd Lk

Items

sku	price	name
1	50	Thingamajig
2	25	Whatchamacallit
3	75	Whoozeewhatzit

Shipments

ShipmentID	OrderID	Ship.Details

Order Items

OrderID	ItemNumber	sku	quantity
100	1	1	1
100	2	2	2
101	1	3	1
101	2	2	3
102	1	1	1

Orders

OrderID	CustomerID	Date
100	5	1/08/2024
101	7	1/08/2024
102	9	1/08/2024

Third Normal Form

Denormalized Form

- contains redundant data
- contains nested data

1NF

- Each column must:
  - be unique
  - have a single value
- Unique primary key

SalesOrders

OrderID	OrderItems	CustomerID	CustomerName	address	OrderDate
100	[{ “sku”:1, “price”:50, “quantity”: 1, “name”: “Thingamajig” }, { “sku”:2, “price”:25, “quantity”: 3, “name”: “Whatchamacallit” }]	5	Joe Reis	1st. St	1/08/2024
101	[{ “sku”:3, “price”:75, “quantity”: 1, “name”: “Whoozeewhatzit” }, { “sku”:2, “price”:25, “quantity”: 3, “name”: “Whatchamacallit” }]	7	Matt Housely	2nd Ave.	1/08/2024
102	[{ “sku”:1, “price”:50, “quantity”: 1, “name”: “Thingamajig” }]	9	Colleen Fotsch	2nd Ave.	1/08/2024

1NF

SalesOrders

OrderID	sku	price	quantity	name	CustomerID	CustomerName	address	OrderDate
100	1	50	1	Thingamajig	5	Joe Reis	1st. St	1/08/2024
100	2	25	2	Whatchamacallit	5	Joe Reis	1st. St	1/08/2024
101	3	75	1	Whoozeewhatzit	7	Matt Housely	2nd Ave.	1/08/2024
101	2	25	3	Whatchamacallit	7	Matt Housely	2nd Ave.	1/08/2024
102	1	50	1	Thingamjig	9	Colleen Fotsch	3rd Lk	1/08/2024

1NF

SalesOrders

OrderID	ItemNumber	sku	price	quantity	name	CustomerID	CustomerName	address	OrderDate
100	1	1	50	1	Thingamajig	5	Joe Reis	1st. St	1/08/2024
100	2	2	25	2	Whatchamacallit	5	Joe Reis	1st. St	1/08/2024
101	1	3	75	1	Whoozeewhatzit	7	Matt Housely	2nd Ave.	1/08/2024
101	2	2	25	3	Whatchamacallit	7	Matt Housely	2nd Ave.	1/08/2024
102	1	1	50	1	Thingamjig	9	Colleen Fotsch	3rd Lk	1/08/2024

2NF

- The requirements of 1NF must be met
- **Partial dependencies** should be removed

Partial Dependency

A subset of non-key columns that depend on some columns in the composite key

1NF

SalesOrders

OrderID	ItemNumber	sku	price	quantity	name	CustomerID	CustomerName	address	OrderDate
100	1	1	50	1	Thingamajig	5	Joe Reis	1st. St	1/08/2024
100	2	2	25	2	Whatchamacallit	5	Joe Reis	1st. St	1/08/2024
101	1	3	75	1	Whoozeewhatzit	7	Matt Housely	2nd Ave.	1/08/2024
101	2	2	25	3	Whatchamacallit	7	Matt Housely	2nd Ave.	1/08/2024
102	1	1	50	1	Thingamjig	9	Colleen Fotsch	3rd Lk	1/08/2024

2NF

Order Items

Orders

order ID
100
101
102

3NF

- The requirements of 2NF must be met
- **Transitive dependencies** should be removed

Transitive Dependency — A non-key column depends on another non-key column

2NF

Order Items

OrderID	ItemNumber	sku	price	quantity	name
100	1	1	50	1	Thingamajig
100	2	2	25	2	Whatchamacallit
101	1	3	75	1	Whoozeewhatzit
101	2	2	25	3	Whatchamacallit
102	1	1	50	1	Thingamjig

Orders

order ID	CustomerID	CustomerName	address	OrderDate
100	5	Joe Reis	1st. St	1/08/2024
101	7	Matt Housely	2nd Ave.	1/08/2024
102	9	Colleen Fotsch	3rd Lk	1/08/2024

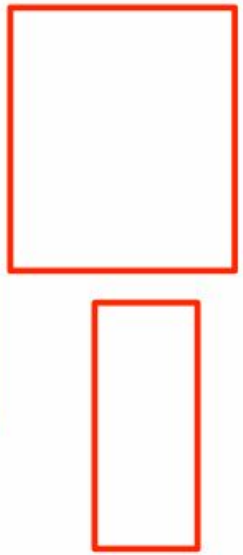
3NF

Order Items

OrderID	ItemNumber	sku	quantity
100	1	1	1
100	2	2	2
101	1	3	1
101	2	2	3
102	1	1	1

Orders

OrderID	CustomerID	OrderDate
100	5	1/08/2024
100	5	1/08/2024
101	7	1/08/2024
101	7	1/08/2024
102	9	1/08/2024



Customers

Items

2NF

Order Items

OrderID	ItemNumber	sku	price	quantity	name
100	1	1	50	1	Thingamajig
100	2	2	25	2	Whatchamacallit
101	1	3	75	1	Whoozeewhatzit
101	2	2	25	3	Whatchamacallit
102	1	1	50	1	Thingamjig

Orders

order ID	CustomerID	CustomerName	address	OrderDate
100	5	Joe Reis	1st. St	1/08/2024
101	7	Matt Housely	2nd Ave.	1/08/2024
102	9	Colleen Fotsch	3rd Lk	1/08/2024

3NF

Order Items

OrderID	ItemNumber	sku	quantity
100	1	1	1
100	2	2	2
101	1	3	1
101	2	2	3
102	1	1	1

Orders

OrderID	CustomerID	OrderDate
100	5	1/08/2024
100	5	1/08/2024
101	7	1/08/2024
101	7	1/08/2024
102	9	1/08/2024

Customers

CustomerID	CustomerName	address
5	Joe Reis	1st. St
7	Matt Housely	2nd Ave.
9	Colleen Fotsch	3rd Lk

Items

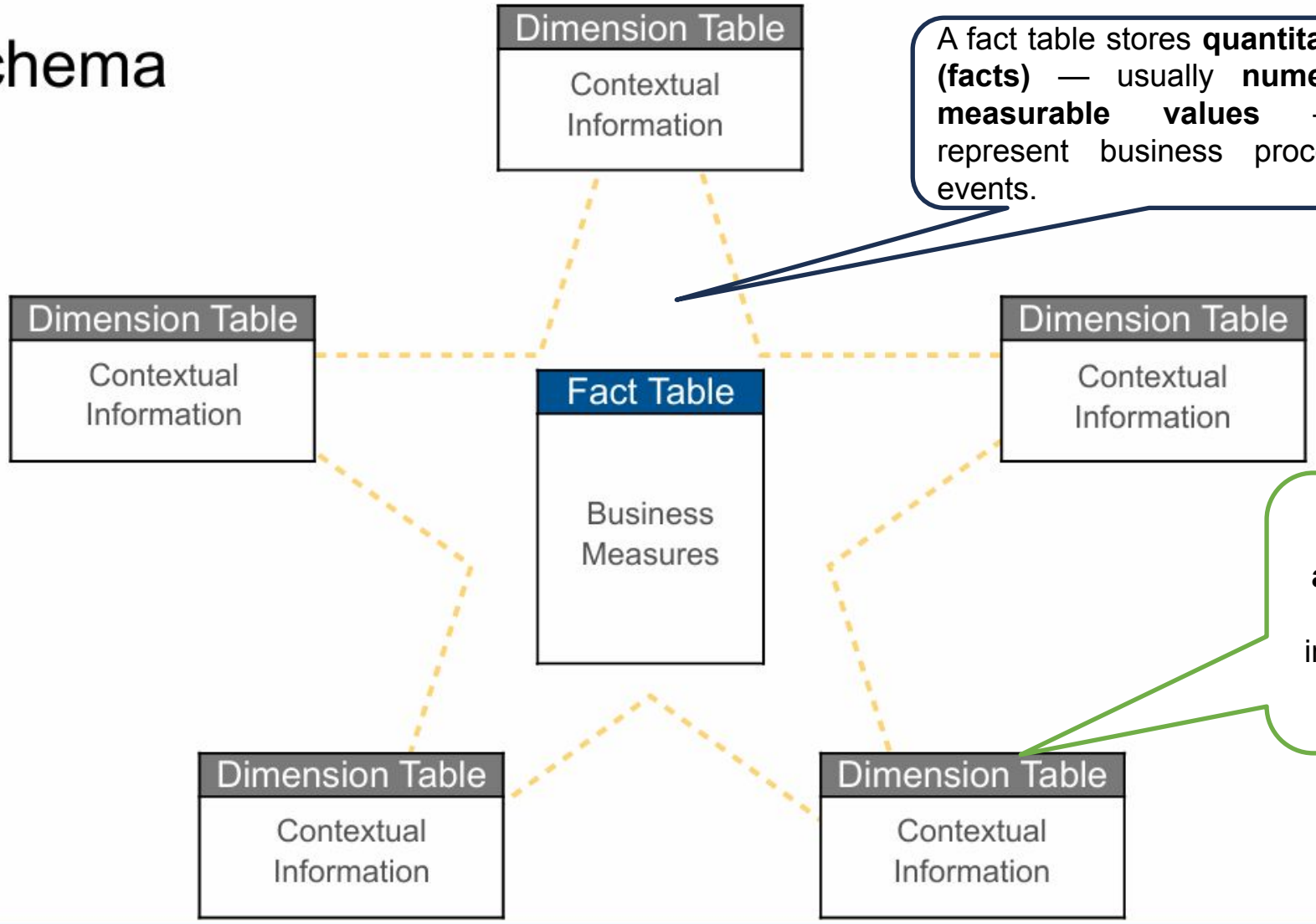
sku	price	name
1	50	Thingamajig
2	25	Whatchamacallit
3	75	Whoozeewhatzit

Convention: A normalized database means it's in third normal form.

Intro to Data Modeling for Analytics

Dimensional Modeling - Star Schema

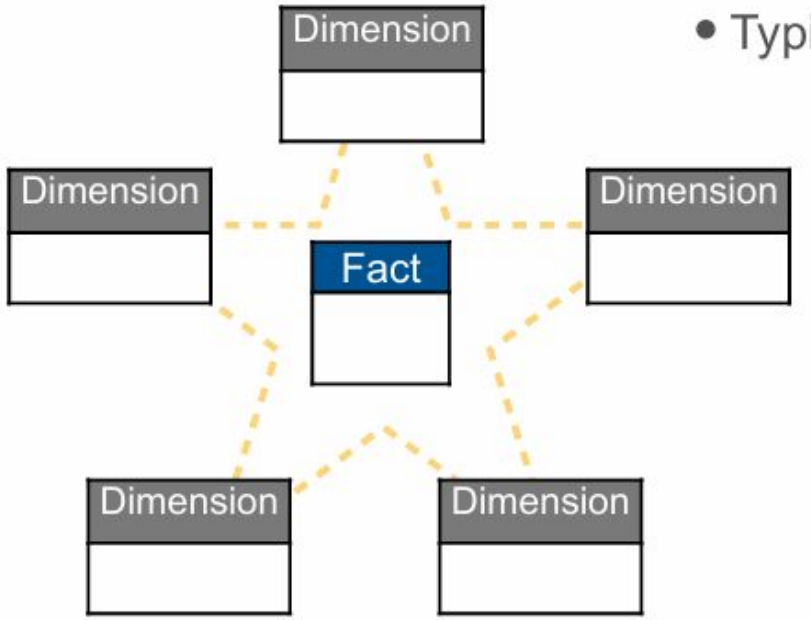
Star Schema


Dimension Table
Contextual Information
Dimension Table
Contextual Information
Fact Table
Business Measures
Dimension Table
Contextual Information
Dimension Table
Contextual Information
Dimension Table
Contextual Information
A fact table stores **quantitative data (facts)** — usually **numeric and measurable values** — that represent business processes or events.
A dimension table stores **descriptive attributes** (textual or categorical information) about the entities involved in facts.

Fact Table

Contains quantitative business measurements that result from a business event or process.

- Each row contains the facts of a particular business event
- Immutable (append-only)
- Typically narrow and long

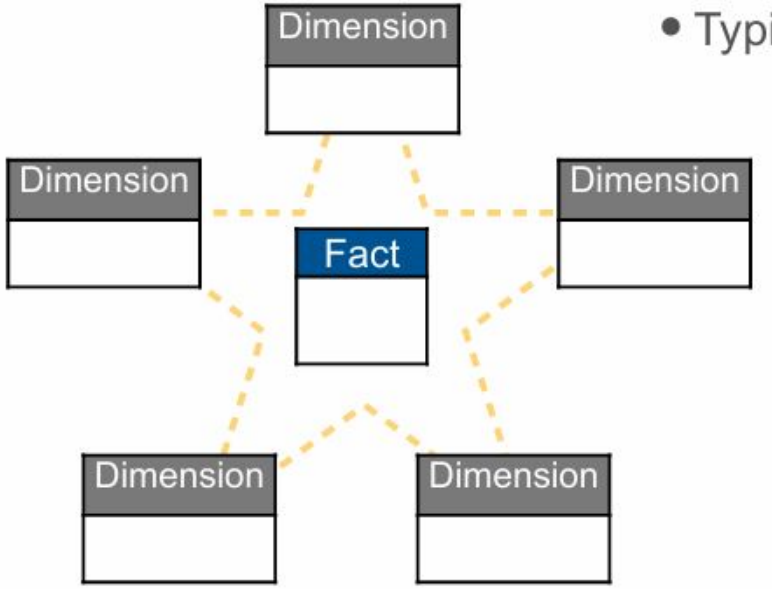


Business Event	Facts	Atomic Grain
Order a ride share	Trip duration, trip price, tip paid, trip delays, etc.	One completed ride by a customer

Fact Table

Contains quantitative business measurements that result from a business event or process.

- Each row contains the facts of a particular business event
- Immutable (append-only)
- Typically narrow and long



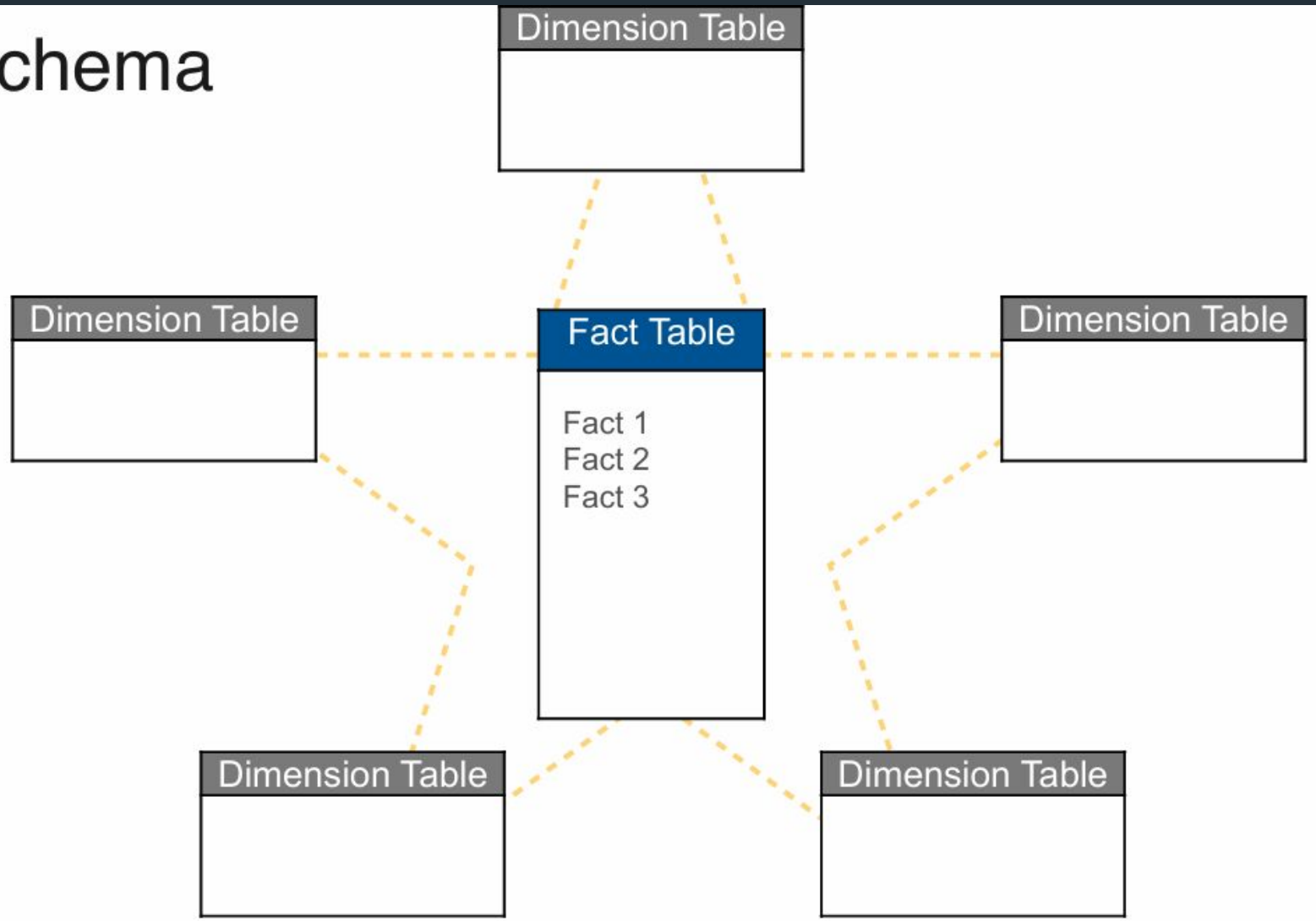
Business Event	Facts	Atomic Grain	Dimensions
Order a ride share	Trip duration, trip price, tip paid, trip delays, etc.	One completed ride by a customer	Customers, drivers, trip locations

Dimension Tables

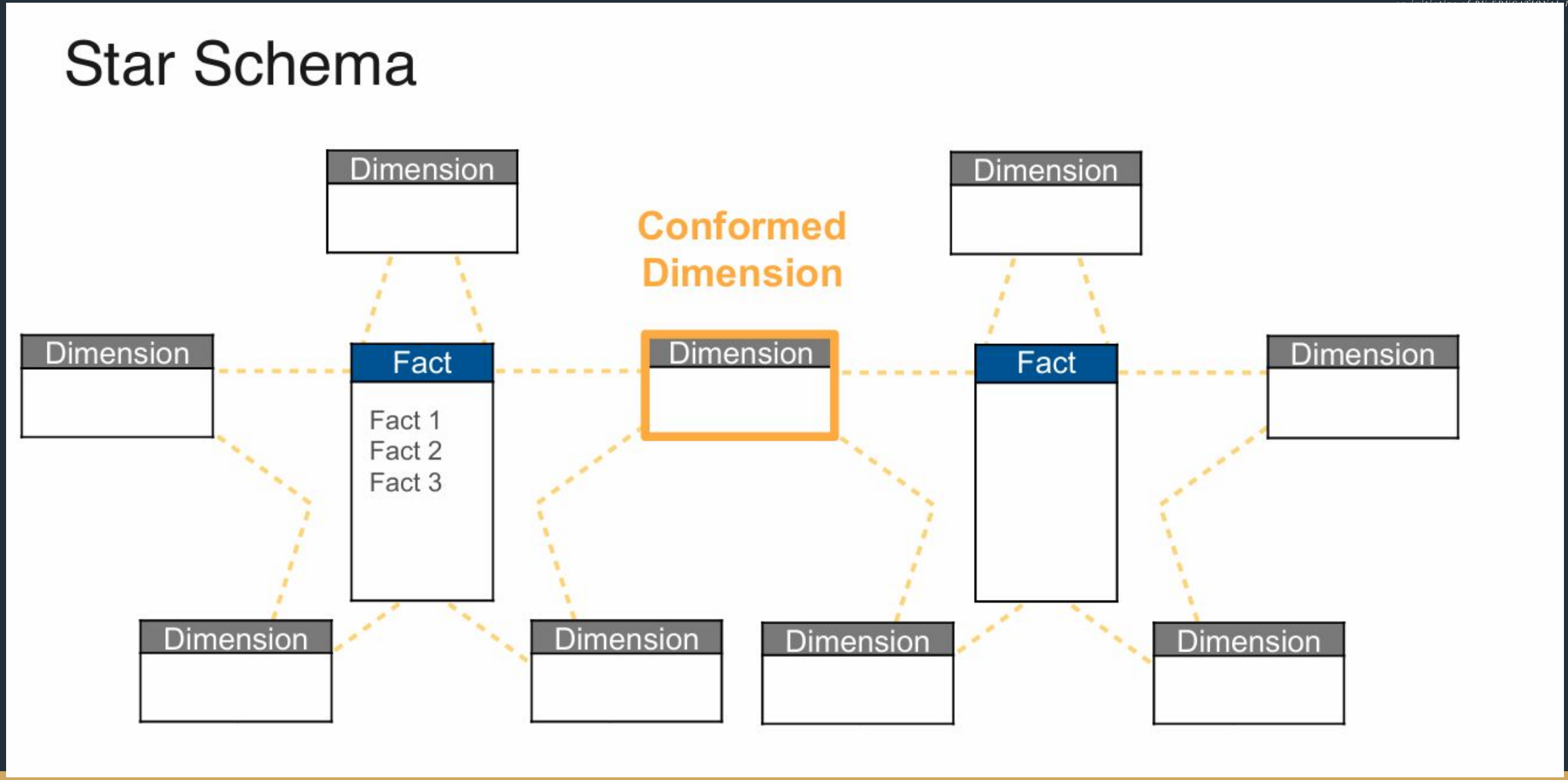
Provide the reference data, attributes and relational context for the events in the fact table.

- Describe the events' *what, who, where and when*
- Typically wide and short

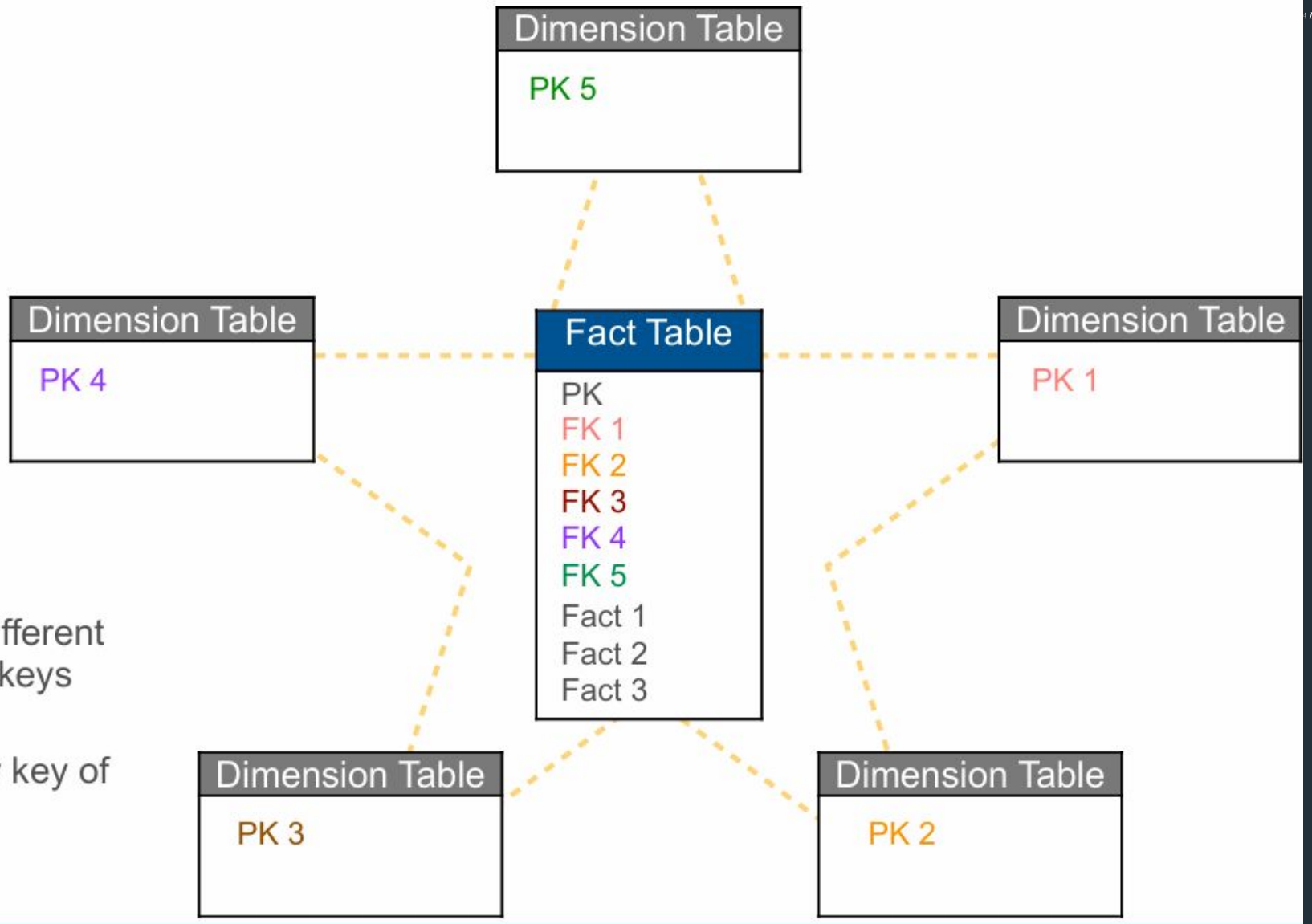
Star Schema


Dimension Table
Dimension Table
Fact Table
Fact 1
Fact 2
Fact 3
Dimension Table
Dimension Table
Dimension Table

Galaxy schema



Star Schema

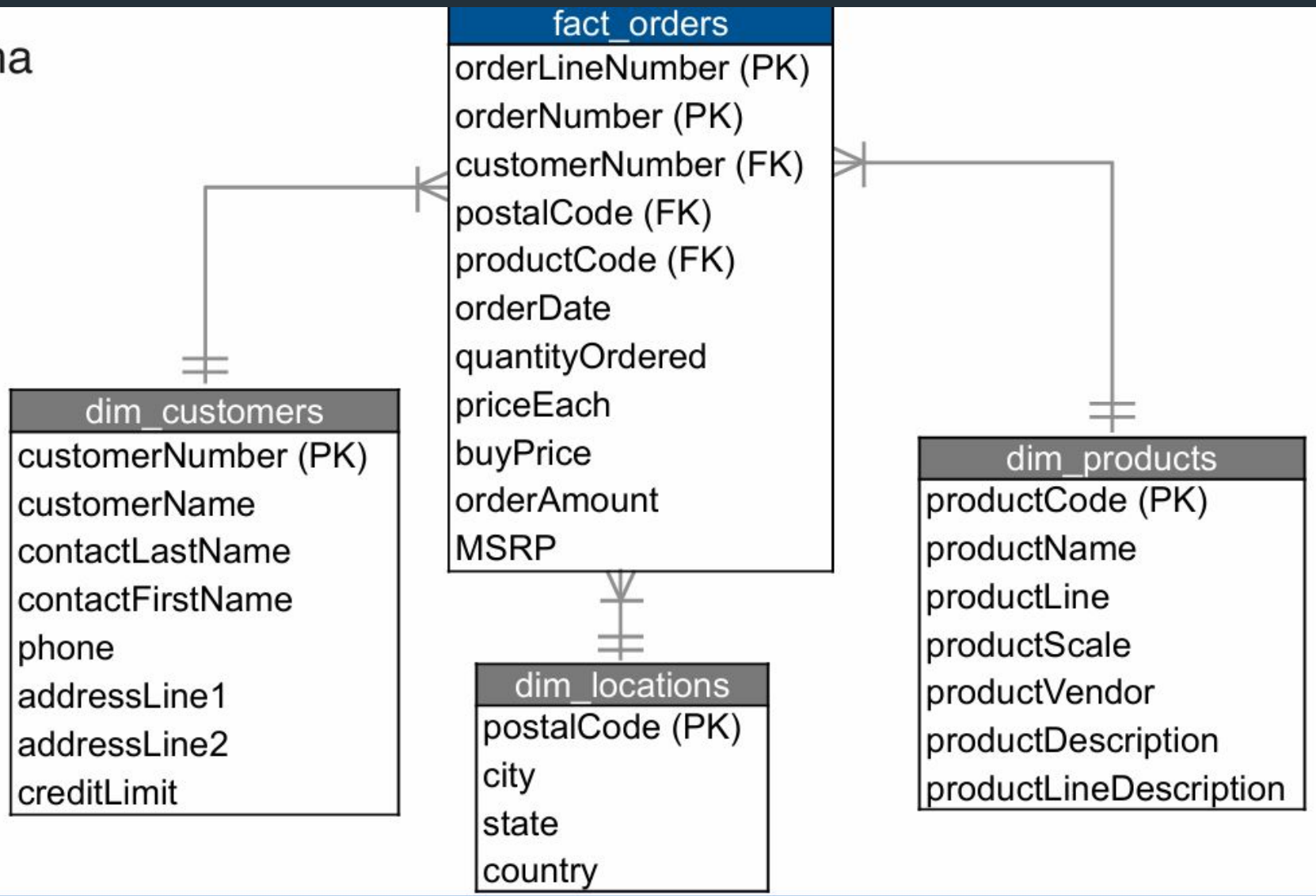


Best practice:

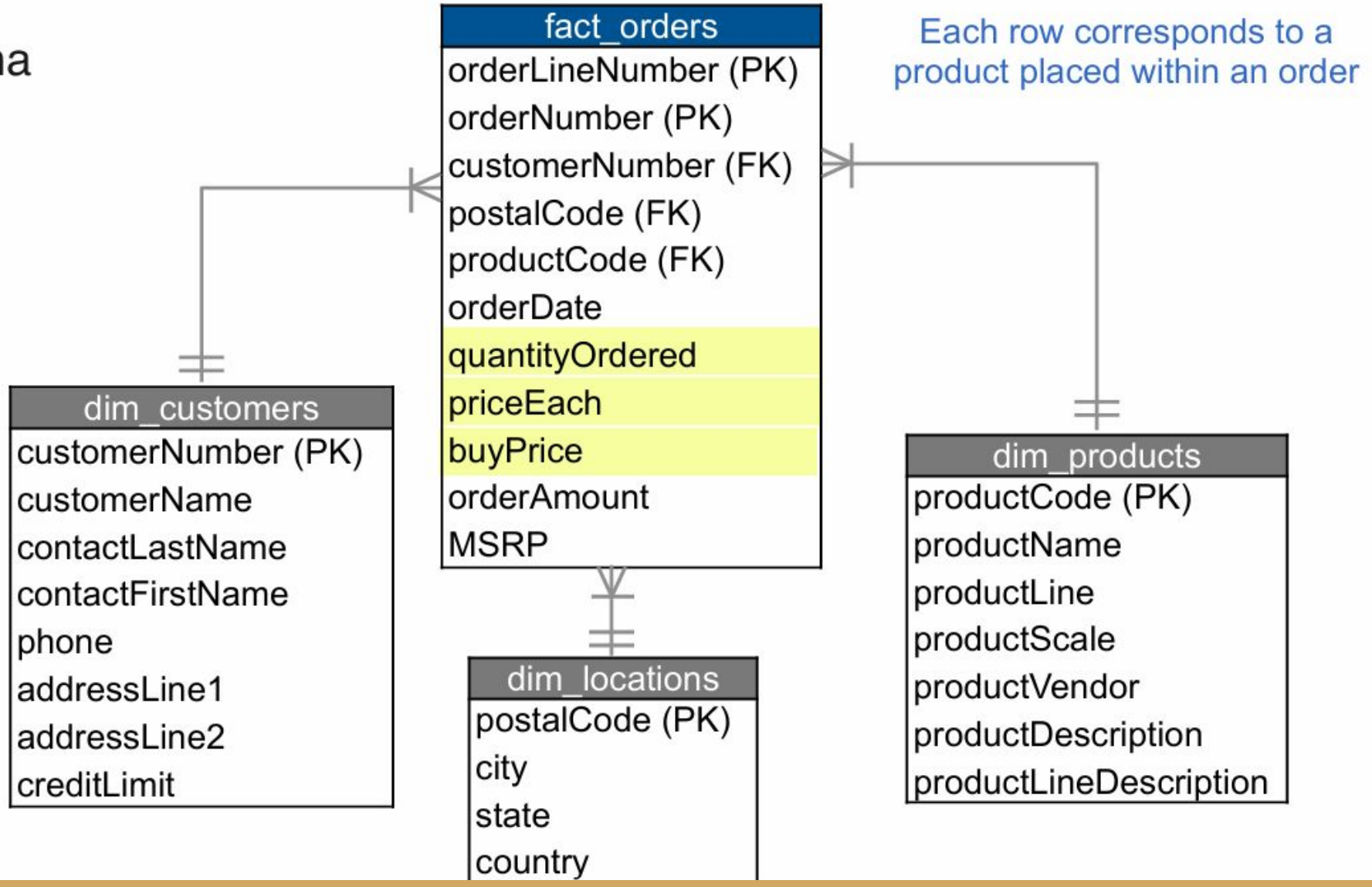
Create a surrogate key

- Used to combine data from different systems with natural primary keys that are in different formats
- Used to decouple the primary key of the star schema from source systems

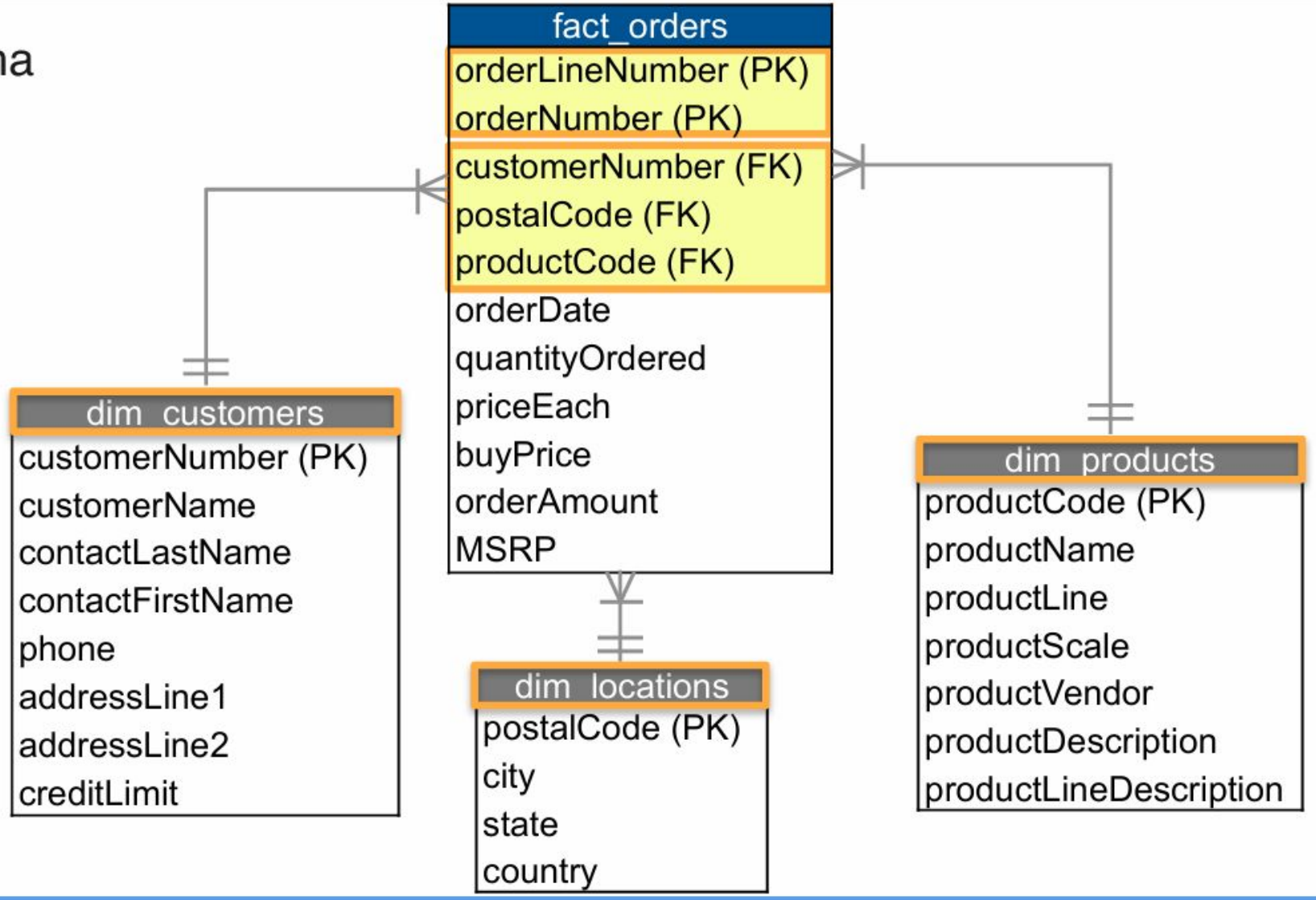
Star Schema Example



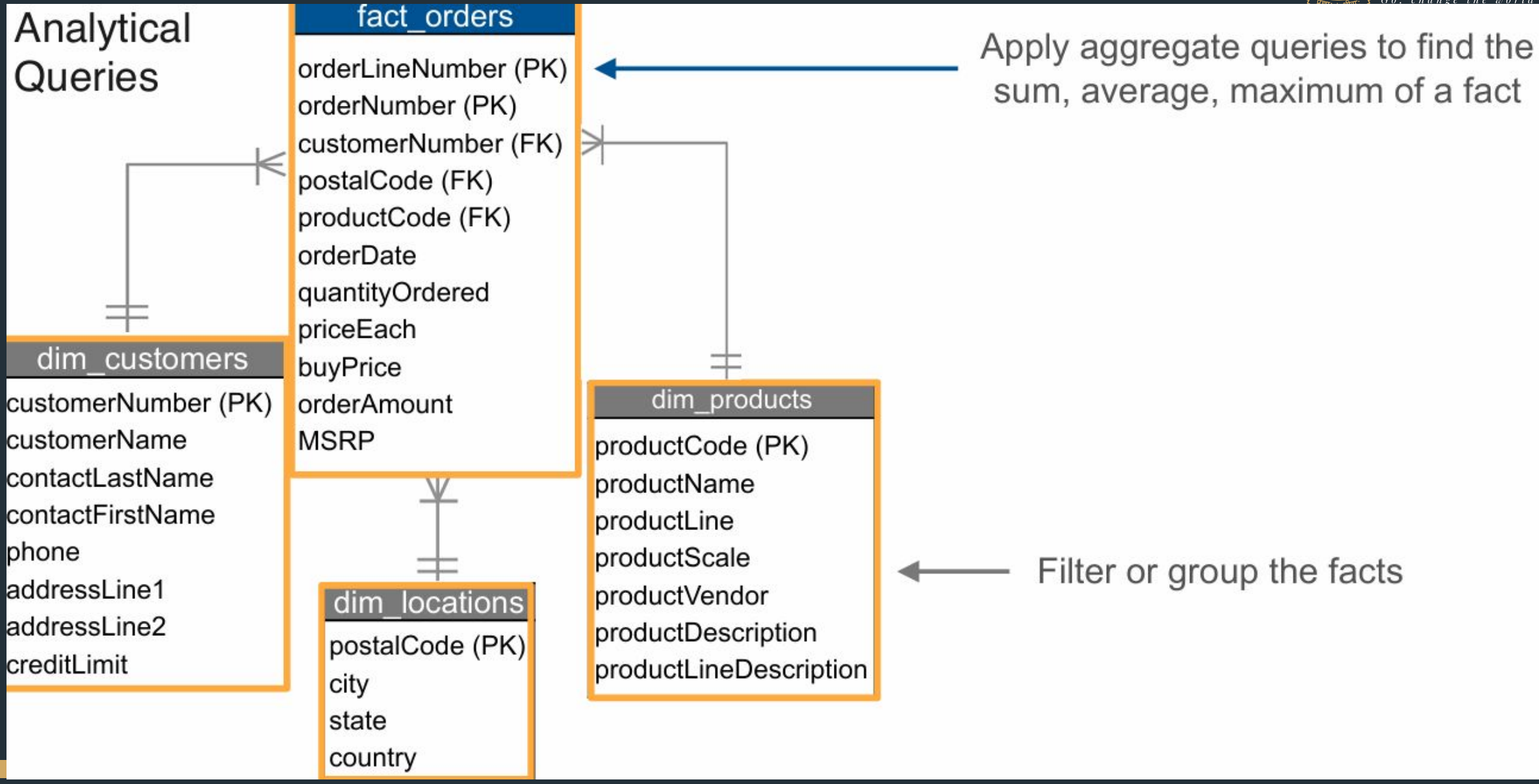
Star Schema Example



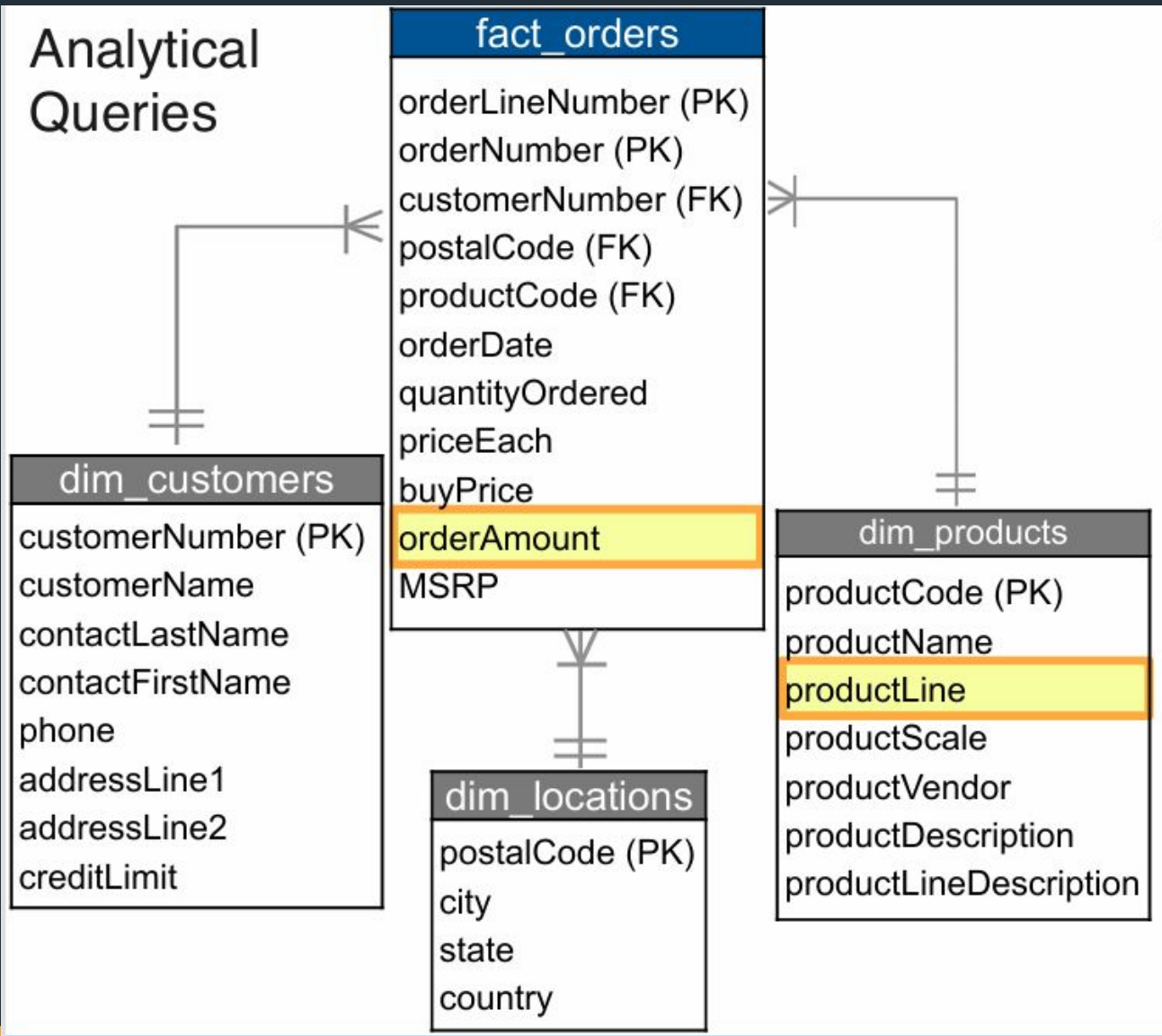
Star Schema Example



Analytical Queries



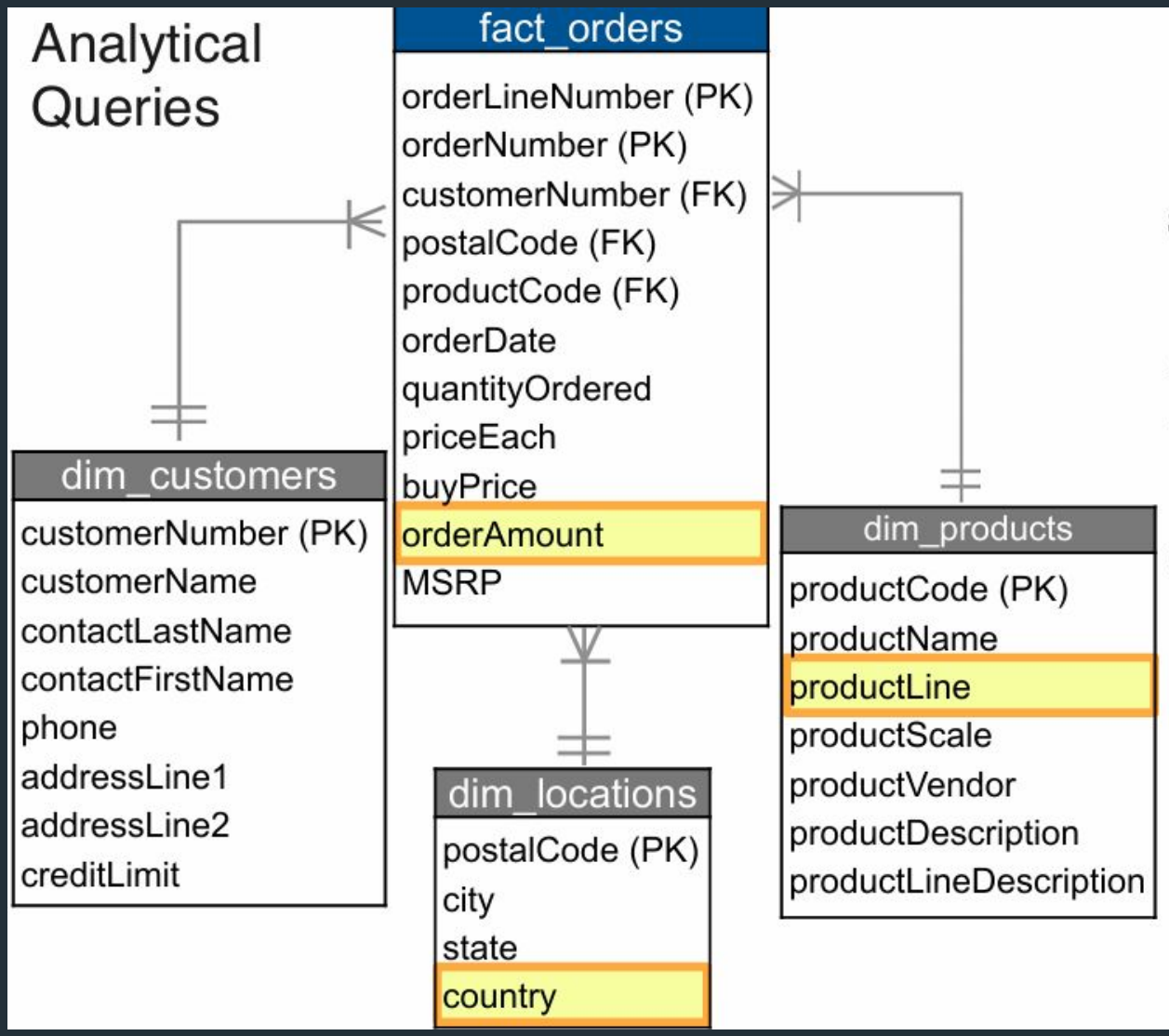
Analytical Queries



Find **the total sales amount** for each product line within the USA

```
SELECT
  SUM(fact_orders.orderAmount) AS total_sales
FROM fact_orders
JOIN dim_products ON
  fact_orders.productCode =
  dim_products.productCode
```

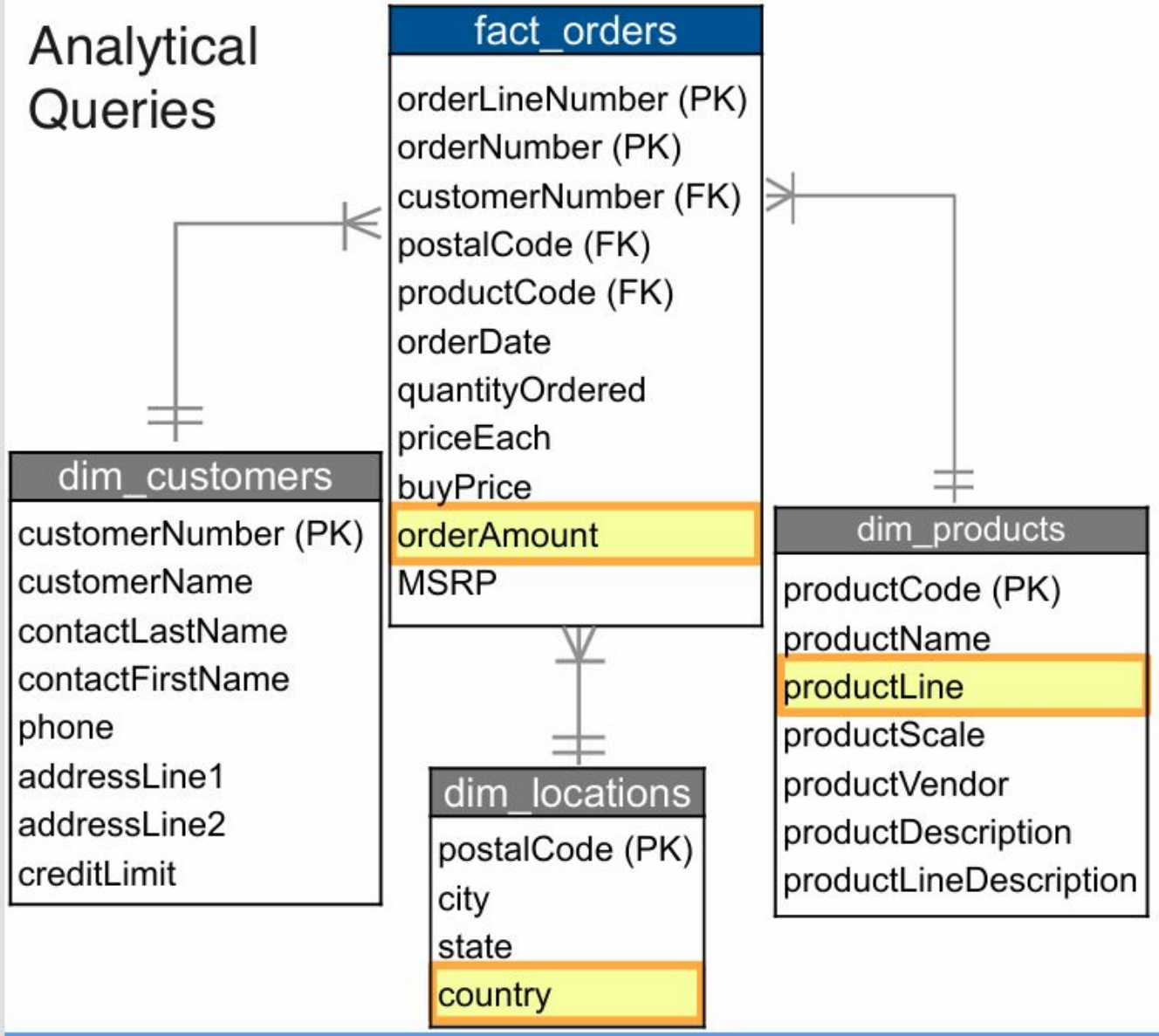
Analytical Queries



Find **the total sales amount** for each product line within the USA

```
SELECT
  dim_products.productLine,
  SUM(fact_orders.orderAmount) AS total_sales
FROM fact_orders
JOIN dim_products ON
  fact_orders.productCode =
  dim_products.productCode
GROUP BY dim_products.productLine
```

Analytical Queries

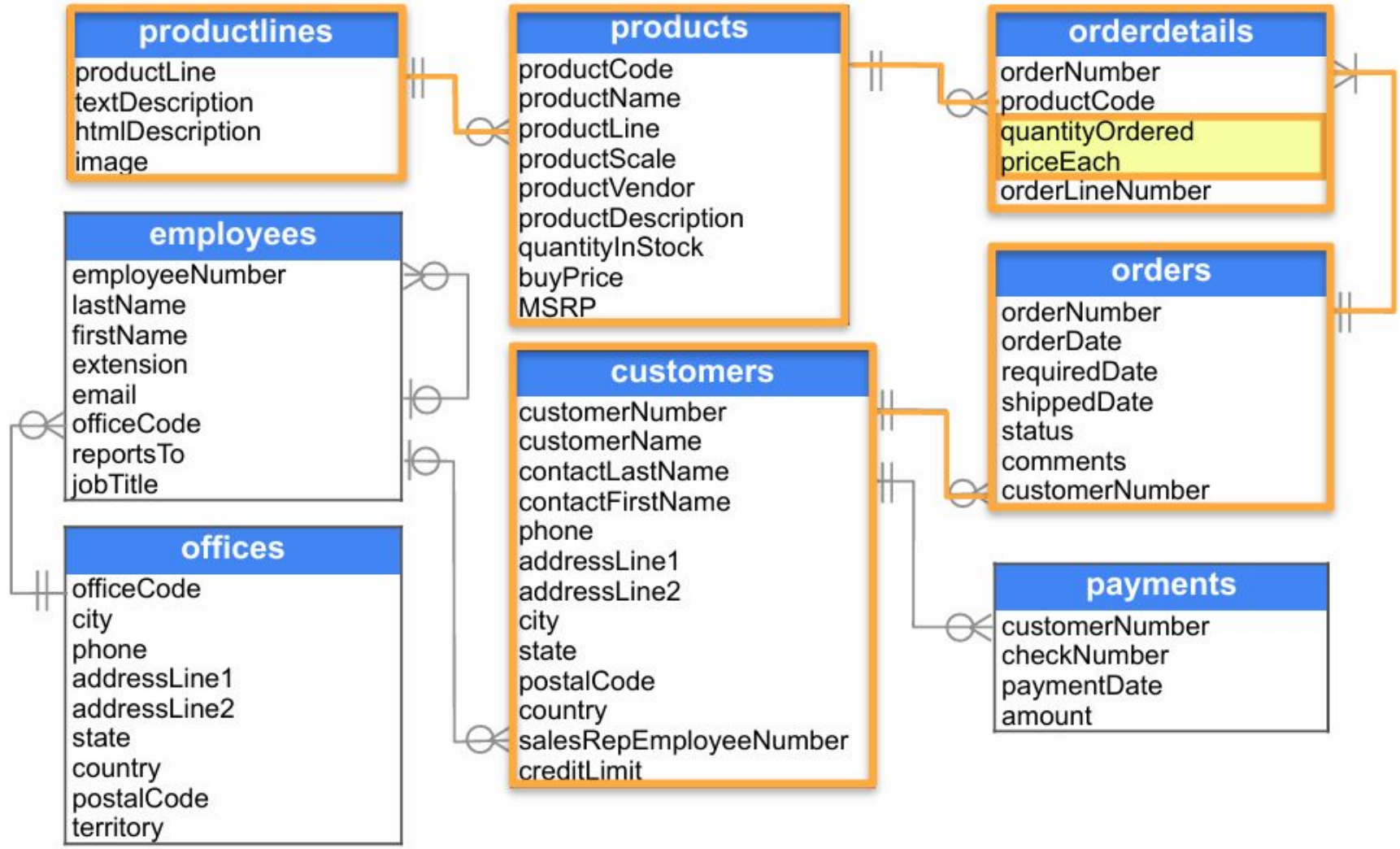


Find **the total sales amount** for each product line within the USA

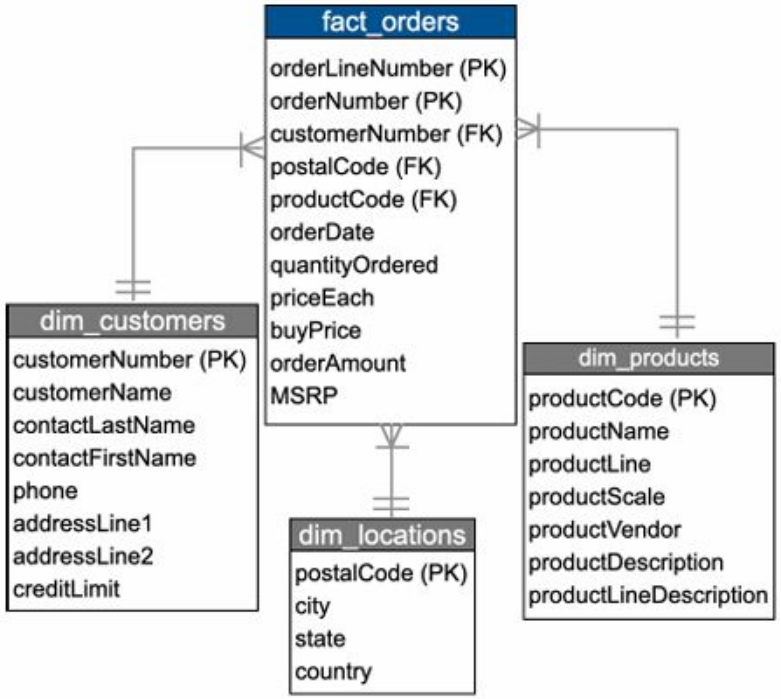
```
SELECT
  dim_products.productLine,
  SUM(fact_orders.orderAmount) AS total_sales
FROM fact_orders
JOIN dim_products ON
  fact_orders.productCode =
  dim_products.productCode
JOIN dim_locations ON
  fact_orders.postalCode =
  dim_locations.postalCode
WHERE dim_locations.country = 'USA'
GROUP BY dim_products.productLine
```

Star Schema VS 3NF

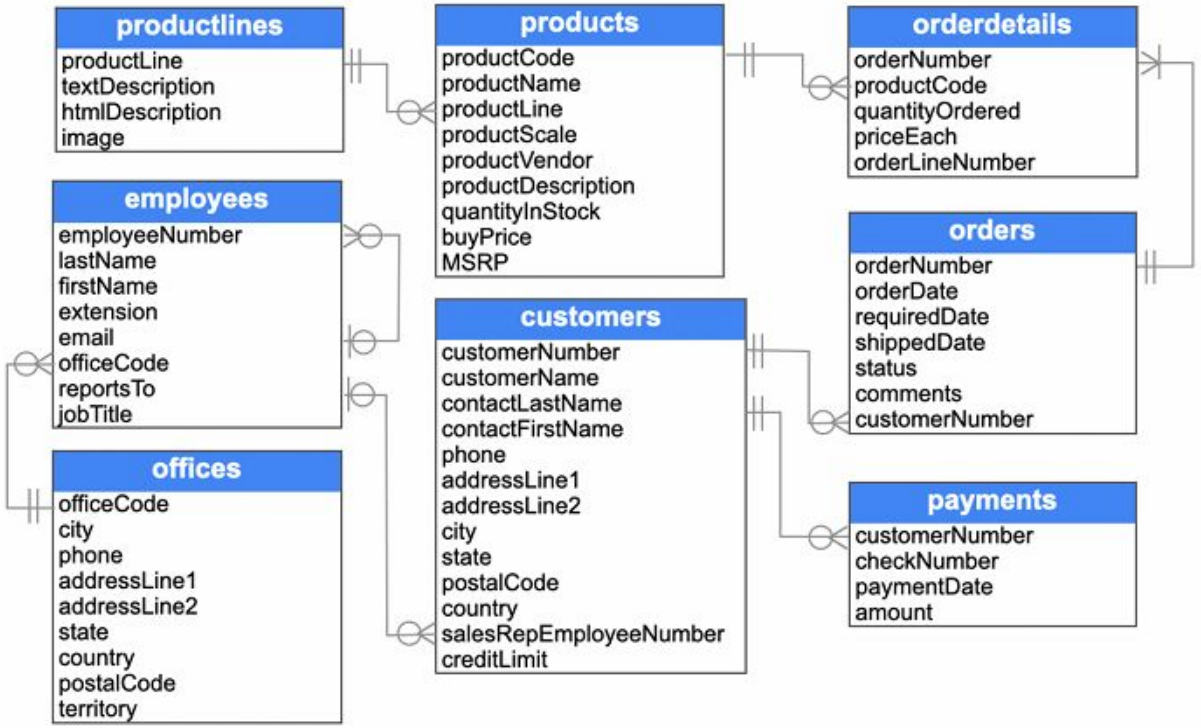
Find **the total sales amount** for each product line within the USA



Star Schema



Normalized Model (3NF)



- Star Schema organizes data so it's easier for business users to understand, navigate and use.
- Star Schema results in simpler queries with fewer joins.

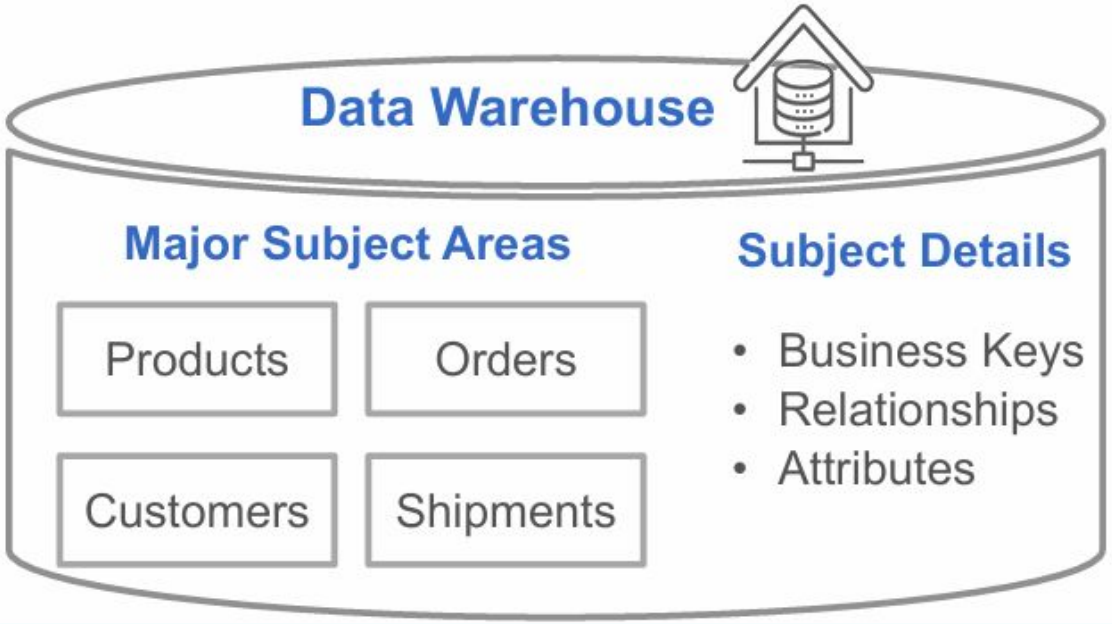
Data Modeling Techniques

Inmon vs Kimball Data Warehouse Architecture

Inmon Data Modeling Approach

A **subject-oriented**, **integrated**, **nonvolatile**, and **time-variant** collection of data in support of management's decisions.

The data warehouse contains **granular** corporate data. Data in the data warehouse is able to be used for many different purposes, including sitting and waiting for future requirements which are unknown today.



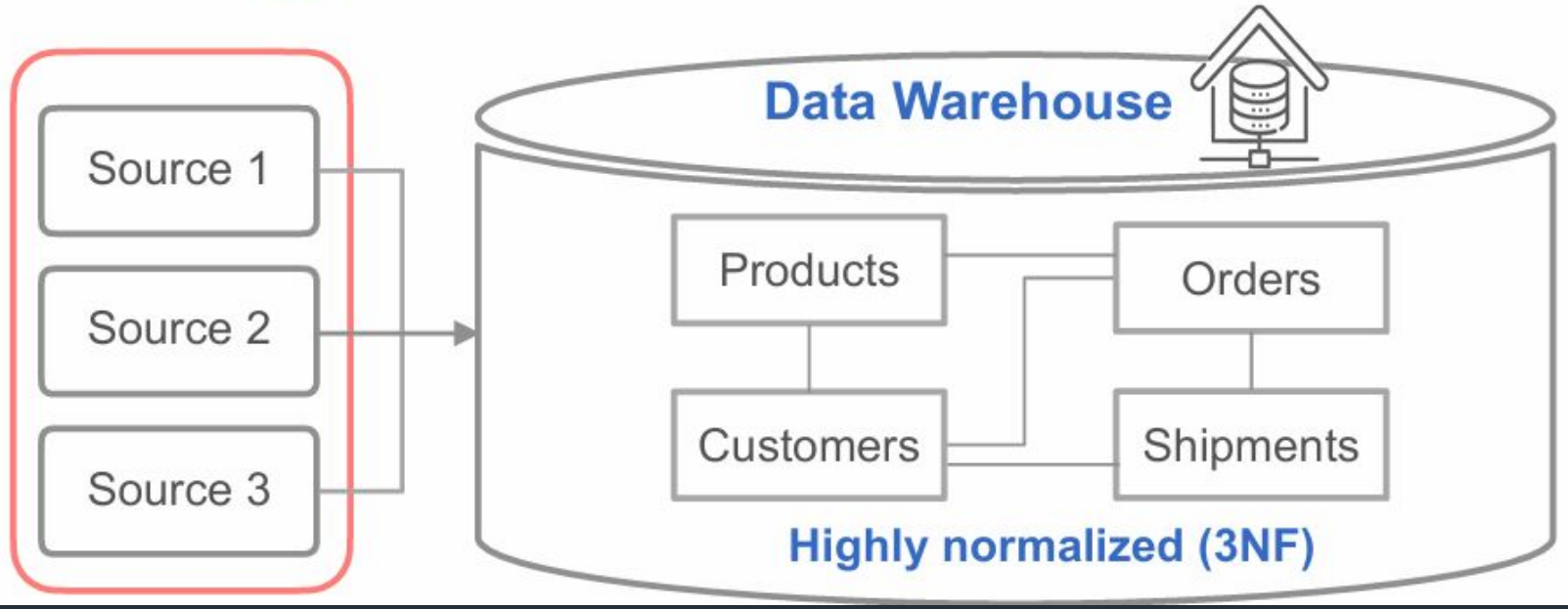
Inmon Data Modeling Approach

A **subject-oriented**, **integrated**, **nonvolatile**, and **time-variant** collection of data in support of management's decisions.

The data warehouse contains **granular** corporate data. Data in the data warehouse is able to be used for many different purposes, including sitting and waiting for future requirements which are unknown today.

A **subject-oriented**, **integrated**, **nonvolatile**, and **time-variant** collection of data in support of management's decisions.

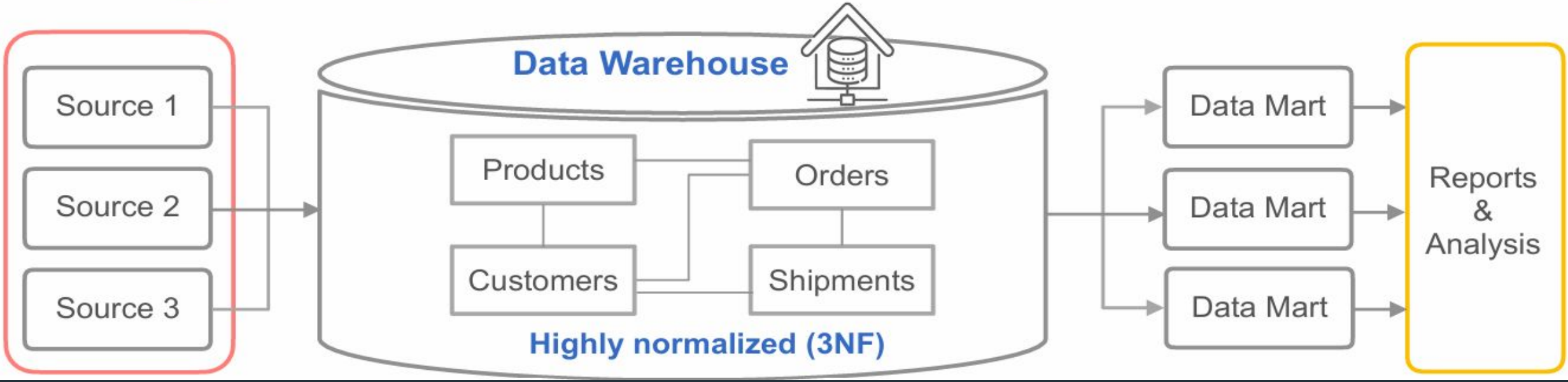
The data warehouse contains **granular** corporate data. Data in the data warehouse is able to be used for many different purposes, including sitting and waiting for future requirements which are unknown today.



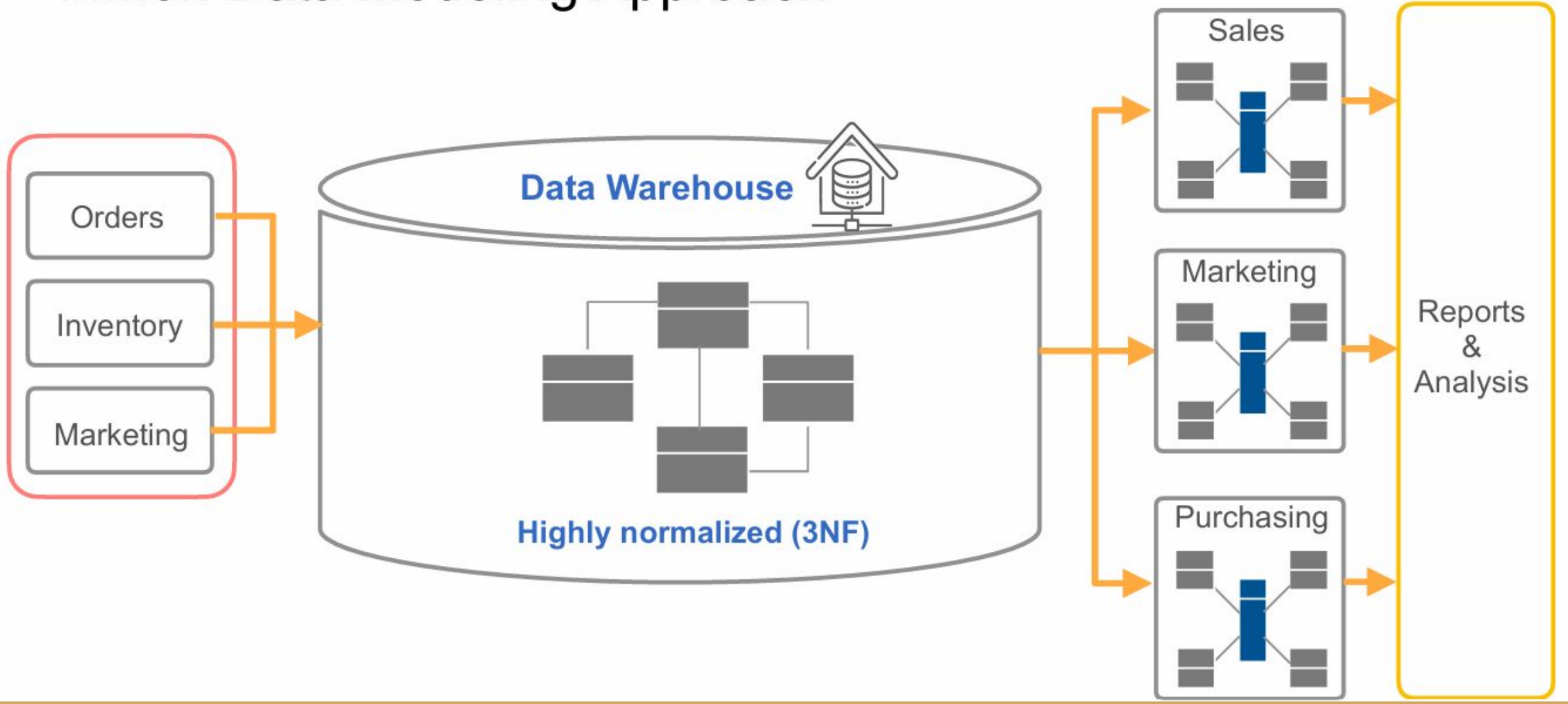
Inmon Data Modeling Approach

A **subject-oriented**, **integrated**, **nonvolatile**, and **time-variant** collection of data in support of management's decisions.

The data warehouse contains **granular** corporate data. Data in the data warehouse is able to be used for many different purposes, including sitting and waiting for future requirements which are unknown today.



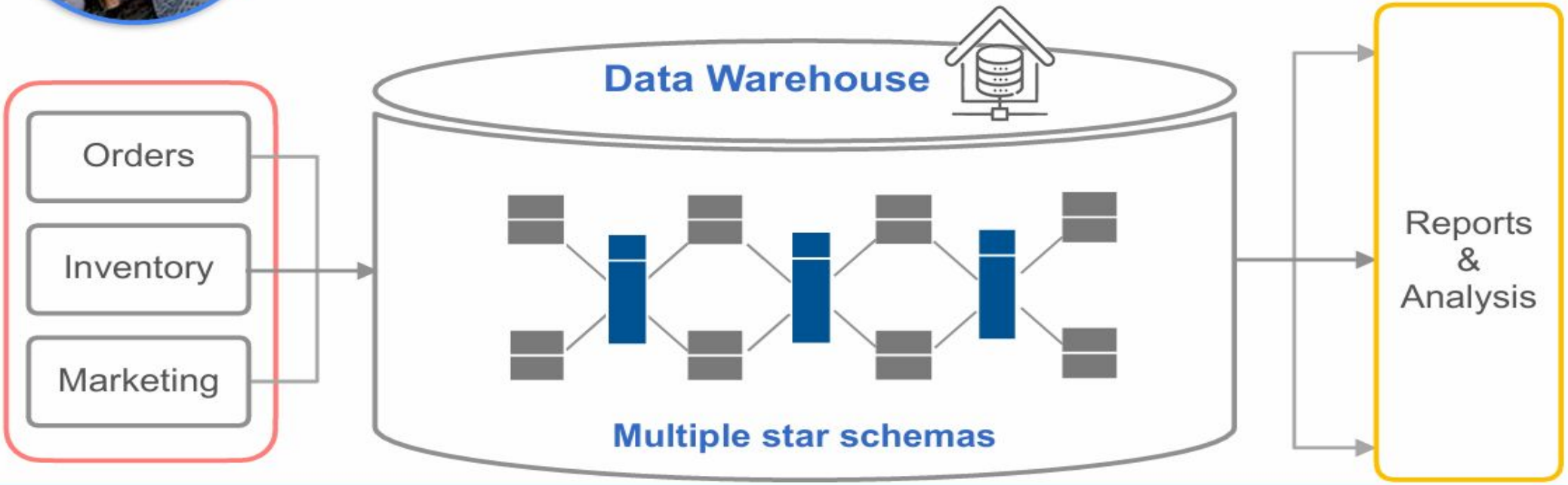
Inmon Data Modeling Approach



Kimball Data Modeling Approach

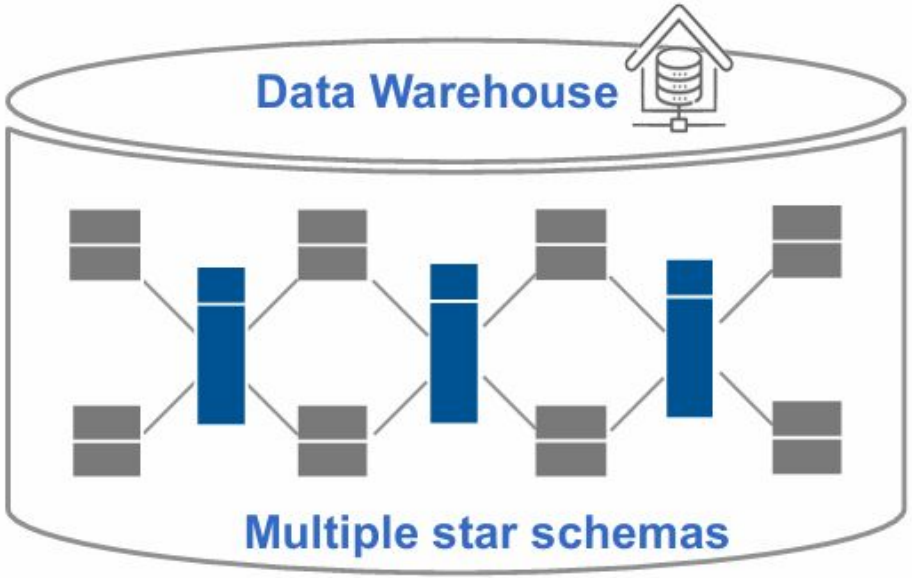
Kimball's approach effectively allows you to serve data that's structured as star schemas (or similar variants) directly from the data warehouse.

- Faster modeling and iteration
- More data redundancy and duplications



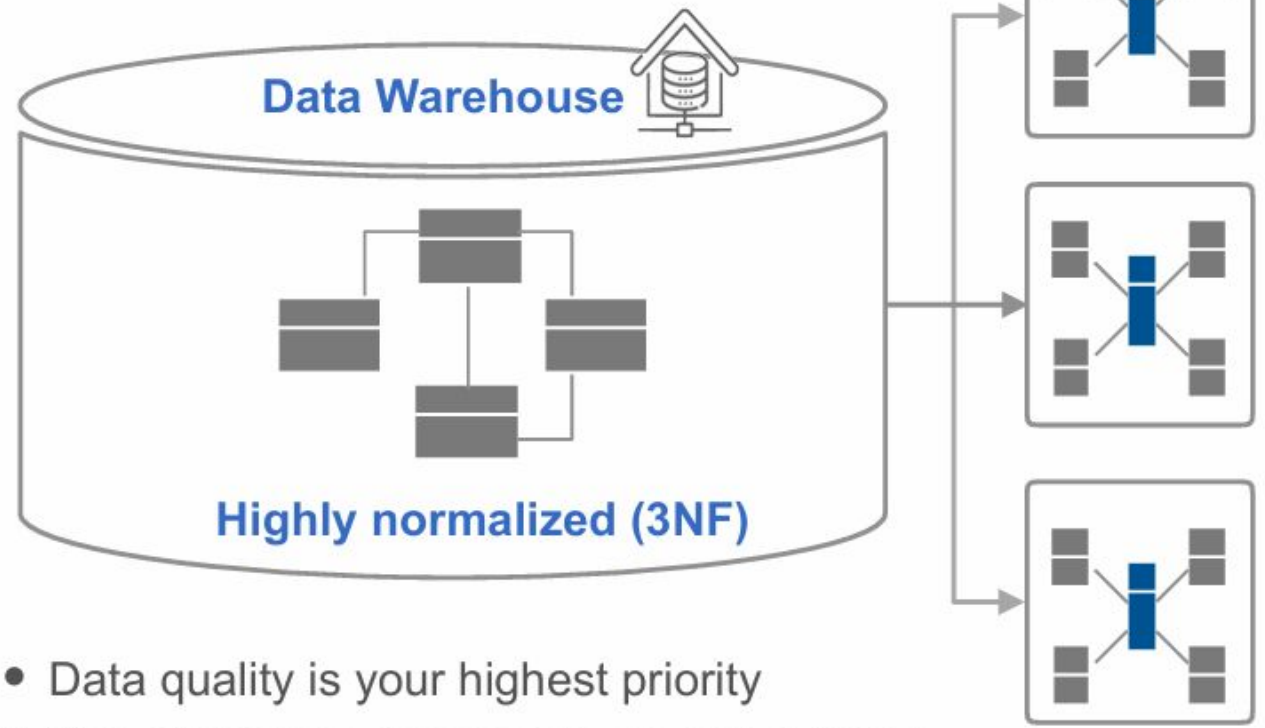
When to Choose Each Approach?

Kimball's Modeling Approach



- Quick insights are your highest priority
- Rapid implementation and iteration

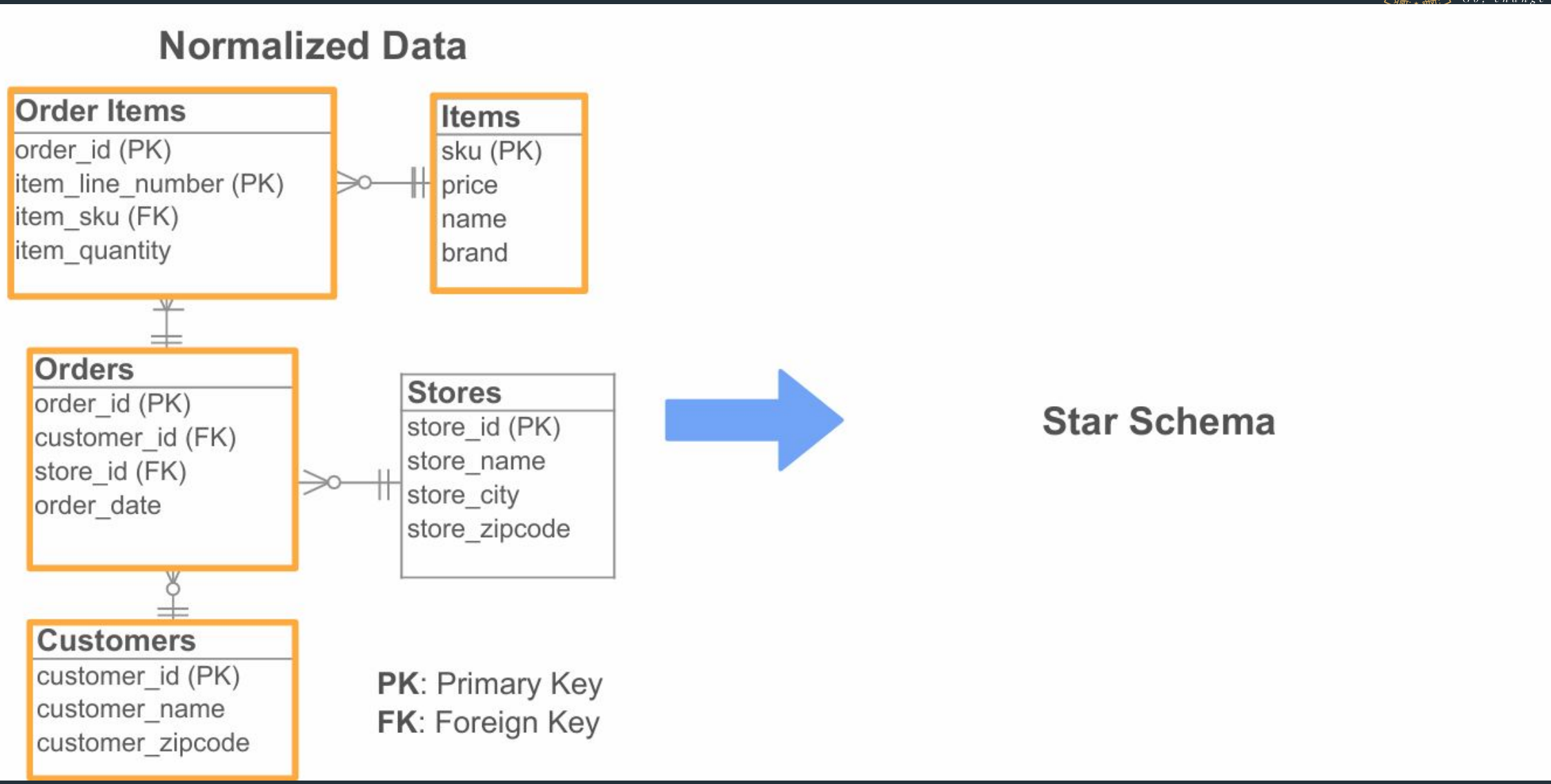
Inmon's Data Warehouse



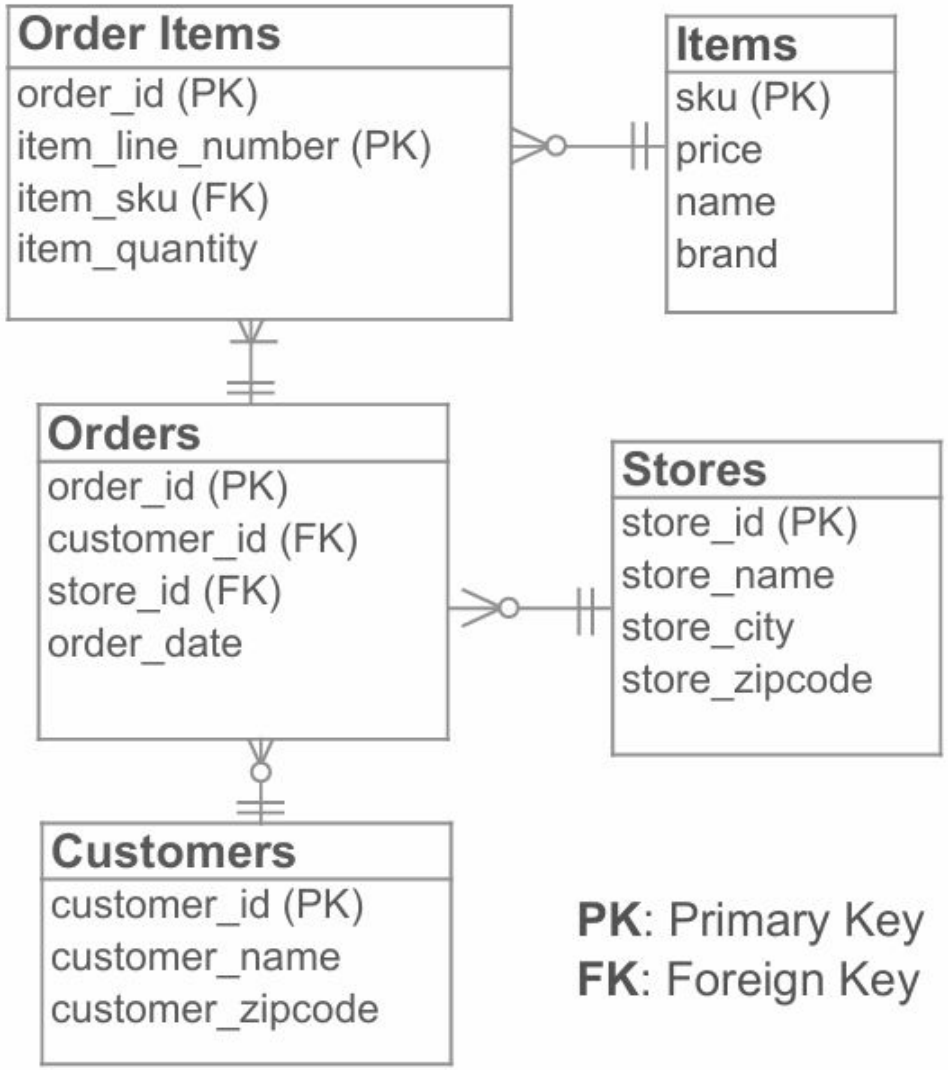
- Data quality is your highest priority
- The analysis requirements are not defined

Data Modeling Techniques

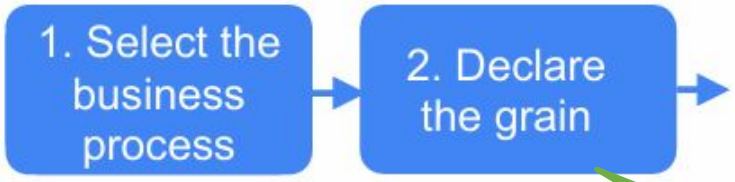
Exercise: From Normalization to Star Schema


Normalized Data
Order Items
order_id (PK)
item_line_number (PK)
item_sku (FK)
item_quantity
Items
sku (PK)
price
name
brand
Orders
order_id (PK)
customer_id (FK)
store_id (FK)
order_date
Stores
store_id (PK)
store_name
store_city
store_zipcode
Customers
customer_id (PK)
customer_name
customer_zipcode
PK: Primary Key
FK: Foreign Key
Star Schema

Normalized Data



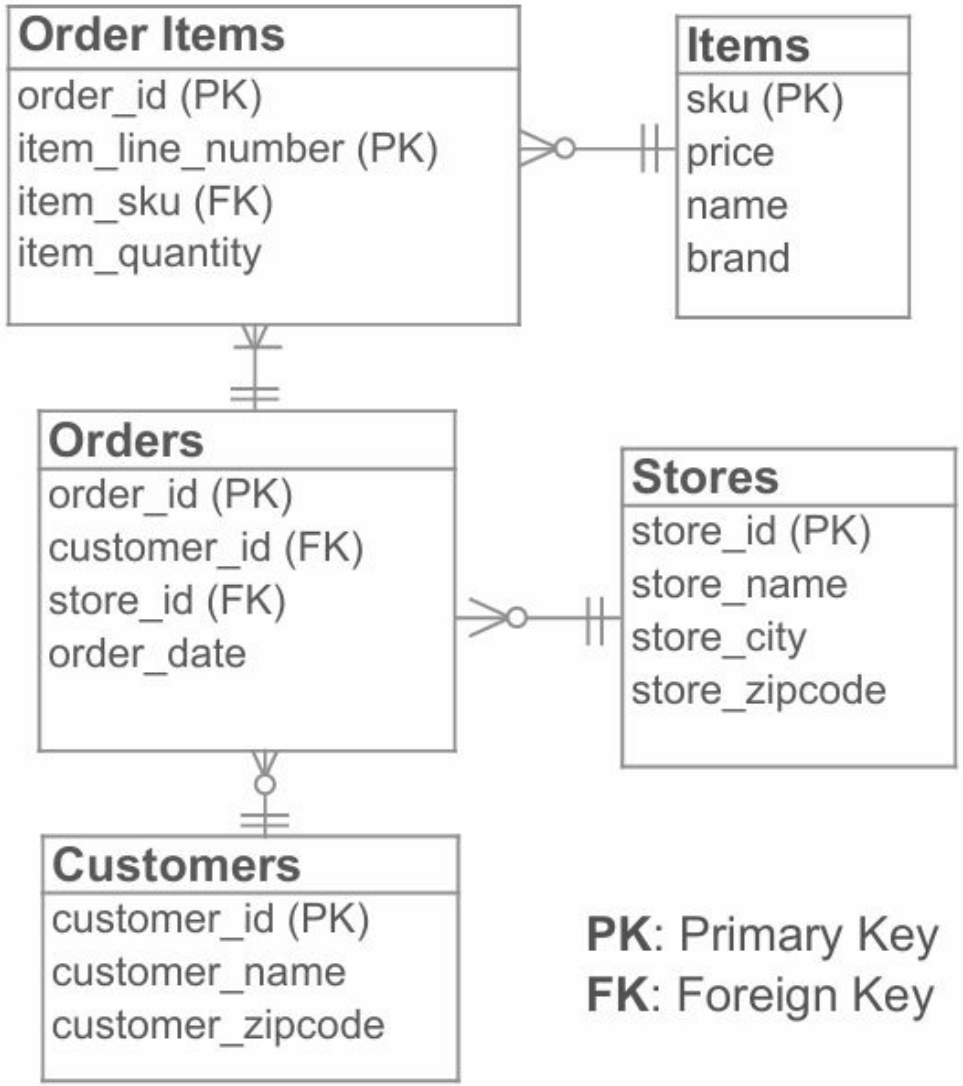
Star Schema



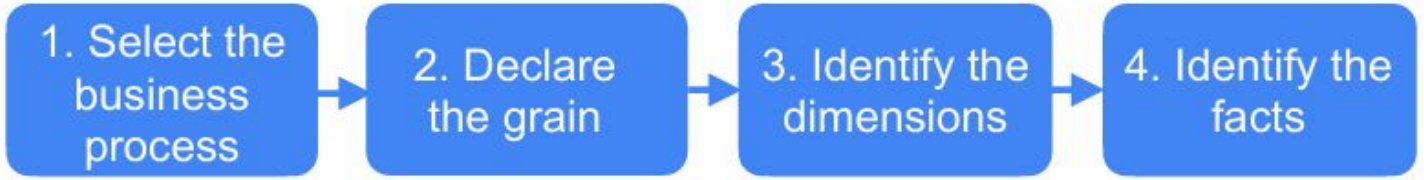
Understand the needs of the business

Grain defines the "atomic level" of the fact table, telling you what each row represents. It's the foundation for how facts and dimensions relate.

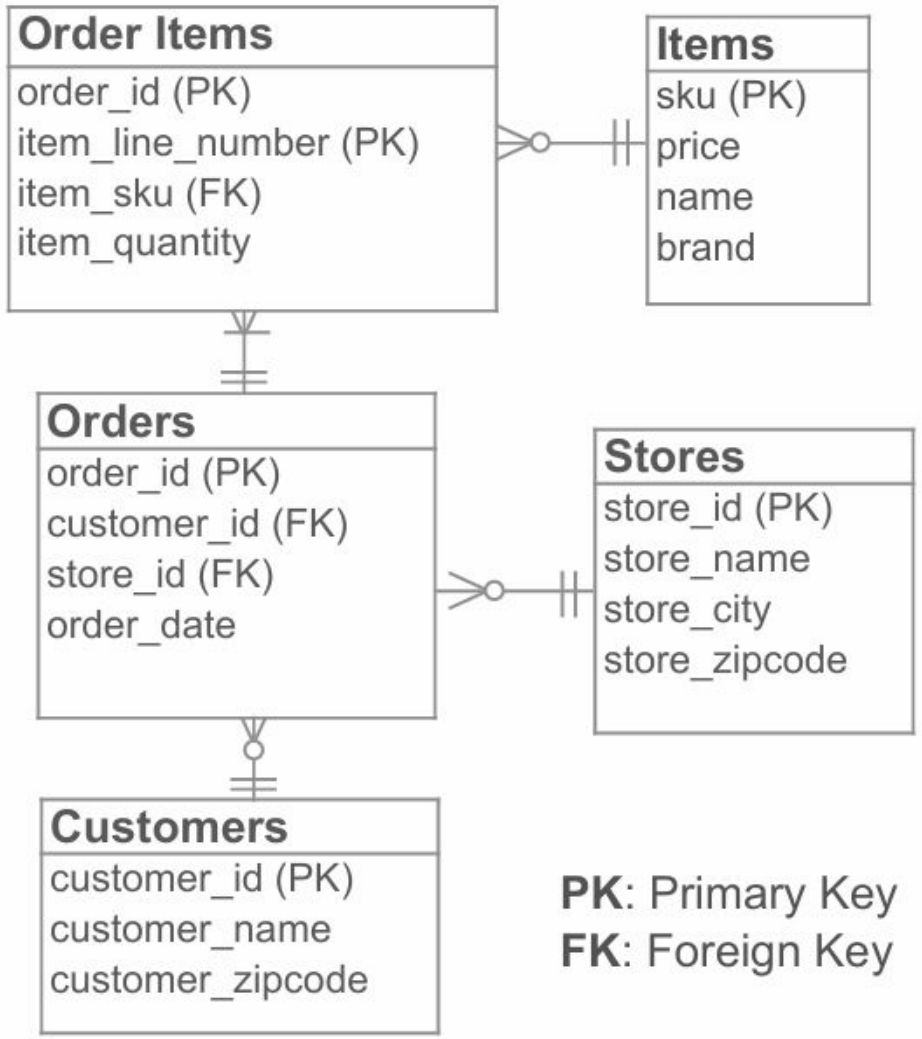
Normalized Data



Star Schema



Normalized Data



PK: Primary Key
FK: Foreign Key

Star Schema

1. Select the business process → 2. Declare the grain → 3. Identify the dimensions → 4. Identify the facts

Analyze the sales data:

- which products are selling in which stores on a given day
- differences in the sales between the stores
- which product brands are most popular

Business process: company's sales transactions

Atomic Grain:

- Total sales transactions on a particular day
- Single sales transaction
- Individual product item in a sales transaction

Normalized Data

Order Items
order_id (PK)
item_line_number (PK)
item_sku (FK)
item_quantity

Items
sku (PK)
price
name
brand

Orders
order_id (PK)
customer_id (FK)
store_id (FK)
order_date

Stores
store_id (PK)
store_name
store_city
store_zipcode

Customers
customer_id (PK)
customer_name
customer_zipcode

PK: Primary Key
FK: Foreign Key

Star Schema

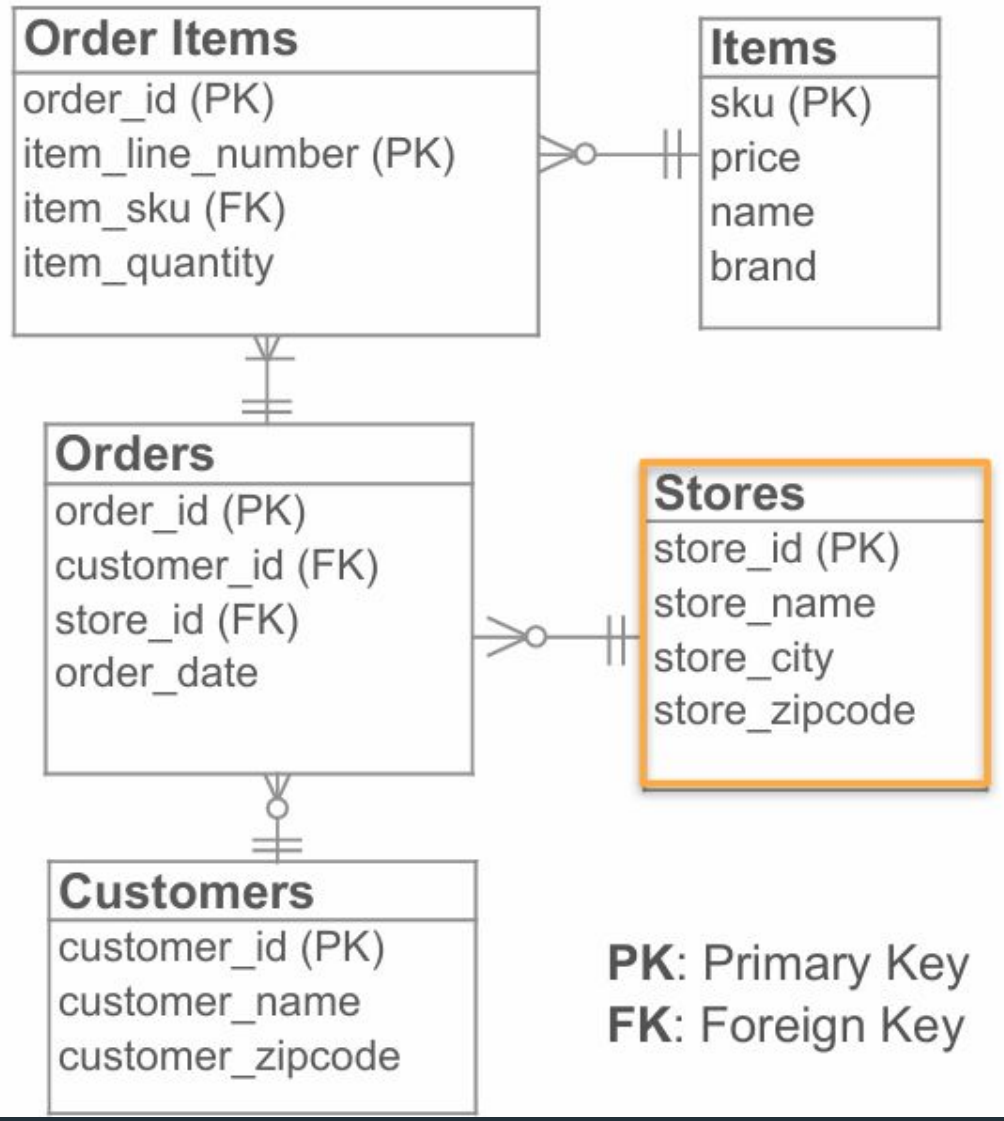
Bus. process: sales transactions → Grain: individual product in a transaction → 3. Identify the dimensions → 4. Identify the facts

Analyze the sales data:

- which products are selling in which stores on a given day
- differences in the sales between the stores
- which product brands are most popular

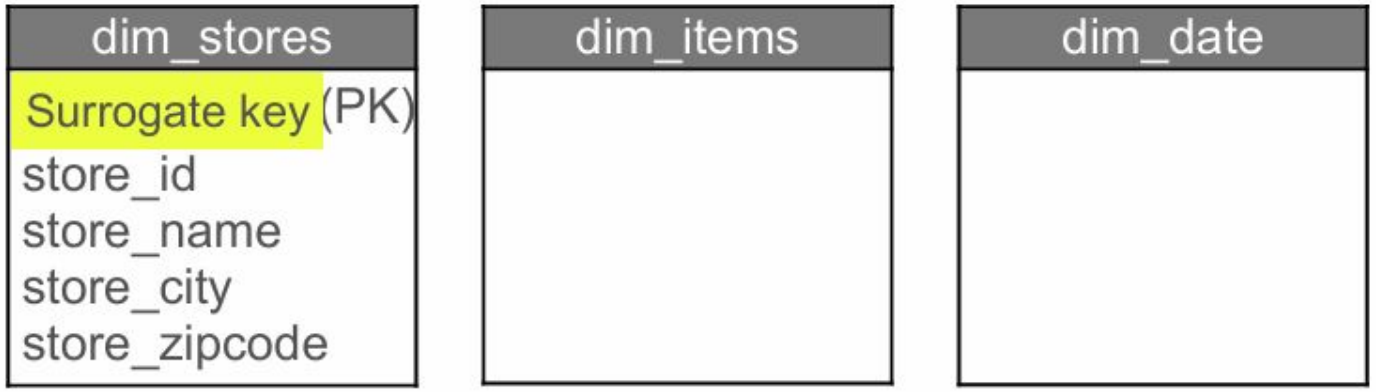
dim_stores	dim_items	dim_date

Normalized Data



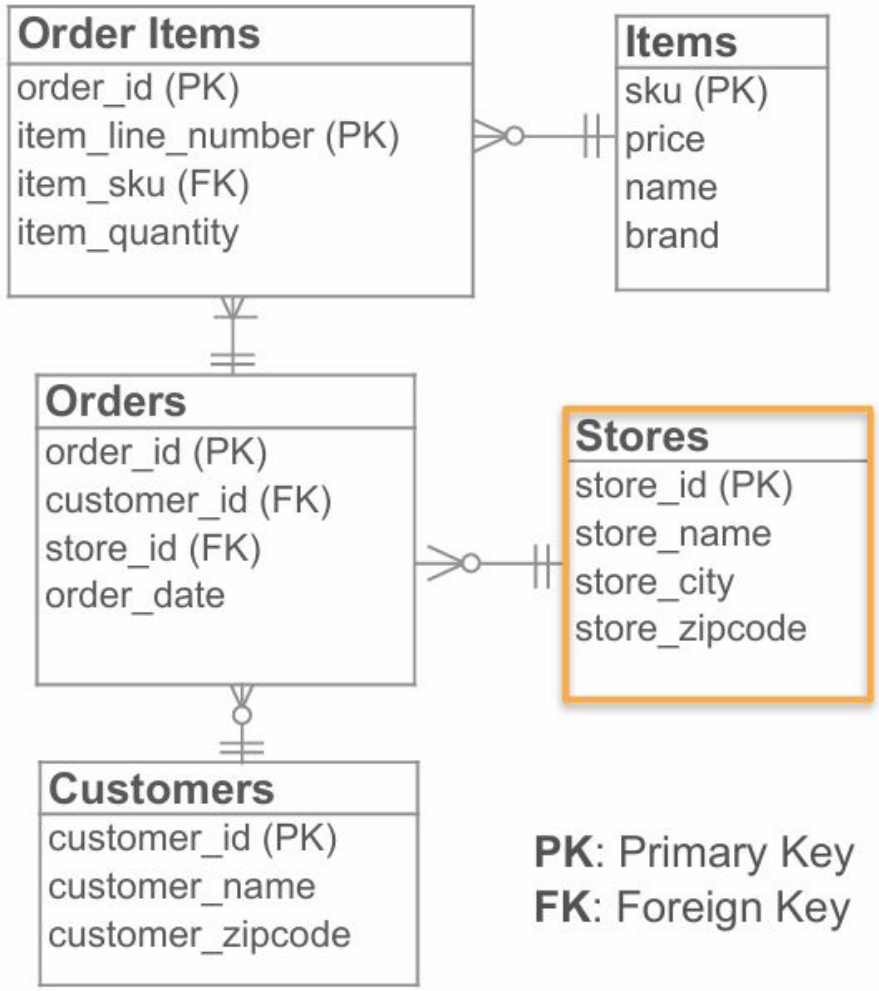
Star Schema

Bus. process: sales transactions → Grain: individual product in a transaction → 3. Identify the dimensions → 4. Identify the facts



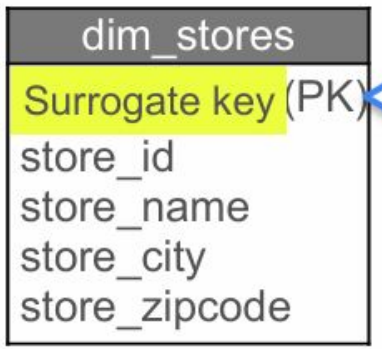

```
SELECT store_id,
       store_name,
       store_city,
       store_zipcode

FROM stores;
```

Normalized Data



Star Schema

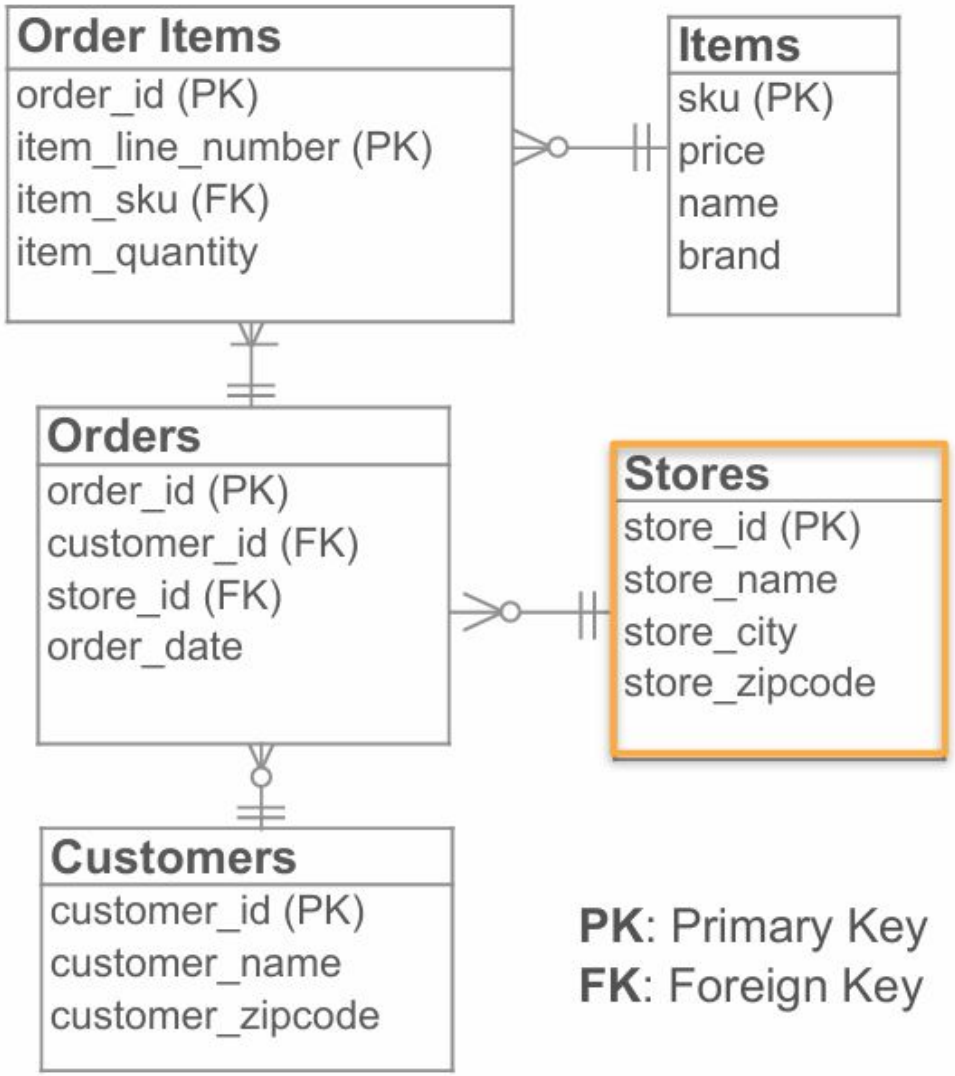
Bus. process: sales transactions → Grain: individual product in a transaction → 3. Identify the dimensions → 4. Identify the facts



- Create a sequence of integers and assign one integer to each store
- Use a hash function to generate a unique surrogate key
  - Supported by many DBMSs like PostgreSQL and MySQL
  - Example: MD5

```
SELECT store_id,
       store_name,
       store_city,
       store_zipcode

FROM stores;
```

Normalized Data



Star Schema

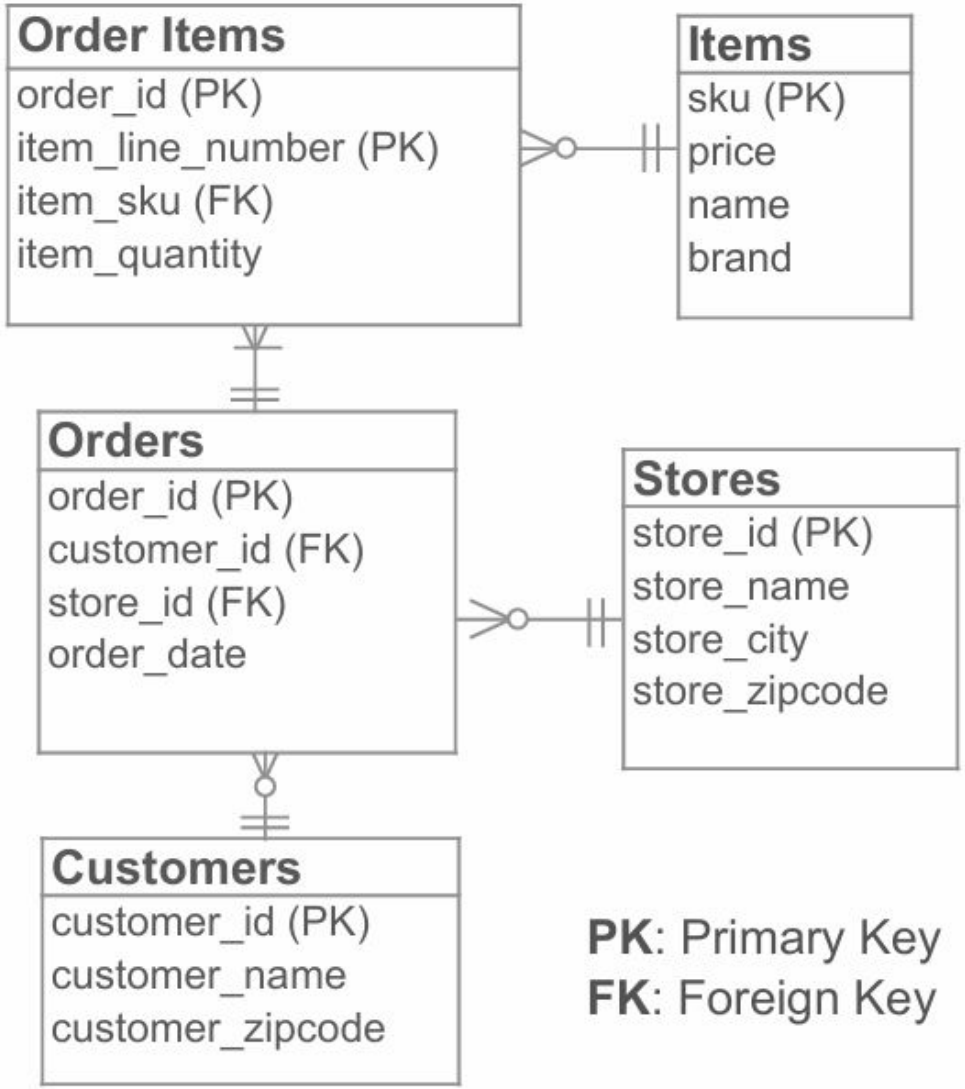
Bus. process: sales transactions → Grain: individual product in a transaction → 3. Identify the dimensions → 4. Identify the facts

dim_stores	
store_key	(PK)
store_id	
store_name	
store_city	
store_zipcode	

- Create a sequence of integers and assign one integer to each store
- Use a hash function to generate a unique surrogate key
  - Supported by many DBMSs like PostgreSQL and MySQL
  - Example: MD5

```
SELECT MD5(store_id) as store_key,
       store_id,
       store_name,
       store_city,
       store_zipcode

FROM stores;
```

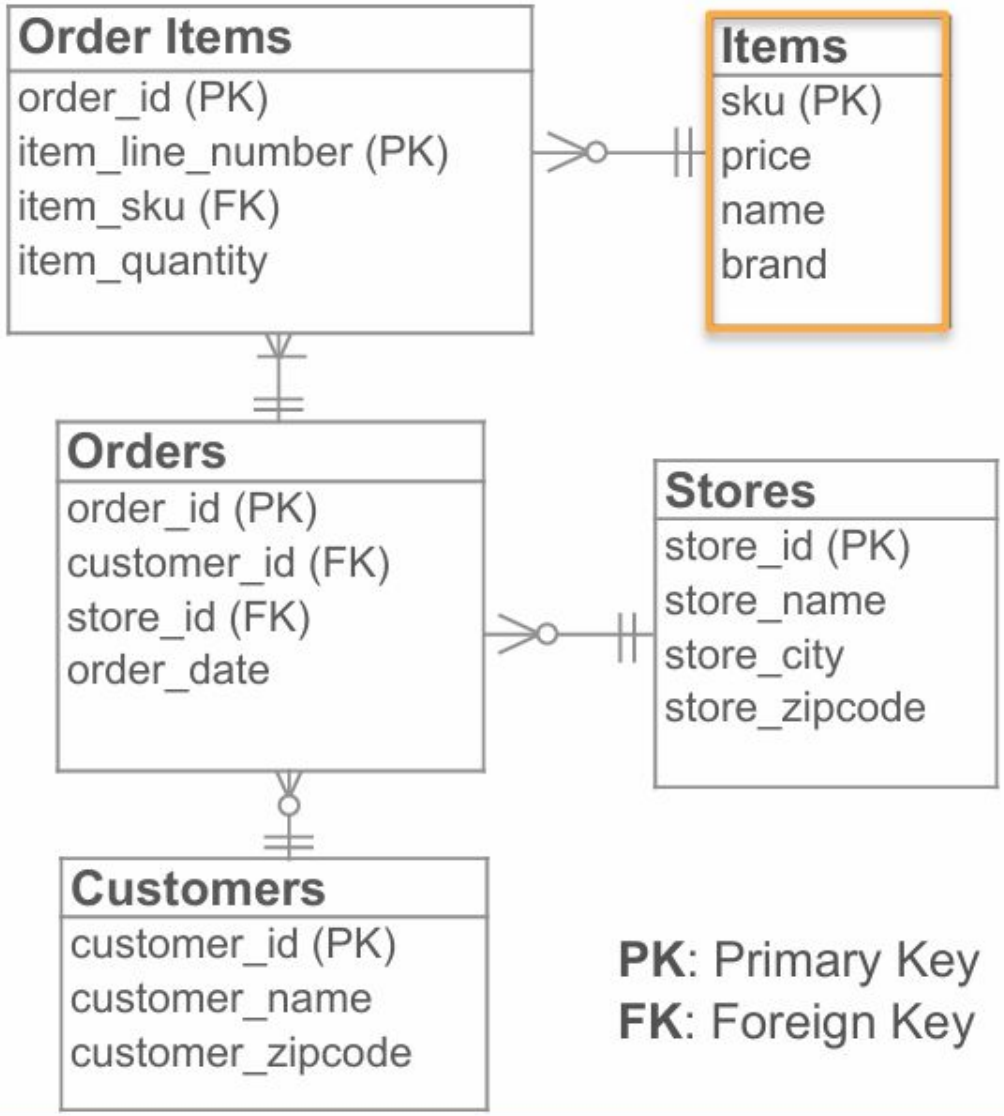
Normalized Data



Star Schema


```
SELECT MD5(store_id) as store_key,
       store_id,
       store_name,
       store_city,
       store_zipcode

FROM stores;
```

Normalized Data

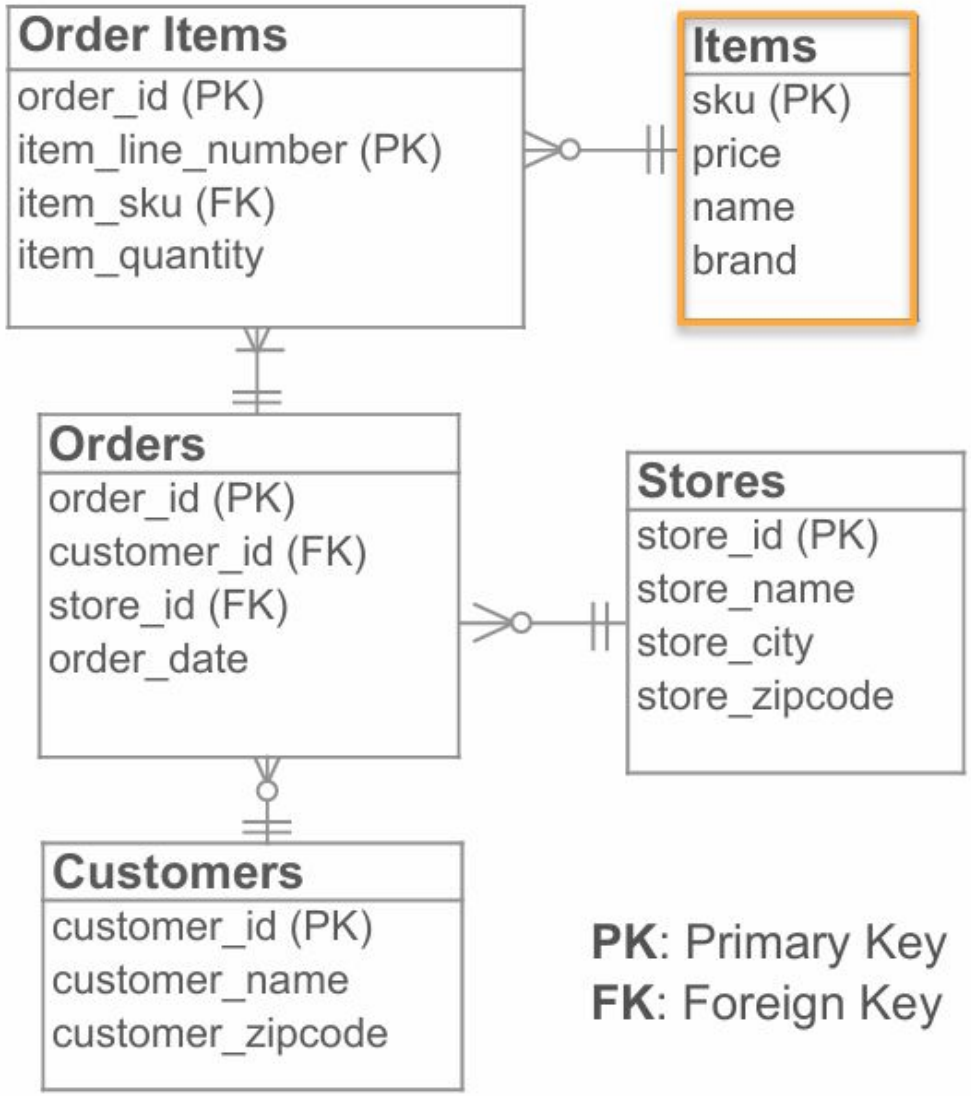


Star Schema

Bus. process: sales transactions → Grain: individual product in a transaction → 3. Identify the dimensions → 4. Identify the facts


```
SELECT sku,
       name,
       brand
FROM items;
```

Normalized Data

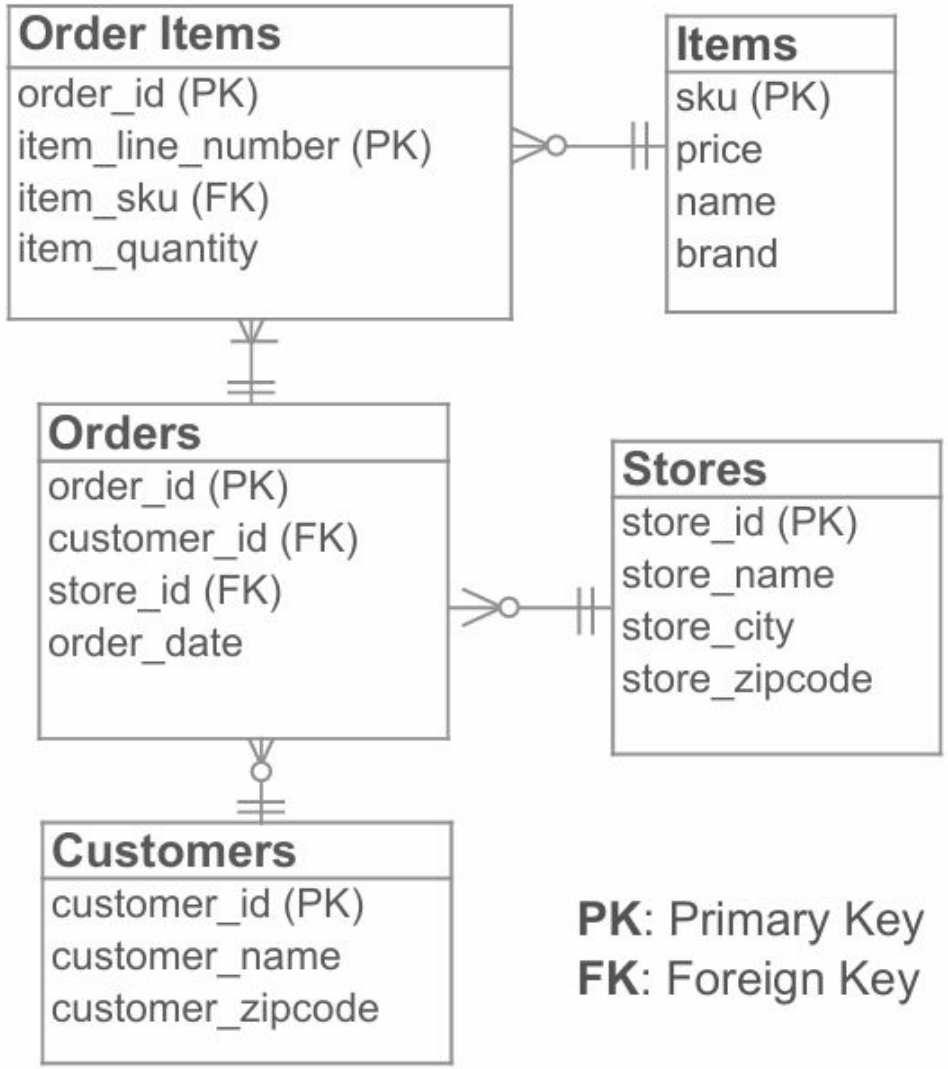


Star Schema

Bus. process: sales transactions → Grain: individual product in a transaction → 3. Identify the dimensions → 4. Identify the facts

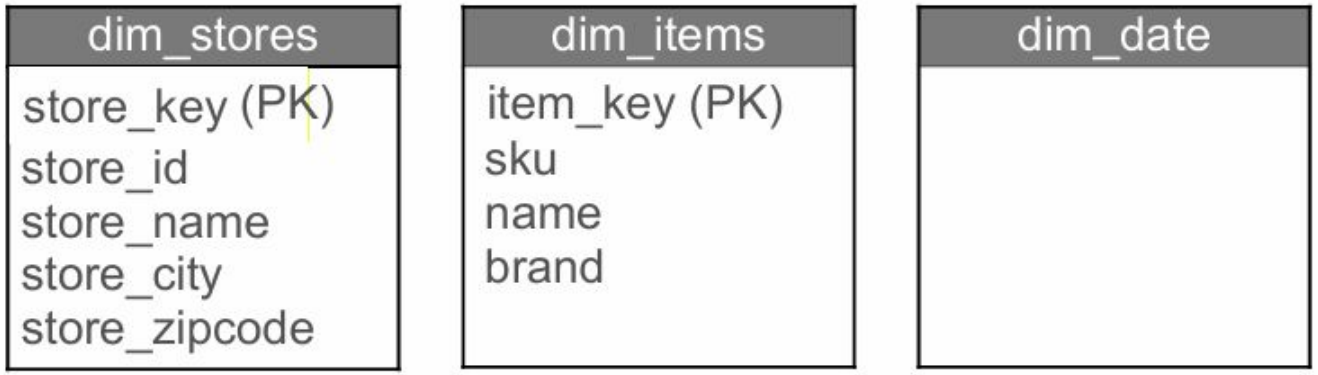

```
SELECT MD5(sku) as item_key,
       sku,
       name,
       brand
FROM items;
```

Normalized Data



Star Schema

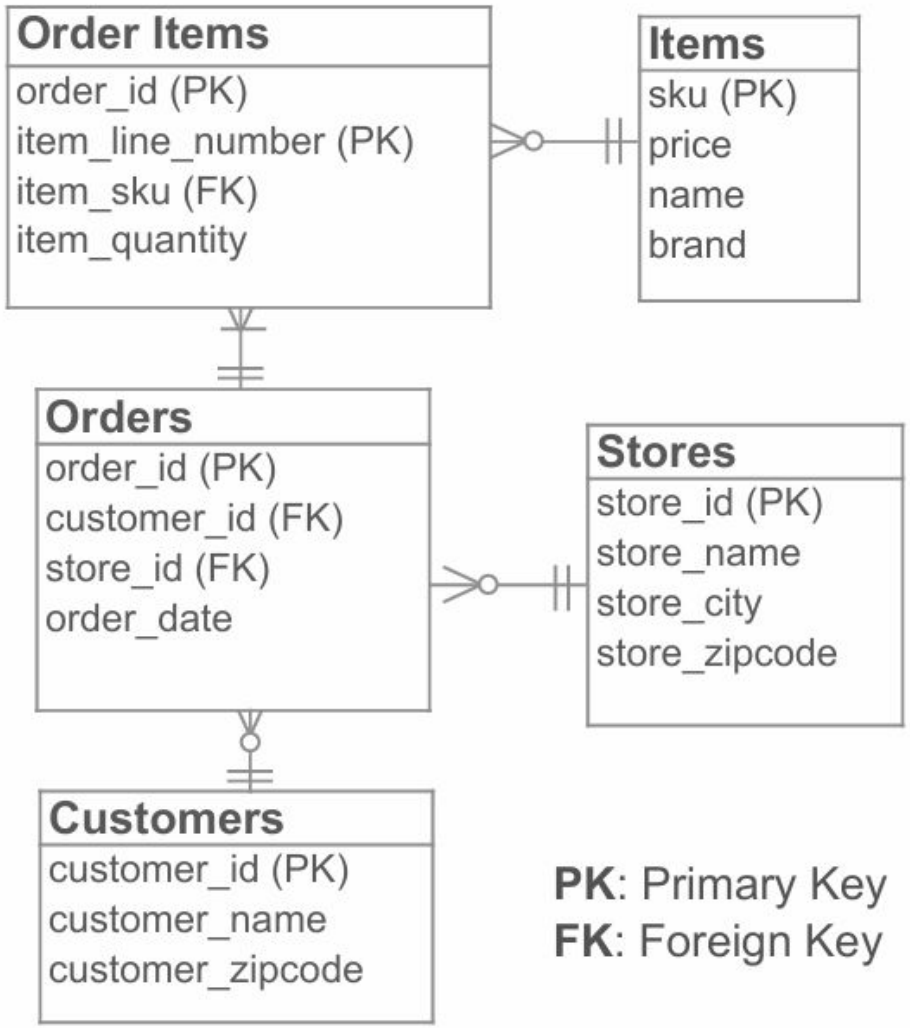
Bus. process: sales transactions → Grain: individual product in a transaction → 3. Identify the dimensions → 4. Identify the facts



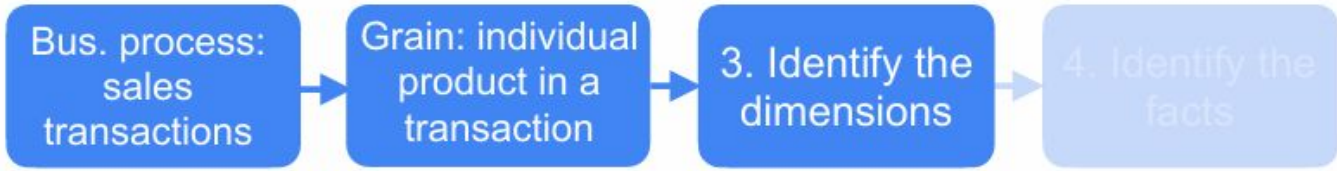
Date	Year	Quarter	Month	Day-of-week
2022-03-01	2022	1	3	Tuesday
2022-03-02	2022	1	3	Wednesday

- What are the total sales in the first quarter of 2022?
- What products are more popular on the weekends?

Normalized Data



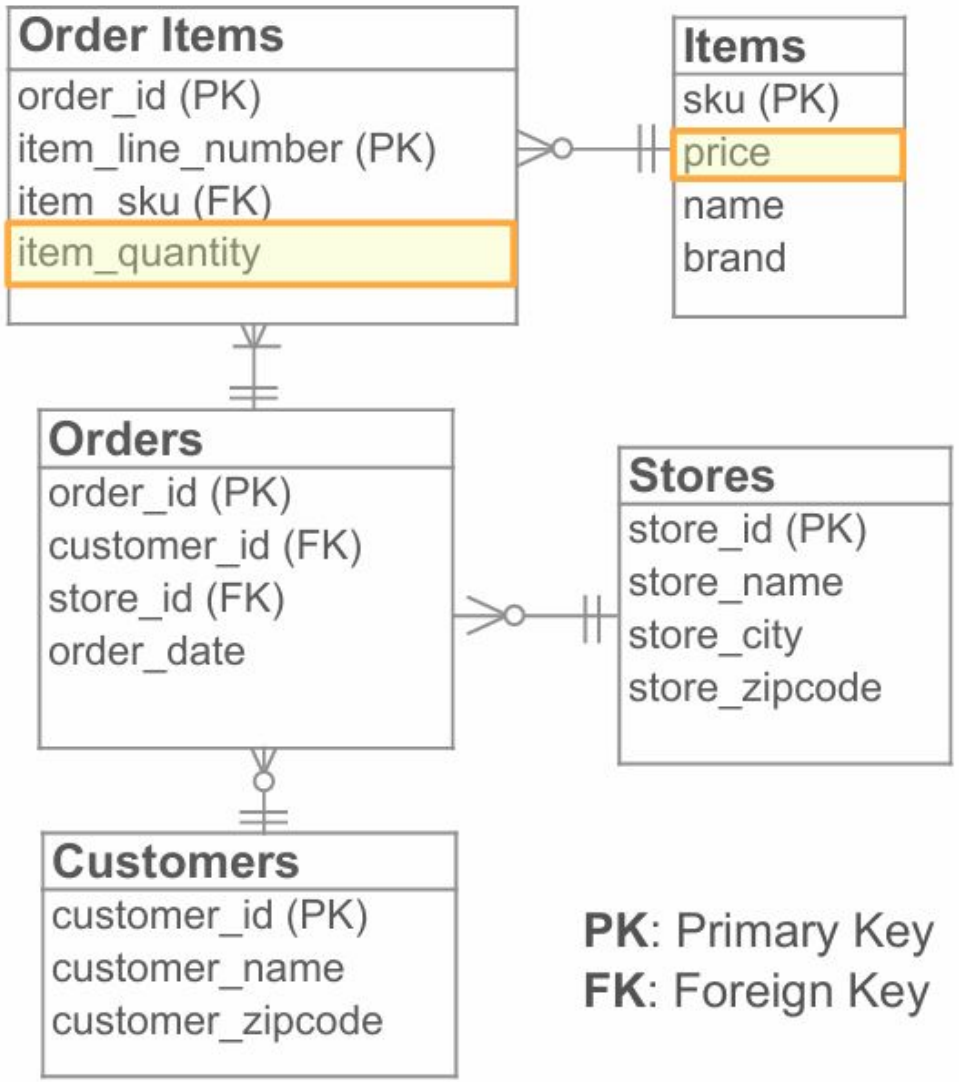
Star Schema



dim_stores	dim_items	dim_date
store_key (PK)	item_key (PK)	date_key(PK)
store_id	sku	day_of_week
store_name	name	month
store_city	brand	quarter
store_zipcode		Year

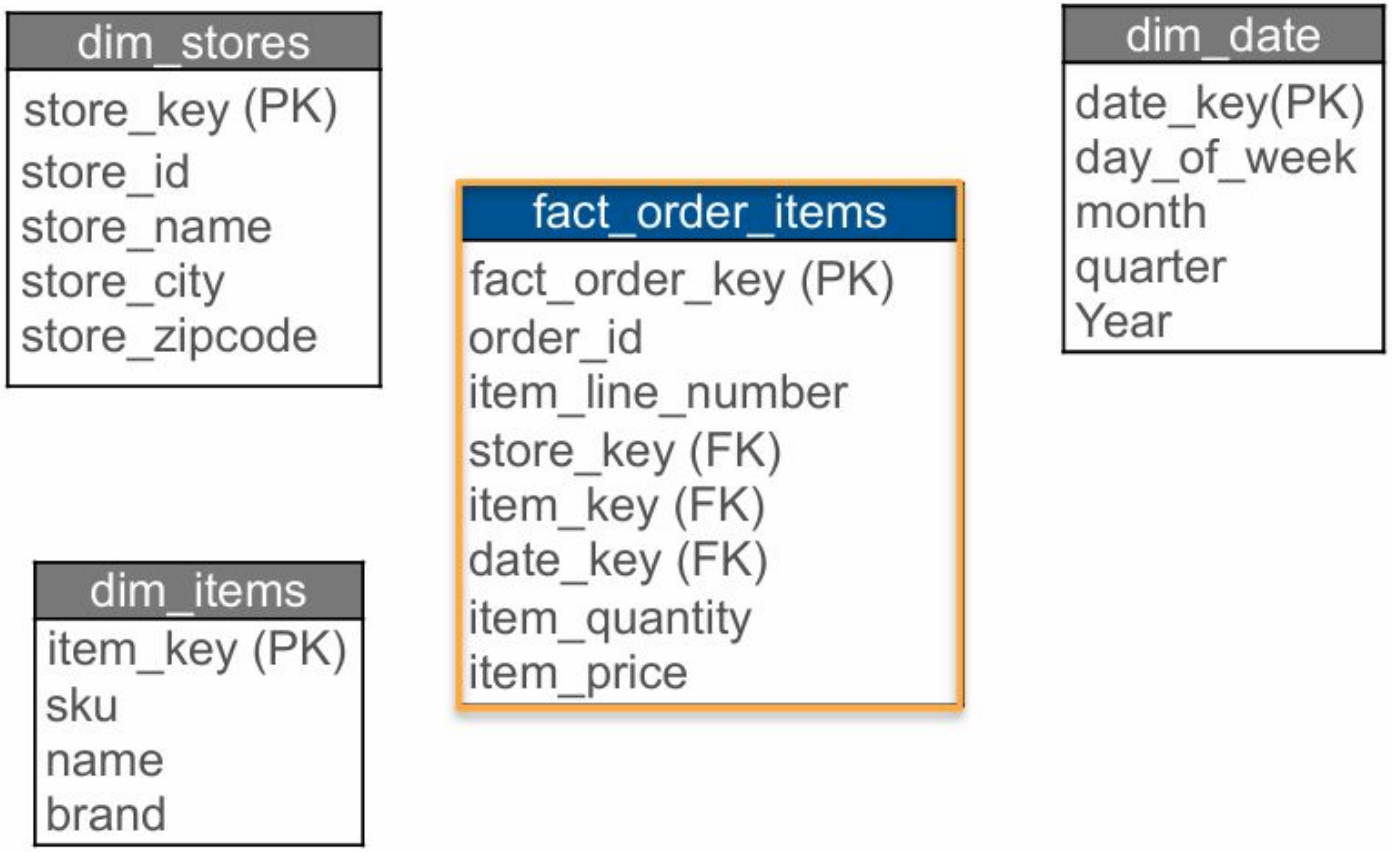
```
SELECT date_key,
  EXTRACT(DAY FROM date_key) AS day_of_week,
  EXTRACT(MONTH FROM date_key) AS month,
  EXTRACT(Quarter FROM date_key) AS quarter,
  EXTRACT(year FROM date_key) AS YEAR
FROM generate_series ('2020-01-01'::date,
                      '2025-01-01'::date,
                      '1 day'::interval) As date_key
```

Normalized Data

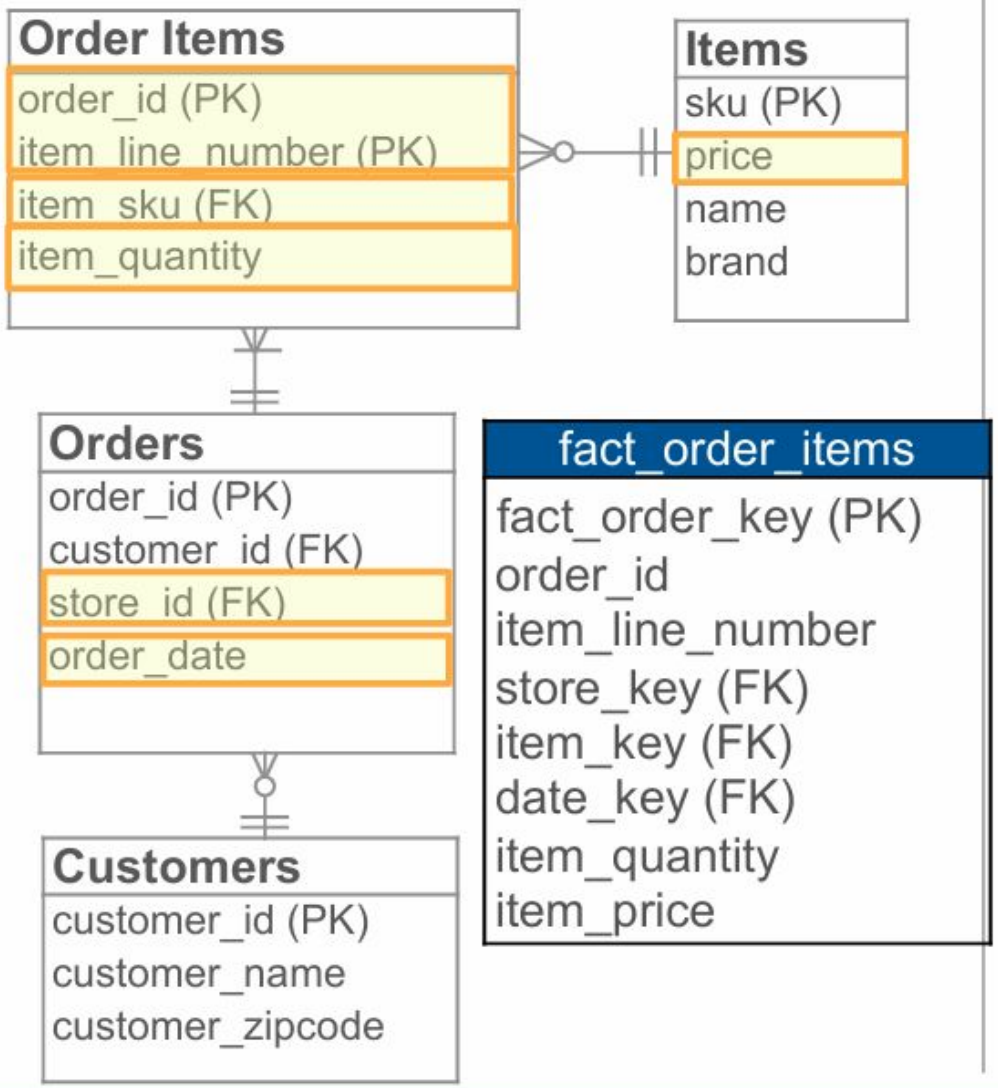


Star Schema

Bus. process: sales transactions → Grain: individual product in a transaction → 3. Identify the dimensions → 4. Identify the facts



Normalized Data

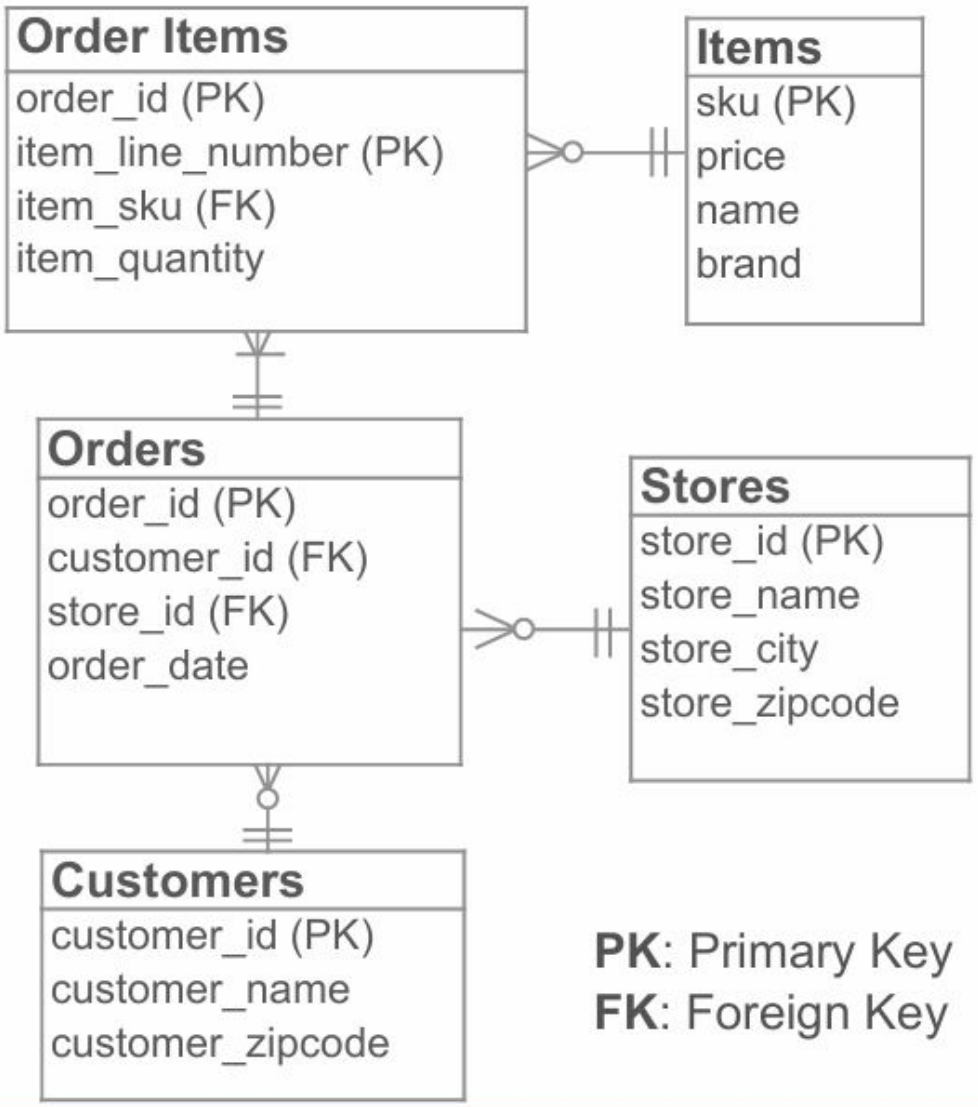


Star Schema

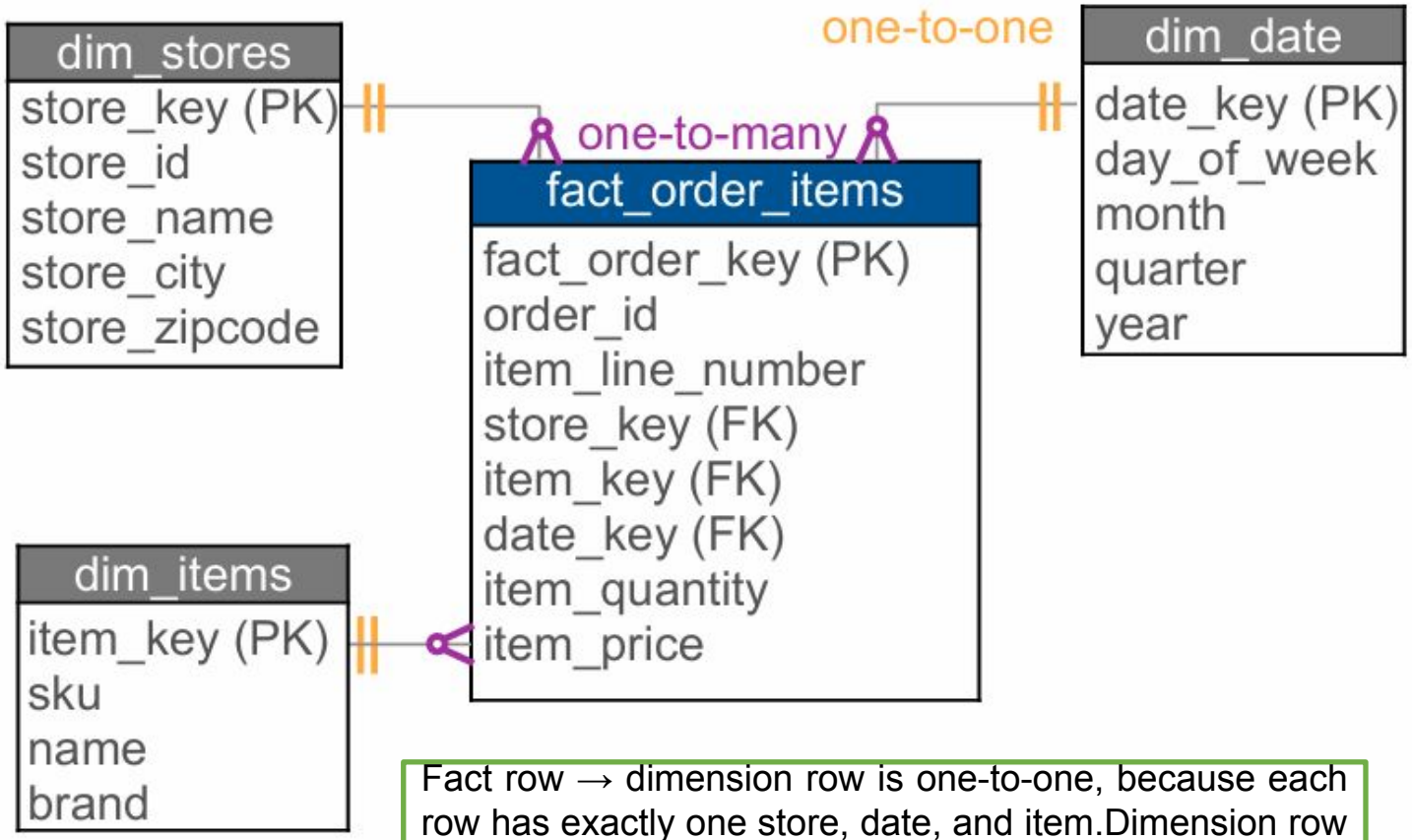
Bus. process: sales transactions → Grain: individual product in a transaction → Identify the dimensions → Identify the facts

```
Select
   MD5(CONCAT(OrderItems.order_id,
              OrderItems.item_line_number ))
   AS fact_order_key,
   OrderItems.order_id,
   OrderItems.item_line_number,
   MD5(Orders.store_id) AS store_key,
   MD5(OrderItems.item_sku) AS item_key,
   Orders.order_date AS date_key,
   OrderItems.item_quantity,
   Items.price AS item_price

FROM OrderItems
Join Orders ON Orders.order_id = OrderItems.order_id
Join Items ON Items.sku = OrderItems.item_sku
```

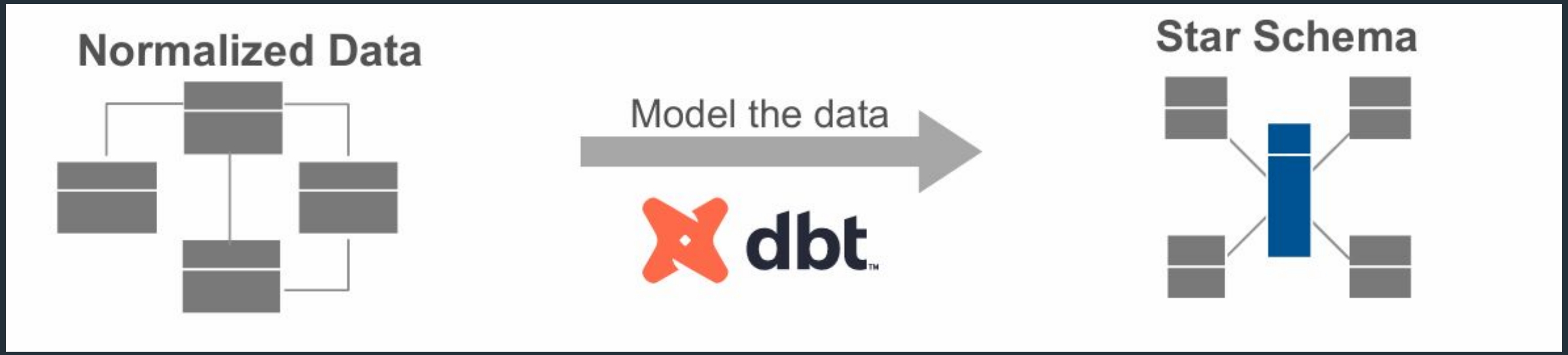
Normalized Data



Star Schema



Fact row → dimension row is one-to-one, because each row has exactly one store, date, and item.Dimension row → fact rows is one-to-many, because a store, item, or date can appear in multiple order items.

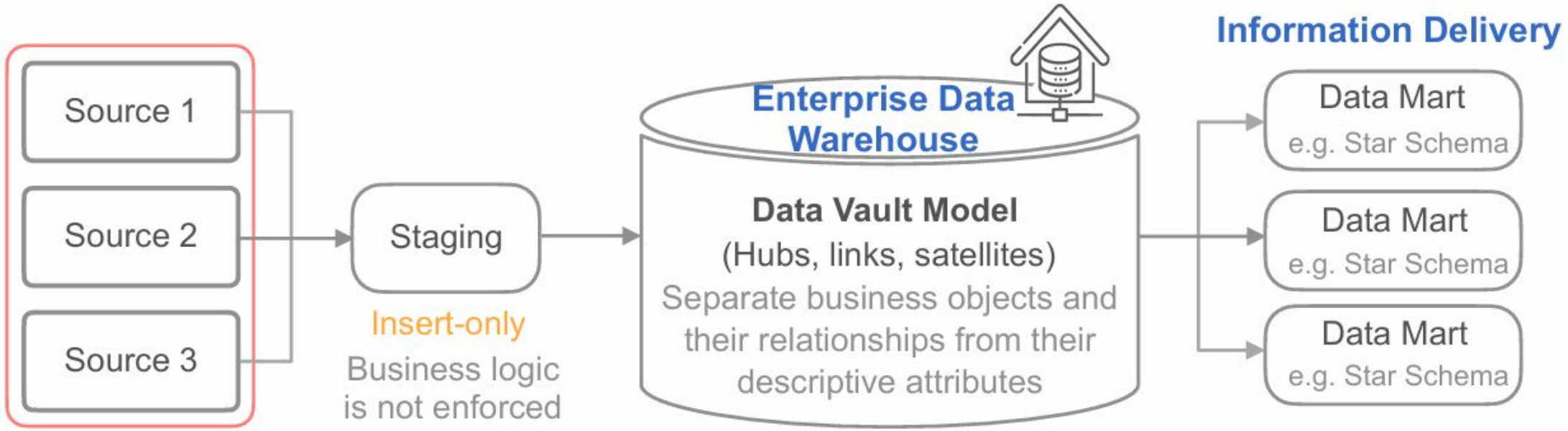

Normalized Data
Model the data
dbt
Star Schema

Data Modeling Techniques

Data Vault

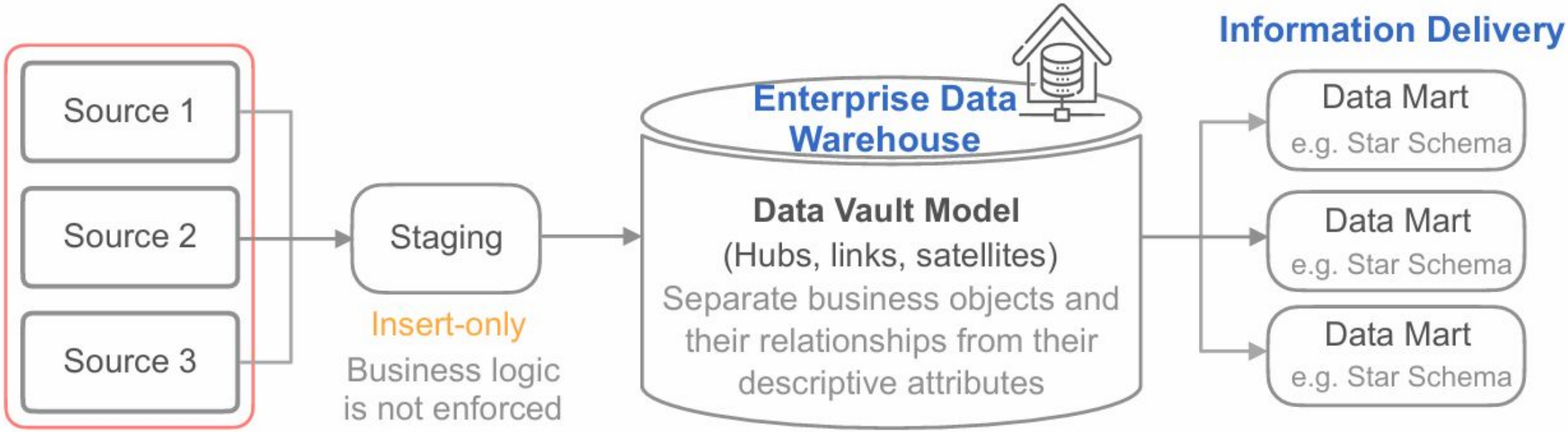
Data Vault

Dan Linstedt introduced Data Vault as a different approach to modeling the data in the data warehouse.



Data Vault

- No notion of good, bad, or conformed data in a data vault
- Only change the structure in which data is stored:
  - Allows you to trace the data back to its source
  - Helps you avoid restructuring the data when business requirements change



Data Vault Model

Three main types of tables:

Hub

Stores a unique list of business keys

Customers, products, employees, vendors

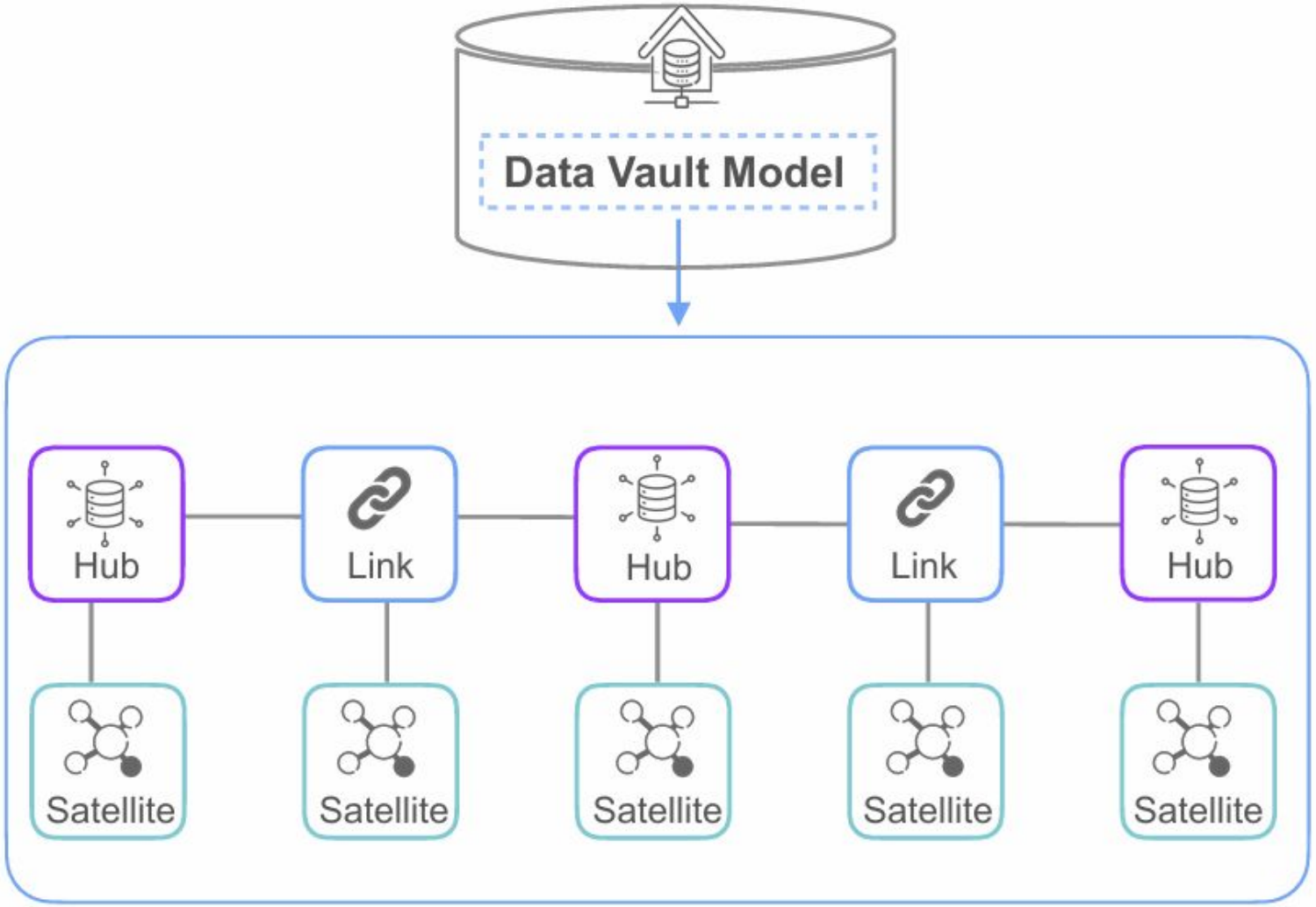
Link

Connects two or more hubs

Relationship, transaction, event

Satellite

Contains attributes that provide context for hubs and links

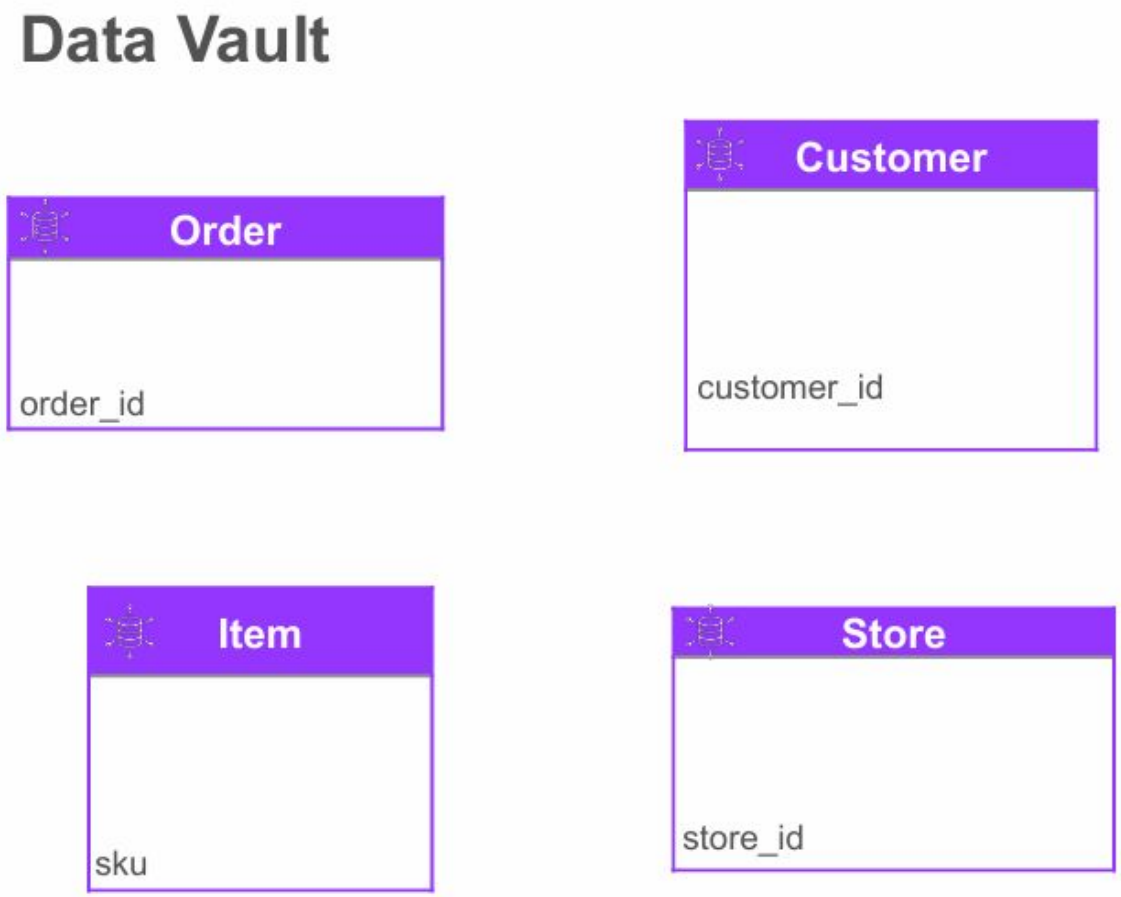


Data Vault - Step 1: Model the Hubs

Data Vault

- What is the identifiable business element?
- How do users commonly look for data?
- A business key:
  - column(s) used by the business to identify and locate the data
  - not be a key generated in or tied to a particular source system

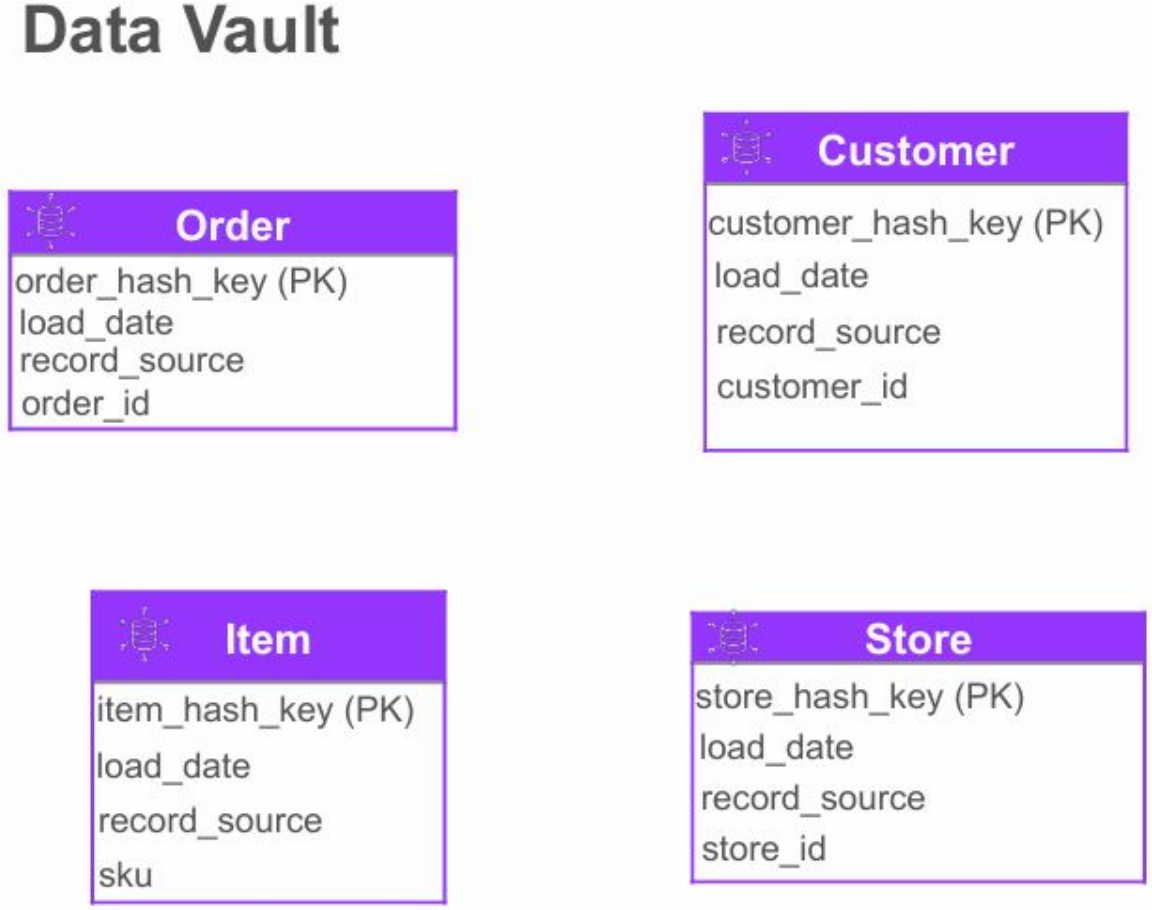
Data Vault - Step 1: Model the Hubs



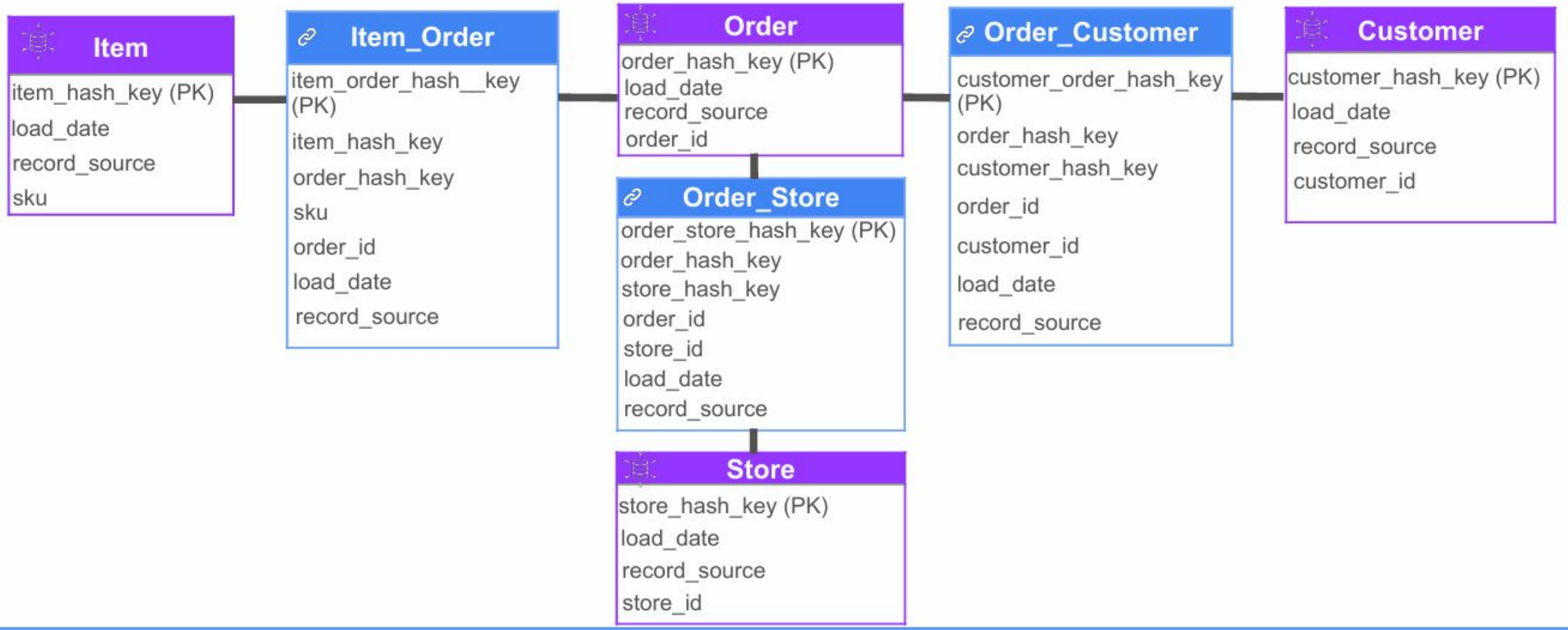
Data Vault - Step 1: Model the Hubs

A hub should contain:

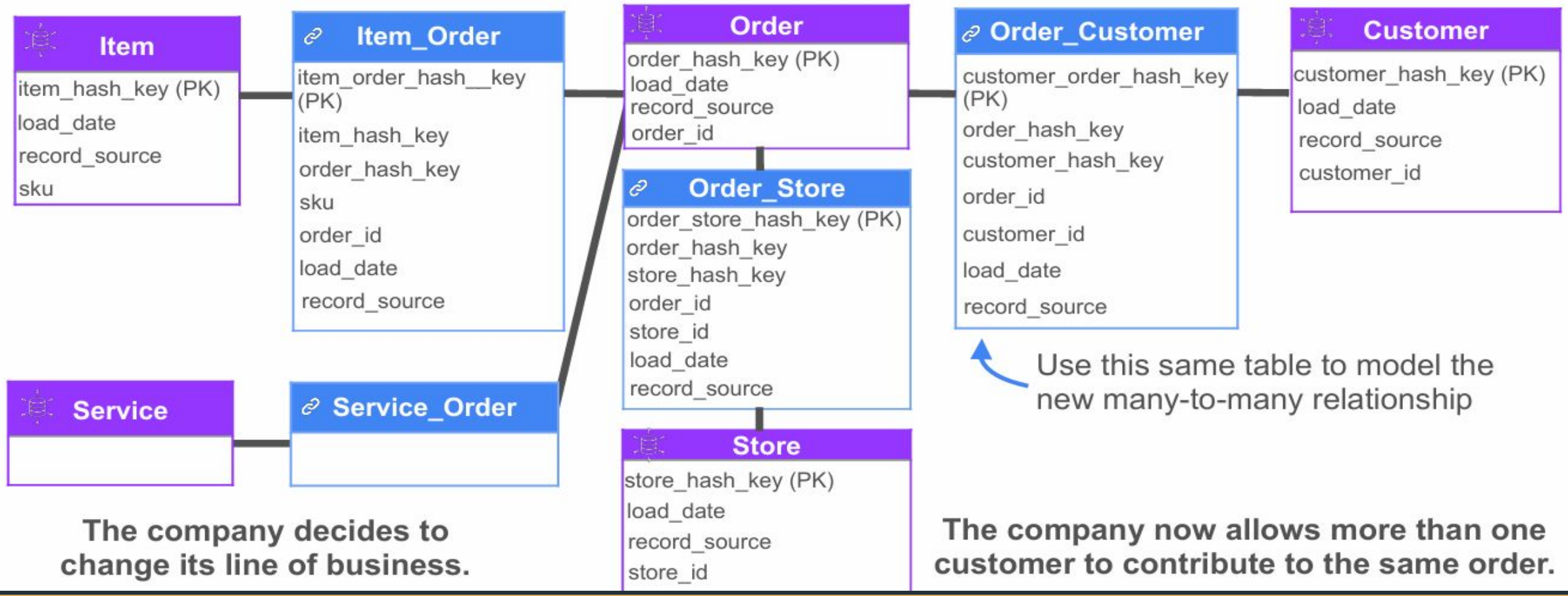
- **The business key**
- **The hash key**:
  - Calculated as a hash of the business key
  - Used as the Hub primary key
- **The load date**: date on which the business key was first loaded
- **The record source**: the source of the business key



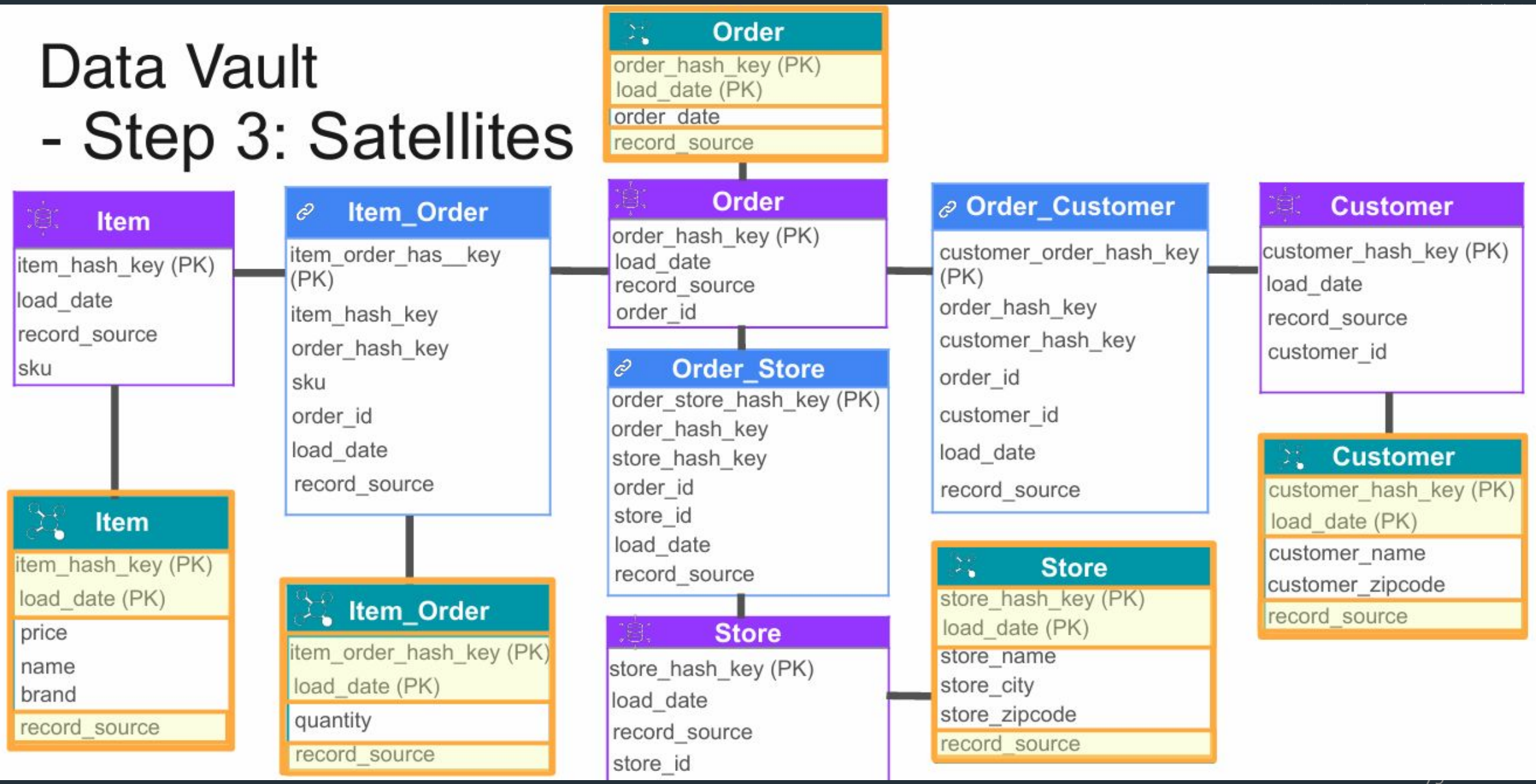
Data Vault - Step 2: Model the Links



Data Vault - Step 2: Model the Links



Data Vault
- Step 3: Satellites



Data Modeling Techniques

One Big Table

One Big Table (OBT)

Many columns
(Can be thousands of columns)

Field 1	Field 2	Field 3	Field 4	Field 5	Field 6	Field 7	Field 8
…	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…

Highly denormalized and flexible

Single value or nested data

Wide Table Example

- Can have hundreds or more columns
- Combines various data types

OrderID	Orderitems	CustomerID	CustomerName	address	OrderDate

100	[{
	“sku”:1,
	“price”:50,
	“quantity”: 1,
	“name”: “Thingamajig”
	}, {
	“sku”:2,
	“price”:25,
	“quantity”: 3,
	“name”: “Whatchamacallit”
	}]
		5	Joe Reis	1st. St	1/08/2024

101	[{
	“sku”:3,
	“price”:75,
	“quantity”: 1,
	“name”: “Whoozeewhatzit”
	}, {
	“sku”:2,
	“price”:25,
	“quantity”: 3,
	“name”: “Whatchamacallit”
	}]
		7	Matt Housely	2nd Ave.	1/08/2024

102	[{ “sku”:1, “price”:50, “quantity”: 1, “name”: “Thingamajig” }]	9	Colleen Fotsch	2nd Ave.	1/08/2024

Wide Table Example

- Can have hundreds or more columns
- Combines various data types
- No need for complex joins
- Supports fast analytical queries

OrderID	Orderitems	CustomerID	CustomerName	address	OrderDate
100	[{ "sku":1, "price":50, "quantity": 1, "name": "Thingamajig" }, { "sku":2, "price":25, "quantity": 3, "name": "Whatchamacallit" }]	5	Joe Reis	1st. St	1/08/2024
101	[{ "sku":3, "price":75, "quantity": 1, "name": "Whoozeewhatzit" }, { "sku":2, "price":25, "quantity": 3, "name": "Whatchamacallit" }]	7	Matt Housely	2nd Ave.	1/08/2024
102	[{ "sku":1, "price":50, "quantity": 1, "name": "Thingamajig" }]	9	Colleen Fotsch	2nd Ave.	1/08/2024

Facts

Dimensions

Why are OBTs becoming popular?

- Low cost of cloud storage 
- Nested data allows for flexible schemas
- Columnar storage helps optimize the storage and processing of OBTs:
  - Wide tables are sparse
  - Columnar database reads only columns selected in a query, and reading nulls is essentially free

Order ID	Price	Product SKU	Quantity	Customer ID
1	40	45865	10	67t
2	23	90234	14	56t
3	45	12558	12	87q
4	50	45682	13	98q

Cons

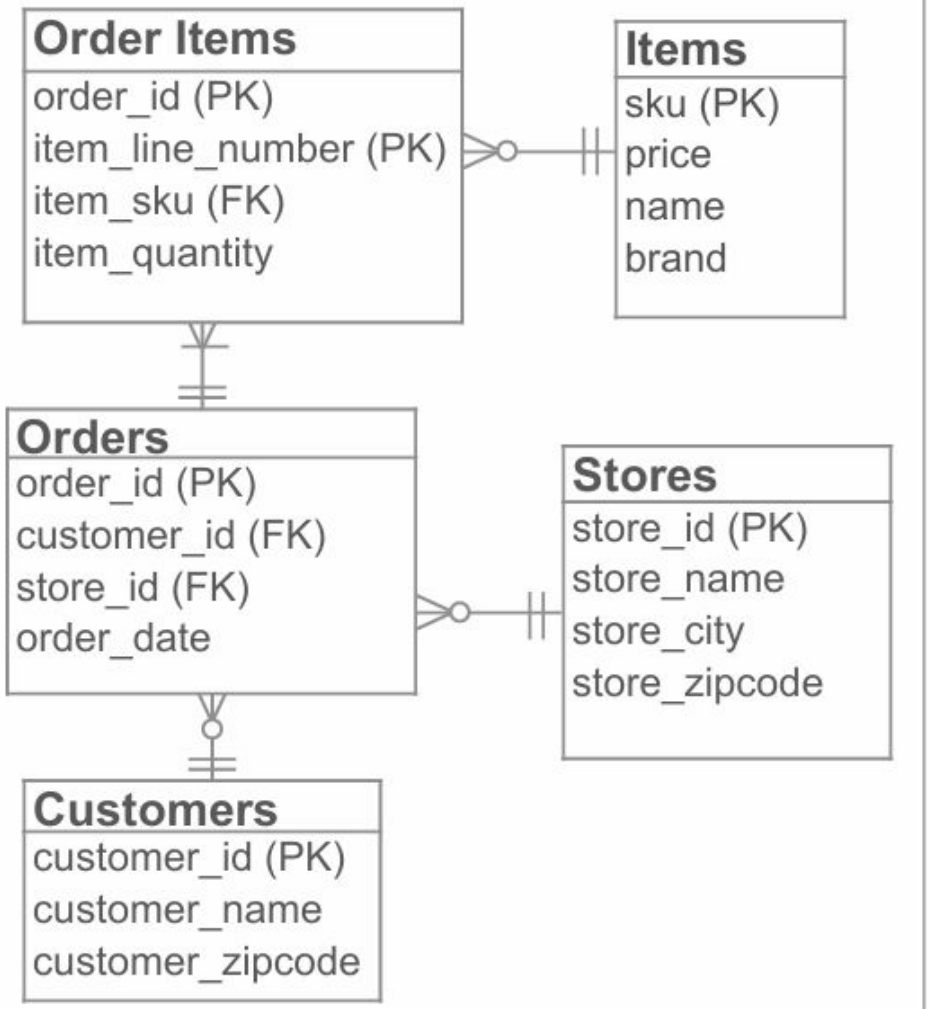
- You might lose the business logic in your analytics
- You need complex data structures to store nested data:
  - Can have poorer update and aggregation performance

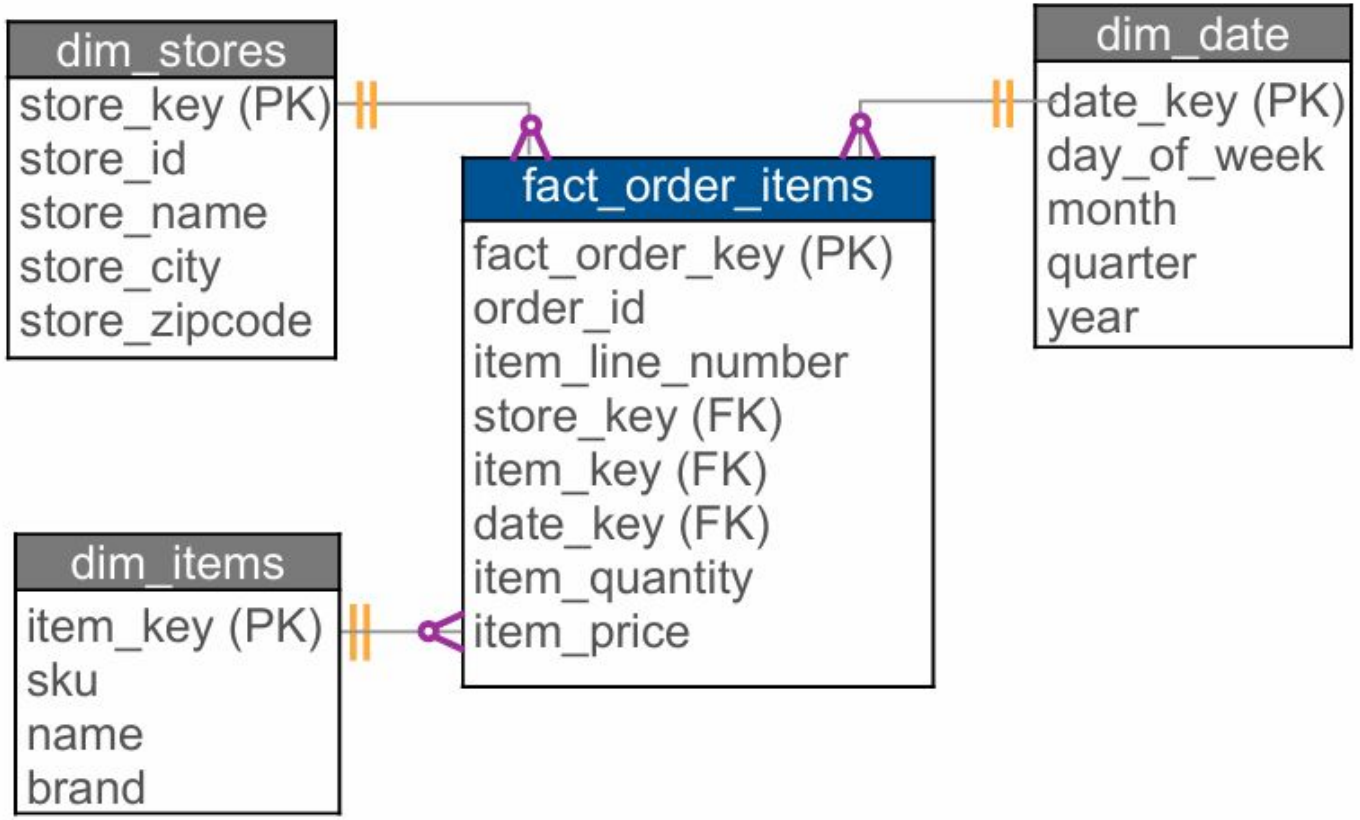
RV UNIVERSITY
Go, change the world
an initiative of RV EDUCATIONAL INSTITUTIONS

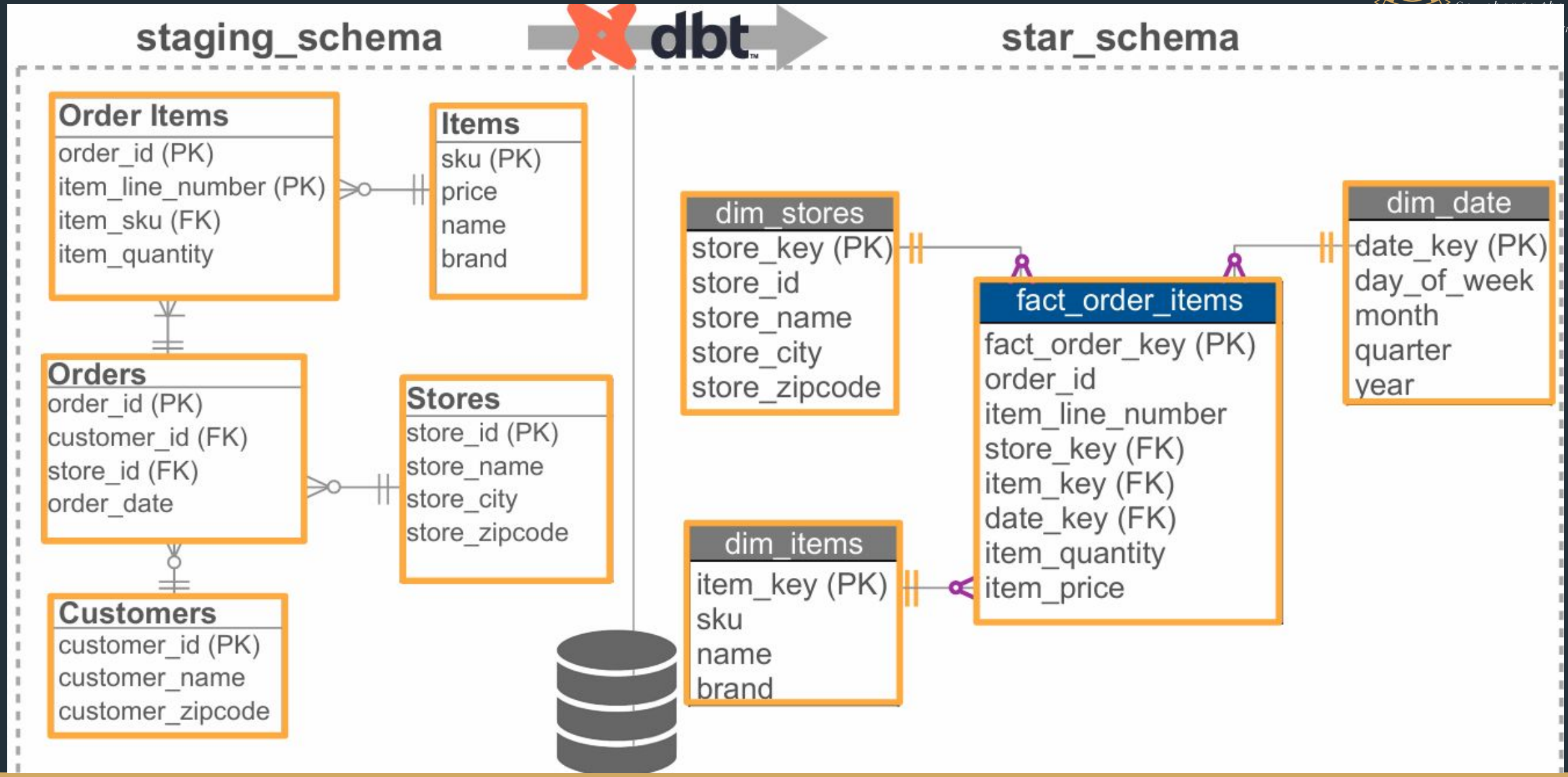
DBT Demo (Part 1)

Normalized Data

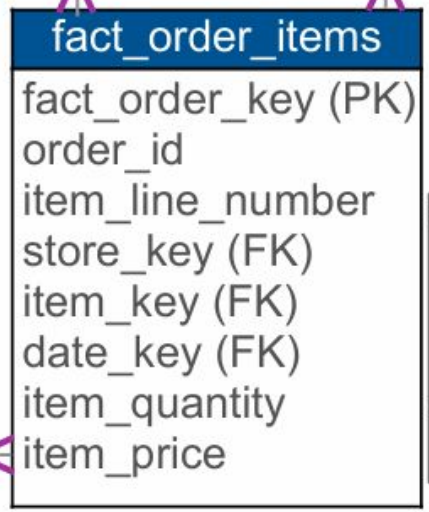
Star Schema





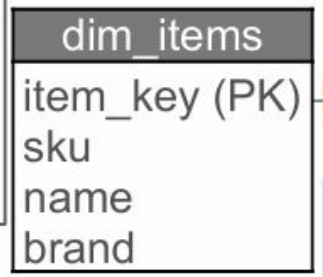

staging_schema
dbt
star_schema
Order Items
order_id (PK)
item_line_number (PK)
item_sku (FK)
item_quantity
Items
sku (PK)
price
name
brand
Orders
order_id (PK)
customer_id (FK)
store_id (FK)
order_date
Stores
store_id (PK)
store_name
store_city
store_zipcode
Customers
customer_id (PK)
customer_name
customer_zipcode
dim_stores
store_key (PK)
store_id
store_name
store_city
store_zipcode
dim_items
item_key (PK)
sku
name
brand
fact_order_items
fact_order_key (PK)
order_id
item_line_number
store_key (FK)
item_key (FK)
date_key (FK)
item_quantity
item_price
dim_date
date_key (PK)
day_of_week
month
quarter
year

```
SELECT MD5(store_id)
   as store_key,
 store_id,
 store_name,
 store_city,
 store_zipcode
FROM stores;
```




```
SELECT date_key,
 EXTRACT(DAY FROM date_key) AS day_of_week,
 EXTRACT(MONTH FROM date_key) AS month,
 EXTRACT(Quarter FROM date_key) AS quarter,
 EXTRACT(year FROM date_key) AS YEAR
FROM generate_series ('2020-01-01'::date,
 '2025-01-01'::date, '1 day'::interval) As date_key
```

```
SELECT
 MD5(sku) as item_key,
 sku,
 name,
 brand
FROM items;
```




```
Select MD5(CONCAT(OrderItems.order_id, OrderItems.item_line_number))
        AS fact_order_key,
  OrderItems.order_id, OrderItems.item_line_number,
  MD5(Orders.store_id) AS store_key, MD5(OrderItems.item_sku) AS item_key,
  Orders.order_date AS date_key, OrderItems.item_quantity,
  Items.price AS item_price
FROM OrderItems
Join Orders ON Orders.order_id = OrderItems.order_id
Join Items ON Items.sku = OrderItems.item_sku
```

dbt

- Wraps the SQL statement with a create statement
- Helps document and validate data within the data warehouse

- dbt Core:
  - open-source command line tool that you can install locally
  - communicate with your databases through adapters

- dbt Cloud:
  - runs dbt core in hosted environment with a browser-based interface

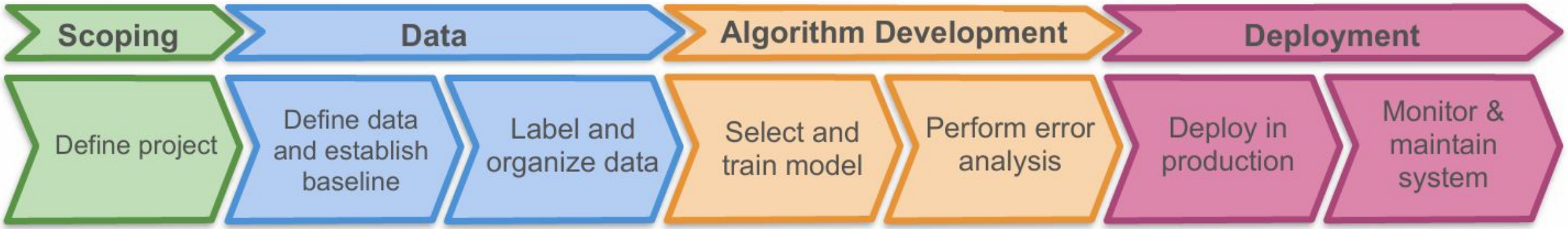
RV UNIVERSITY
Go, change the world
an initiative of RV EDUCATIONAL INSTITUTIONS

Data Transformation, Modeling and Serving

Data Modeling and Transformation for Machine Learning

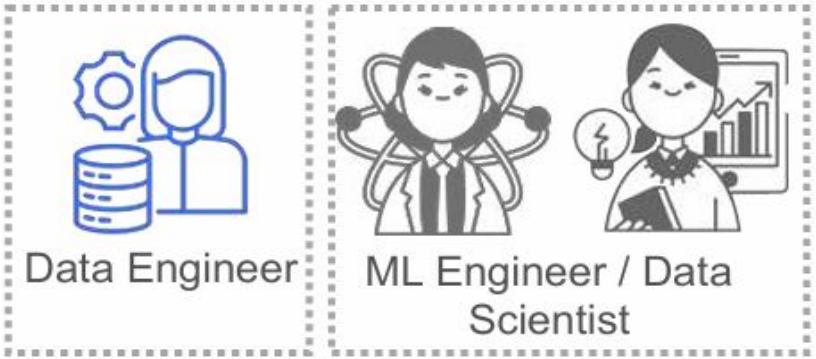
Data Engineering for Machine Learning

- Shape the data into a format suitable for the ML algorithm
- Maybe clean the data, convert it, or even create additional columns



Machine Learning Project Lifecycle Framework

Data Engineer, Data Scientist and Machine Learning Engineer



Separate teams

Serve raw data

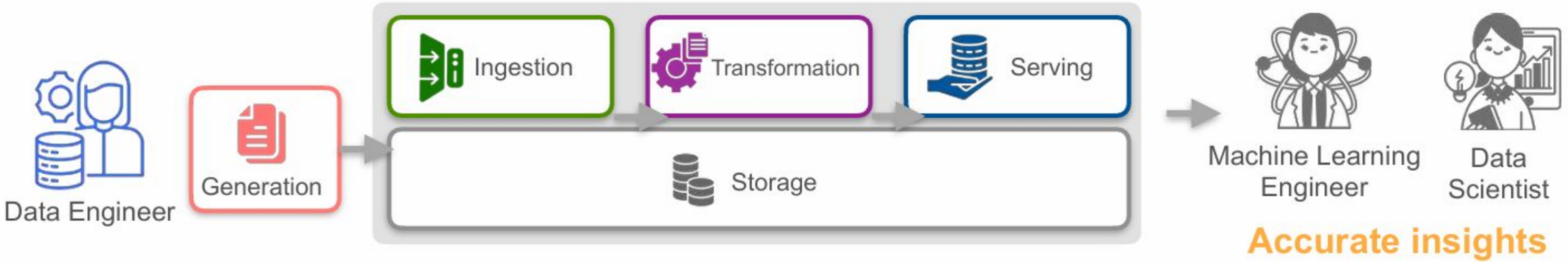
Process the raw data

Process the raw data

Use the processed data for model training

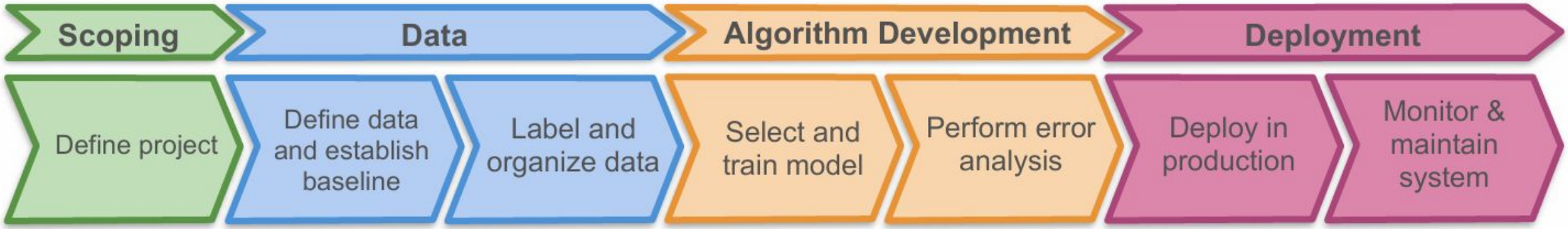
No mature ML team

Handle some extremely ML-specific tasks

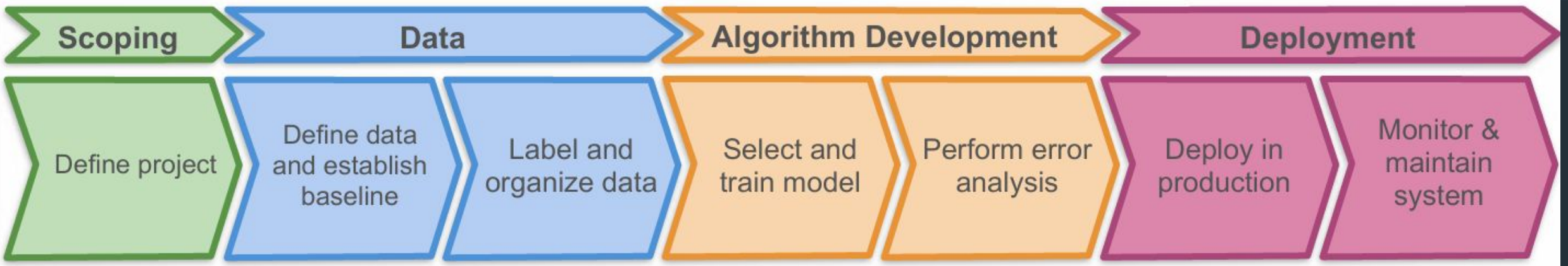


Help organization adopt a **data-centric** approach to ML:

- Enhance the ML system by collecting high-quality data
- "Garbage in, garbage out"



Machine Learning Project Lifecycle Framework



Data Engineer

Set up the pipeline to serve data that supports these phases

Modeling and Processing Tabular Data for Machine Learning

Modeling Data for Traditional Machine Learning Algorithms

Numerical Tabular Form

	No. of items purchased	Date of last purchase	Customer income	Minutes on Platform	Account type	Churned
Customer	14	7/5/2024	$50,000	15	Family	No
	9	3/4/2024	$40,000	13	Platinum	Yes
	Null	8/12/2024	Null	Null	Null	Yes
	2	8/24/2024	Null	35	Basic	No
	…	…	…	…	…	…

Numerical Tabular Form

What most classical ML algorithms expect as training data

No. of items purchased	Days since last purchase	Customer income	Minutes on Platform	Account type	Purchases per minute	Churned
0.93	0.90	0.5	0.24	1	0.93	0
0.57	0.29	0.4	0.20	2	0.69	1 Churn
0.07	0.03	0.35	0.64	0	0.06	0 Not Churn
…	…	…	…	…	…	…

Features **Labels**

- No missing values or duplicate rows
- Each column consists of numerical values that are within a similar range

Feature Engineering

Feature Engineering

Any change or processing done to a raw column, and any creation of new features

- Handling missing values
- Feature scaling
- Converting categorical columns into numerical ones
- Creating new columns by combining or modifying existing ones

Handling Missing Values

Understand why the values are missing and then determine the most appropriate way

No. of items purchased	Days since last purchase	Customer income	Minutes on Platform	Account type	Churned
14	28	$50,000	15	Family	No
9	9	$40,000	13	Platinum	Yes
Null	12	Null	Null	Null	Yes
2	1	Null	35	Basic	No
…	…	…	…	…	…

- Delete the entire column or row (if there's no risk of losing valuable data)
- Impute the missing values with summary statistics
  - Replace missing values with the column mean or median
  - Replace missing values with values from a similar record

Handling Missing Values

Understand why the values are missing and then determine the most appropriate way

No. of items purchased	Days since last purchase	Customer income	Minutes on Platform	Account type	Churned
14	28	$50,000	15	Family	No
9	9	$40,000	13	Platinum	Yes
2	1	$35,000	35	Basic	No
…	…	…	…	…	…

- Delete the entire column or row (if there's no risk of losing valuable data)
- Impute the missing values with summary statistics
  - Replace missing values with the column mean or median
  - Replace missing values with values from a similar record

Scaling Numerical Features

Scale features so that the values of each feature end up within a similar range

No. of items purchased	Days since last purchase	Customer income	Minutes on Platform	Account type	Churned
14	28	$50,000	15	Family	No
9	9	$40,000	13	Platinum	Yes
2	1	$35,000	35	Basic	No
…	…	…	…	…	…

- Training an ML algorithm is based on solving an optimization problem:
  - If values vary drastically → take longer for the optimization algorithm to converge
- Certain ML algorithms are based on distance metrics:
  - Their accuracies can be affected by different ranges of values

Scaling Numerical Features

Scale features so that the values of each feature end up within a similar range

No. of items purchased	Days since last purchase	Customer income	…	…	…rned
14	28	0.5			…lo
9	9	0.4			…es
2	1	$35,000			…lo
…	…	…	…	…	…

$$\frac{\$50{,}000 - \$0}{\$100{,}000 - \$0} = 0.5$$

$$\frac{\$40{,}000 - \$0}{\$100{,}000 - \$0} = 0.4$$

Min: $0
Max: $100,000

Standardization

$$\frac{\text{value - column mean}}{\text{column standard deviation}}$$

Resulting value has mean of 0 and variance of 1

Min-Max Scaling

$$\frac{\text{value - column min}}{\text{column max - column min}}$$

Resulting value is between 0 and 1

Converting Categorical Columns into Numerical Ones

No. of items purchased	Days since last purchase	Customer income	Minutes on Platform	Account type	Churned
14	28	0.5	15	Family	No
9	9	0.4	13	Platinum	Yes
2	1	0.35	35	Basic	No
...	...	...	...	...	...

One Hot Encoding

Account type	Basic	Family	Platinum
Family	0	1	0
Platinum	0	0	1
Basic	1	0	0
...	...	...	...

Ordinal Encoding

Account type
Family
Platinum
Basic
...

middle

most expensive

cheapest

Account type
2
3
1
...

Embeddings

More on this later

RV UNIVERSITY
Go, change the world
an initiative of RV EDUCATIONAL INSTITUTIONS

Modeling and Processing Tabular Data for Machine Learning

Processing Tabular Data for Classical Machine Learning Algorithms Using Scikit-Learn (Part 1)

scikit-learn

Machine Learning in Python

Getting Started | Release Highlights for 1.5

- Simple and efficient tools for predictive data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

Two Processing Methods:

- Standardization for the numerical columns
- One-hot encoding for the categorical columns

Classification

Identifying which category an object belongs to.

Applications: Spam detection, image recognition.
Algorithms: Gradient boosting, nearest neighbors, random forest, logistic regression, and more...

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, stock prices.
Algorithms: Gradient boosting, nearest neighbors, random forest, ridge, and more...

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, grouping experiment outcomes.
Algorithms: k-Means, HDBSCAN, hierarchical clustering, and more...

Download Dataset

CustomerID	Age	Tenure	Usage Frequency	Support Calls	Payment Delay	Subscription Type	Contract Length	Total Spend	Last Interaction	Churn
1	22	25	14	4	27	Basic	Monthly	598	9	1
2	41	28	28	7	13	Standard	Monthly	584	20	0
3	47	27	10	2	29	Annual	Annual	757	21	0
…	…	…	…	…	…	…	…	…	…	…

Preparing Data for Training a Machine Learning Model

ML Engineer Team

Dataset

CustomerID	Age	Tenure	Usage Frequency	Support Calls	Payment Delay	Subscription Type	Contract Length	Total Spend	Last Interaction	Churn
1	22	25	14	4	27	Basic	Monthly	598	9	1
2	41	28	28	7	13	Standard	Monthly	584	20	0
3	47	27	10	2	29	Annual	Annual	757	21	0
...	...	...	...	...	...	...	...	...	...	...

80%

20%

Training Dataset

Customer_id
Standardized numerical columns
One-hot encoded categorical columns

Testing Dataset

Customer_id
Standardized numerical columns
One-hot encoded categorical columns

.parquet

Preparing Data for Training a Machine Learning Model

1. **Split the data into training and test sets**

2. **Process the training data**
   a. Numerical columns → standardize
   b. Categorical columns → one hot encoding
   c. Combine processed columns with the Customer ID into a Pandas data frame
   d. Convert Pandas data frame into a parquet file

Makes it easier

3. **Process the test data**
   a. Numerical columns → standardize
   b. Categorical columns → one hot encoding
   c. Combine processed columns with the Customer ID
   d. Convert Pandas data frame into a parquet file

Use the same computed statistics used on the training set

RV UNIVERSITY
Go, change the world
an initiative of RV EDUCATIONAL INSTITUTIONS

Modeling and Processing Tabular Data for Machine Learning

Processing Tabular Data for Classical Machine Learning Algorithms Using Scikit-Learn (Part 2)

Preparing Data for Training a Machine Learning Model

1. **Split the data into training and test sets**

2. **Process the training data**
   a. Numerical columns → standardize
   b. Categorical columns → one hot encoding
   c. Combine processed columns with the Customer ID into a Pandas data frame
   d. Convert Pandas data frame into a parquet file

3. **Process the test data**
   a. Numerical columns → standardize
   b. Categorical columns → one hot encoding
   c. Combine processed columns with the Customer ID
   d. Convert Pandas data frame into a parquet file

Modeling and Processing Unstructured Data for Machine Learning

Modeling Image Data for Machine Learning Algorithms

Training an ML Algorithm on Image Data

Traditional ML algorithms

No. of items purchased	Days since last purchase	Customer income	Minutes on Platform	Account type	Purchases per minute	Churned
0.93	0.90	0.5	0.24	1	0.93	0
0.57	0.29	0.4	0.20	2	0.69	1
0.07	0.03	0.35	0.64	0	0.06	0
…	…	…	…	…	…	…

- Lose spatial information that can be extracted from the relative location of pixels
- Can create a high-dimensional vector of features
  - e.g. 1000 pixels by 1000 pixels → vector of size 1 million
- Affect the performance of the ML algorithm

Training an ML Algorithm on Image Data

Convolutional Neural Network (CNN)



Each layer tries to identify more image features to help with the ML task

- First layer: Generic features
- Later layers: Complex patterns and textures

ML Engineer Team

- Start with pre-trained CNN algorithms
- Fine tune these models for the specific task

Preparing Image Data for the Training an ML Algorithm

Resizing

Flipping

Rotating

Scaling the pixels

Cropping

Adjusting brightness

Data Engineer

TensorFlow

Data Augmentation

Technique used to create new versions of existing images

(Increases the size & variety of training data)

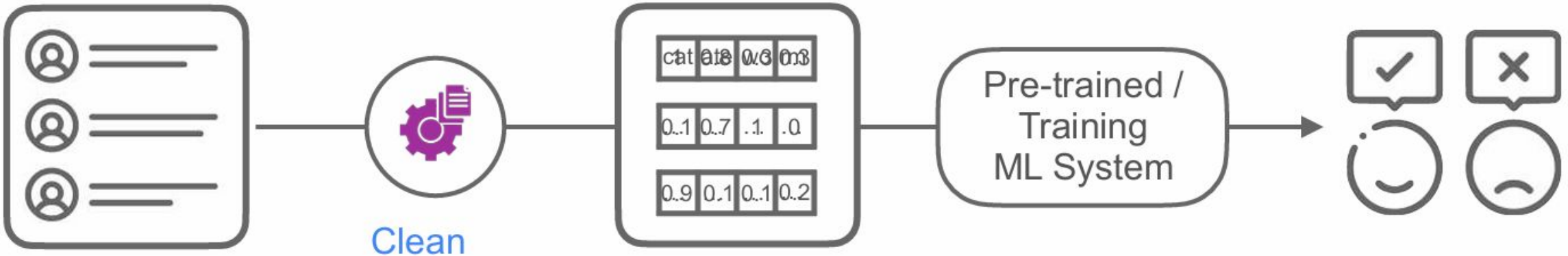
RV UNIVERSITY
Go, change the world
an initiative of RV EDUCATIONAL INSTITUTIONS

Modeling and Processing Unstructured Data for Machine Learning

Preprocessing Textual Data for Analysis and Text Classification

Pre-processing Texts for ML

Sentiment Analysis



This is a wonderful price for the amount #@% you get ….

★★★★

Depends on the type of ML algorithm:

- Classical ML algorithm requires numerical data
- LLMs can work with tokens or words

Pre-processing Texts for ML

Textual data might contain typos, inconsistencies, & repetitions

May contain words or characters not relevant to the NLP task

Training LLMs is expensive and time consuming

Data Engineer

Clean and high-quality data

Remove any irrelevant words or characters

ML Engineer Team

Train a classical or advanced ML model

Other use cases

Processing Text

- **Cleaning**
- Normalization
- Tokenization
- Removal of Stop Words
- Lemmatization

Removing punctuations, extra spaces, characters that add no meaning

Reviews	Cleaned Reviews
This is a wonderful price for the amount #@% you get ….	This is a wonderful price for the amount you get
Great product! Big amt	Great product Big amt
I bought this for my son as his hair is thinning. I don't know yet how well is helping. He said the smell is great.	I bought this for my son as his hair is thinning I don't know yet how well is helping He said the smell is great

Processing Text

Cleaning

Normalization

Tokenization

Removal of Stop Words

Lemmatization

Converting texts to consistent format:

- Transforming to lower-case
- Converting numbers or symbols to characters
- Expanding contractions

kg → kilograms

lbs → pounds

DE / D.E → data engineering

Cleaned Reviews	Normalized Reviews
This is a wonderful price for the amount you get	this is a wonderful price for the amount you get
Great product Big amt → amount	great product big amount
I bought this for my son as his hair is thinning I don't know yet how well is helping He said the smell is great	i bought this for my son as his hair is thinning i do not know yet how well is helping he said the smell is great

do not

Processing Text

- Cleaning
- Normalization
- **Tokenization**
- Removal of Stop Words
- Lemmatization

Splitting each review into individual tokens
(words, subwords, short sentences)

Normalized Reviews
this is a wonderful price for the amount you get
great product big amount
i bought this for my son as his hair is thinning i do not know yet how well is helping he said the smell is great

Tokenized Reviews
[this, is, a, wonderful, price, for, the, amount, you, get]
[great, product, big, amount]
[i, bought, this, for, my, son, as, his, hair, is, thinning, i, do, not, know, yet, how, well, is, helping, he, said, the, smell, is, great]

Processing Text

Cleaning

Normalization

Tokenization

Removal of Stop Words

Lemmatization

- Removing frequently used words such as “is”, “are”, “the”, “for”, “a”
- Define your own list of stop words
- Or use built-in set of NLP libraries

spaCy NLTK Gensim TextBlob

Tokenized Reviews
[this, is, a, wonderful, price, for, the, amount, you, get]
[great, product, big, amount]
[i, bought, this, for, my, son, as, his, hair, is, thinning, i, do, not, know, yet, how, well, is, helping, he, said, the, smell, is, great]

Stop Words Removed
[this, wonderful, price, amount, you, get]
[great, product, big, amount]
[i, bought, this, my, son, his, hair, thinning, i, do, not, know, yet, how, well, helping, he, said, smell, great]

Stop words: {is, a, for, the, as, are}

Processing Text

- Cleaning
- Normalization
- Tokenization
- Removal of Stop Words
- **Lemmatization**

Replacing each word with its base form or lemma (using NLP libraries)

getting / got → get

Stop Words Removed
[this, wonderful, price, amount, you, get]
[great, product, big, amount]
[i, bought, this, my, son, his, hair, thinning, i, do, not, know, yet, how, well, helping, he, said, smell, great]

Tokenized and Lemmatized Reviews
[this, wonderful, price, amount, you, get]
[great, product, big, amount]
[i, buy, this, my, son, his, hair, thin, i, do, not, know, yet, how, well, help, he, say, smell, great]

RV UNIVERSITY
Go, change the world
RV EDUCATIONAL INSTITUTIONS

Modeling and Processing Unstructured Data for Machine Learning

Text Vectorization and Embedding

Traditional Vectorization

Bag of Words

Term-Frequency Inverse-Document-Frequency (TF-IDF)

Reviews
[this, wonderful, price, amount, you, get] → A document
[great, product, big, amount]
[I, buy, this, my, son, his, hair, thin, I, do, not, know, yet, how, well, help, he, say, smell, great]

The corpus

The vocabulary

[this, wonderful, price, amount, you, get, great, product, big, I, buy, my, son, his, hair, thin, do, not, know, yet, how, well, help, he, say, smell]

this	wonderful	price	amount	you	get	great	product	big	I	buy	my	son	his	hair	thin	do	not	not	know	yet	how	well	help	he	say	smell

Example

"purchase" "buy" → High frequency, little meaning

"break" "exceptional" → Low frequency, more significant

Bag of Words

Term-Frequency Inverse-Document-Frequency (TF-IDF)

Each entry: number of occurrences

- Only takes into account the word frequency in each document
- Some frequently appearing words might carry little meaning

scikit learn

The corpus

Reviews
[this, wonderful, price, amount, you, get] → A document
[great, product, big, amount]
[I, buy, this, my, son, his, hair, thin, I, do, not, know, yet, how, well, help, he, say, smell, great]

this	wonderful	price	amount	you	get	great	product	big	I	buy	my	son	his	hair	thin	do	not	not	know	yet	how	well	help	he	say	smell
1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	1	0	0	2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bag of Words

Term-Frequency Inverse-Document-Frequency (TF-IDF)

Account for the weight and rarity of each word

TF: the number of times the term occurred in a document divided by the length of that document

IDF: how common or rare that word is in the entire corpus.

The corpus

Reviews
[this, wonderful, price, amount, you, get] → A document
[great, product, big, amount]
[I, buy, this, my, son, his, hair, thin, I, do, not, know, yet, how, well, help, he, say, smell, great]

Bag of Words

Term-Frequency Inverse-Document-Frequency (TF-IDF)

Account for the weight and rarity of each word

TF: the number of times the term occurred in a document divided by the length of that document

IDF: how common or rare that word is in the entire corpus.

The corpus

Reviews
[this, wonderful, price, amount, you, get] → A document
[great, product, big, amount]
[I, buy, this, my, son, his, hair, thin, I, do, not, know, yet, how, well, help, he, say, smell, great]

this	wonderful	price	amount	you	get	great	product	big	I	buy	my	son	his	hair	thin	do	not	not	know	yet	how	well	help	he	say	smell
0.33	0.44	0.44	0.33	0.44	0.44	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0.43	0	0	0.43	0.56	0.56	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0.16	0	0	0	0	0	0.16	0	0	0.43	0.22	0.22	0.22	0.22	0.22	0.22	0.22	0.22	0.22	0.22	0.22	0.22	0.22	0.22	0.22	0.22	0.22

Word Embedding

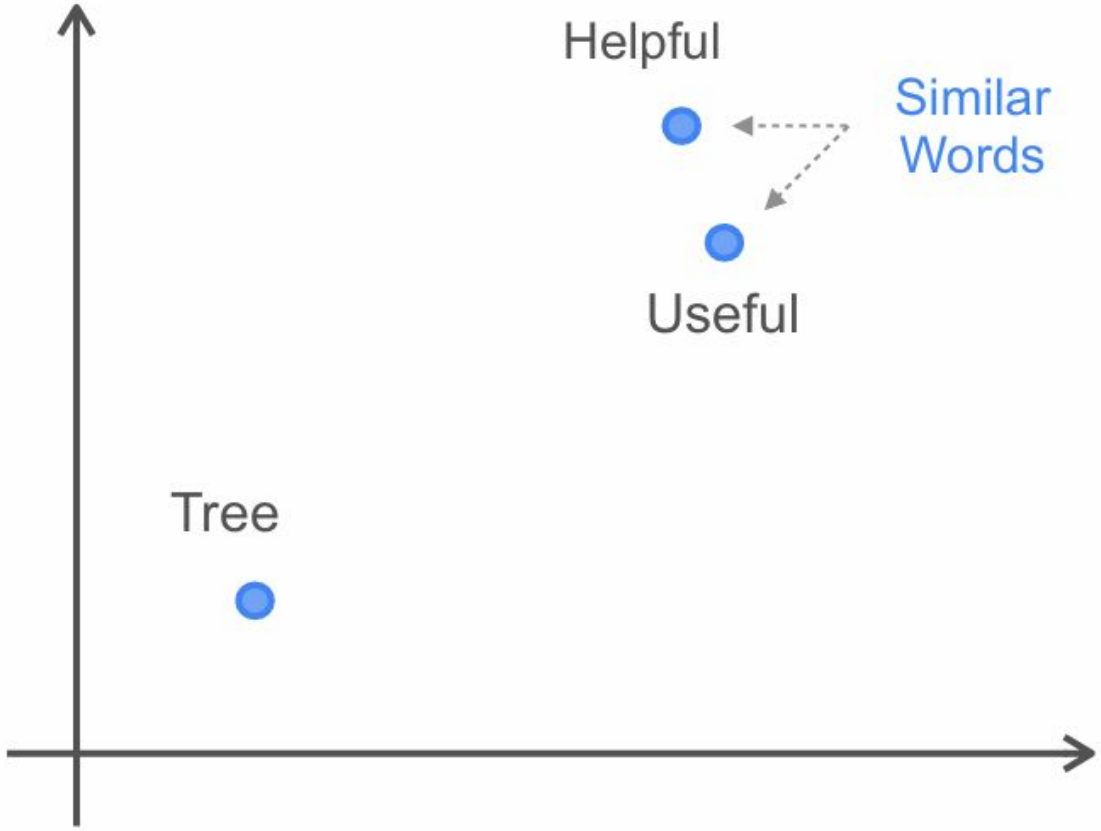
Word

word2vec, GLOVE

Trained to learn the embeddings of words from their co-occurrences

Word Embedding

Vector that captures the semantics meaning of the word



Word Embedding

Reviews
[this, wonderful, price, amount, you, get]
[great, product, big, amount]
[I, buy, this, my, son, his, hair, thin, I, do, not, know, yet, how, well, help, he, say, smell, awesome]

Represent each review, not just each word, by one vector

Word Embedding (great)
+
Word Embedding (product)
+
Word Embedding (big)
+
Word Embedding (amount)

→ **Vector that represents the sentence**

Does not account for the position of the words in the sentence

"A man ate a snake" ≠ "A snake ate a man"

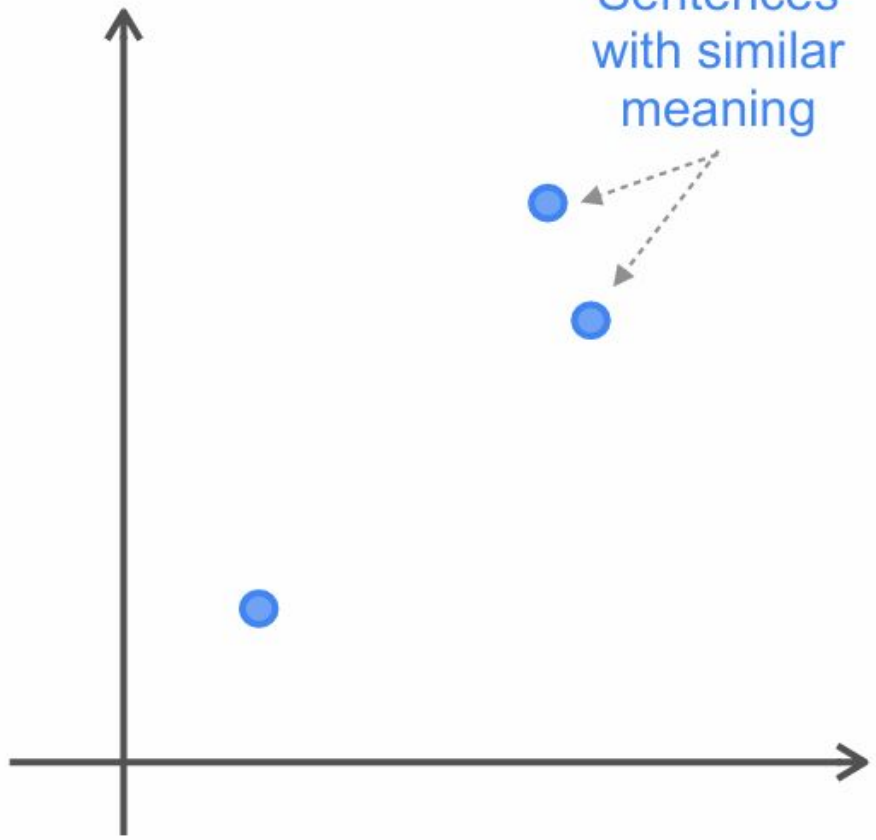
Sentence Embedding

Sentence

Pre-trained NLP models

Sentence Embedding

- Vector that reflects the semantic meaning of the sentence
- Lower dimension than the vector generated by TF-IDF



Sentence Embedding

Sentence

Pre-trained NLP models

based on Large Language Models (LLM)

Open-Source

SBERT.net

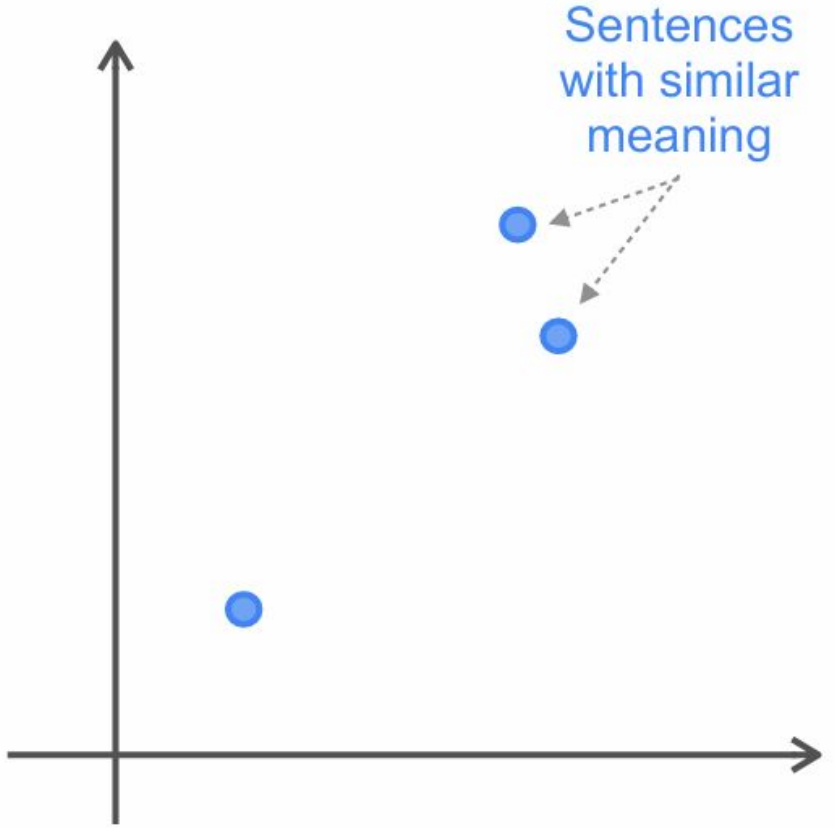
Sentence Transformers

Closed-Source

OpenAI

AI

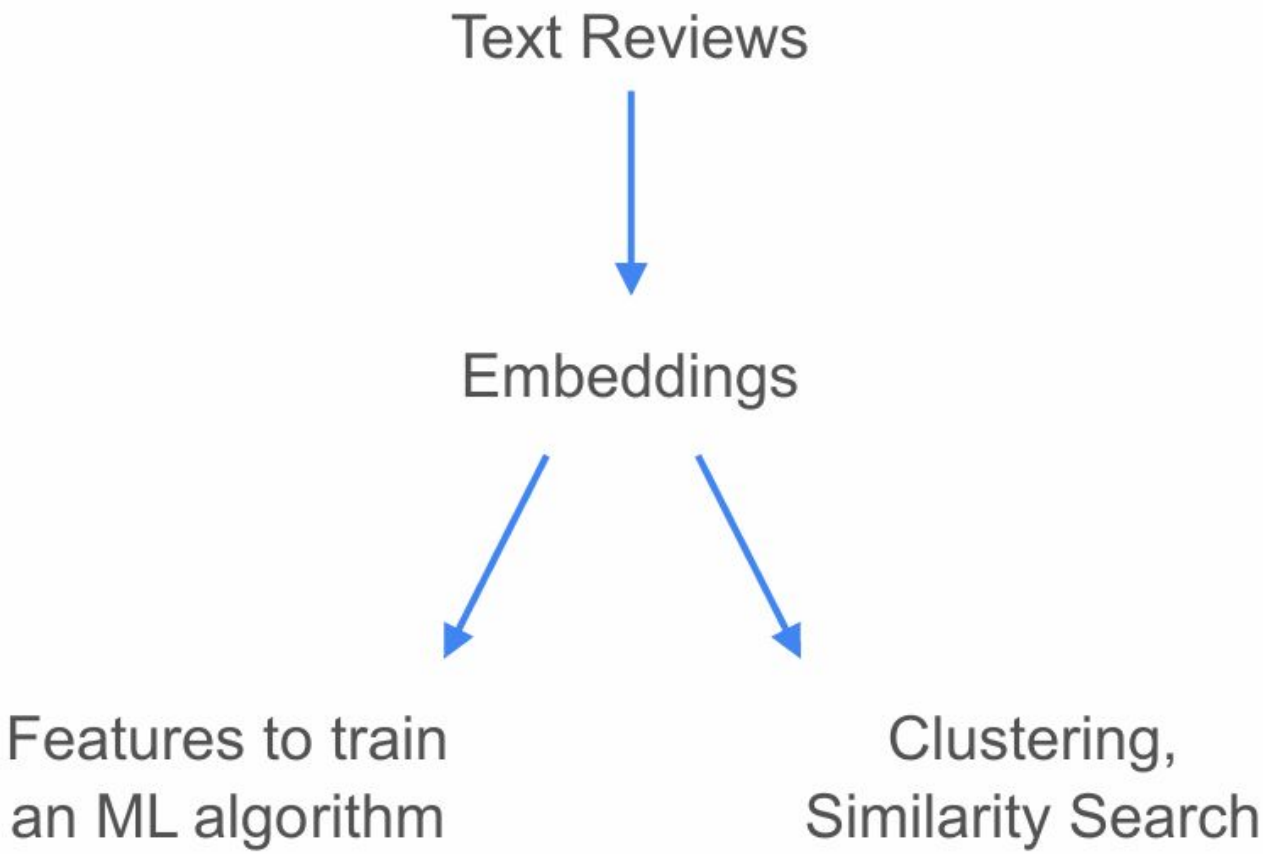
Gemini



Sentence Embedding

- Vector that reflects the semantic meaning of the sentence
- Lower dimension than the vector generated by TF-IDF

Sentence Embeddings



Batch Transformations

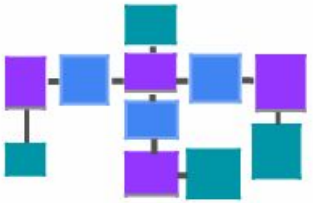
Batch Transformation Patterns and Use Cases

Transformations for Data Modeling

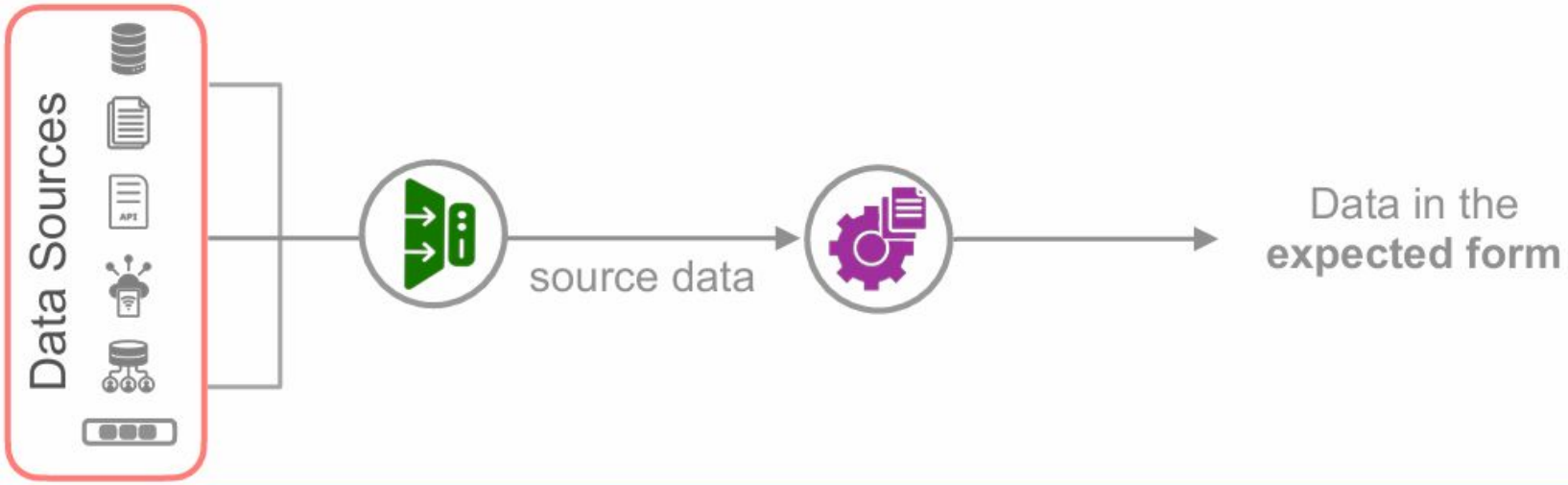
Target Model

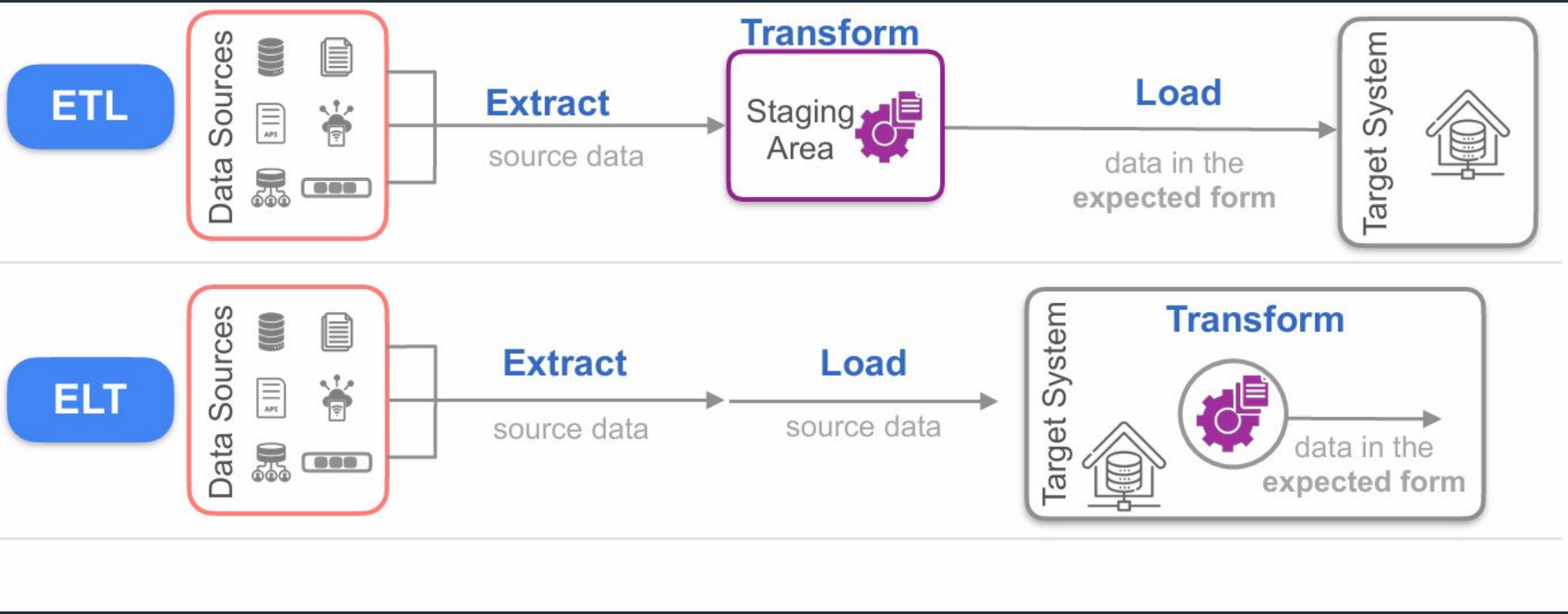
Star schemas

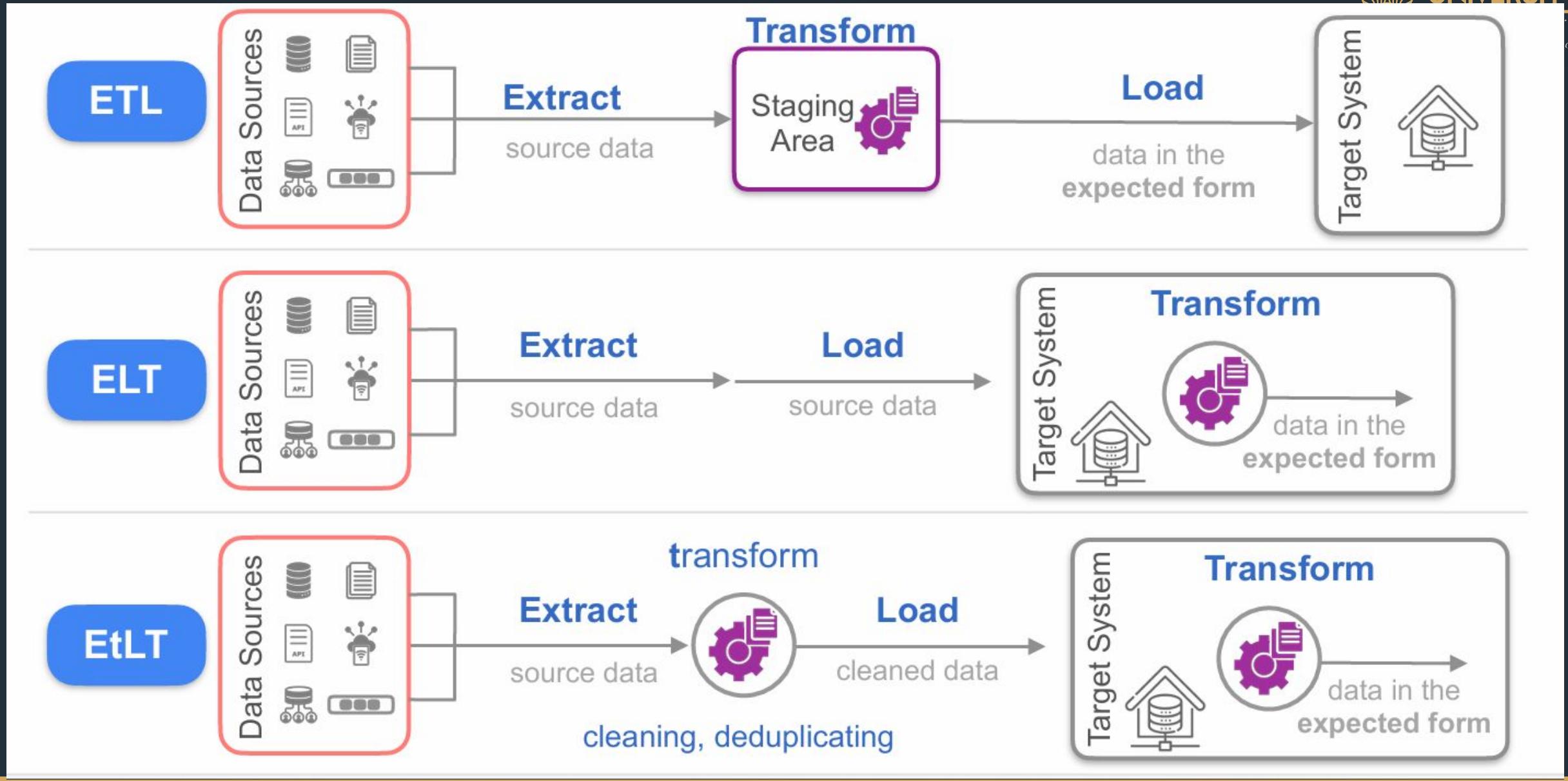
Data Vault



Restructure the source data into the **expected form**



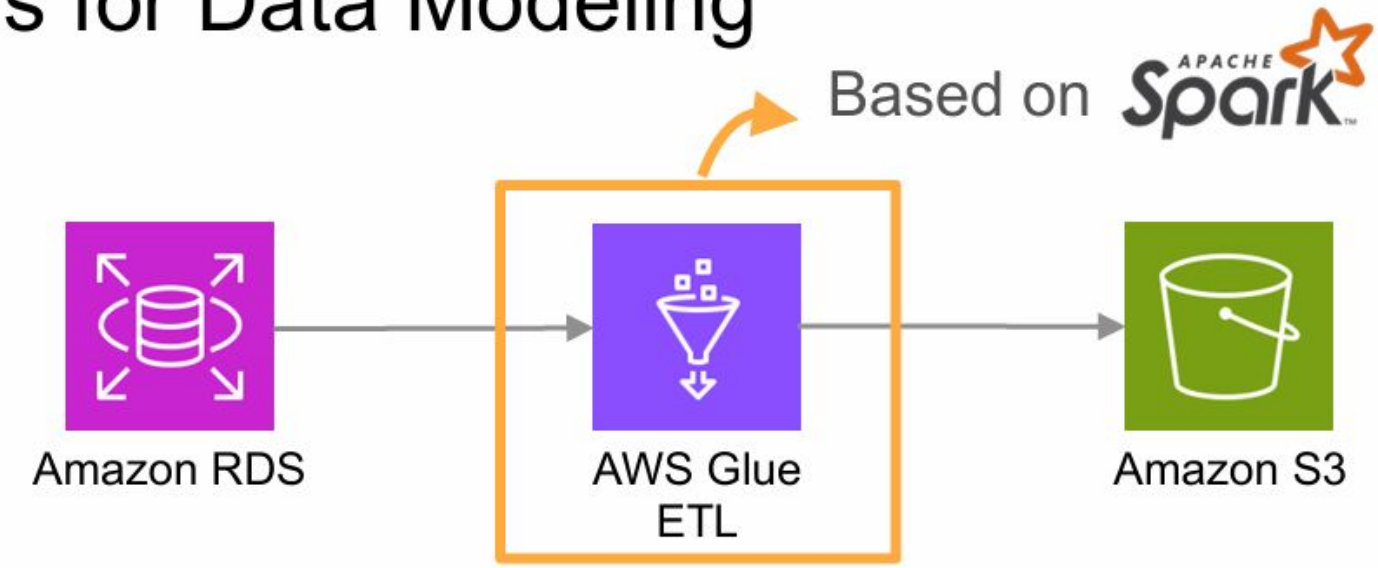

ETL
Data Sources
Extract
source data
Transform
Staging Area
Load
data in the expected form
Target System
ELT
Data Sources
Extract
source data
Load
source data
Target System
Transform
data in the expected form


ETL
Data Sources
Extract
source data
Transform
Staging Area
Load
data in the expected form
Target System
ELT
Data Sources
Extract
source data
Load
source data
Target System
Transform
data in the expected form
EtLT
Data Sources
transform
Extract
source data
Load
cleaned data
cleaning, deduplicating
Target System
Transform
data in the expected form

Transformations for Data Modeling

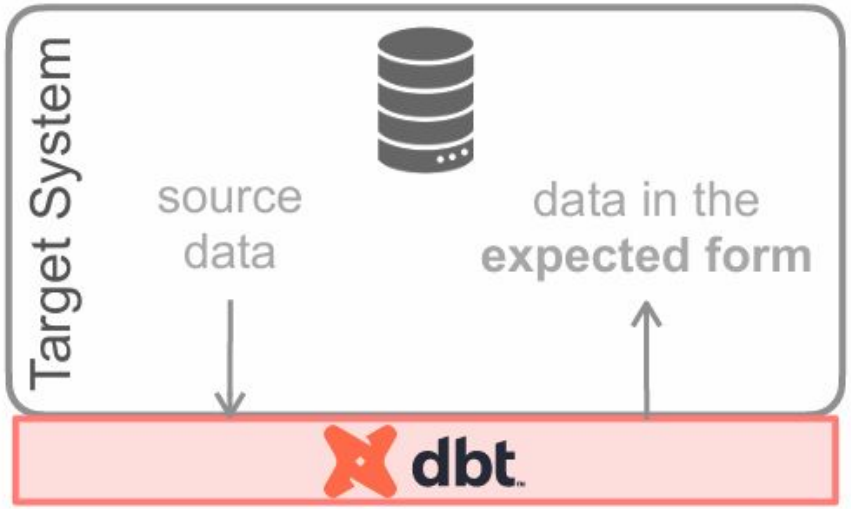
ETL

Course 1



ELT

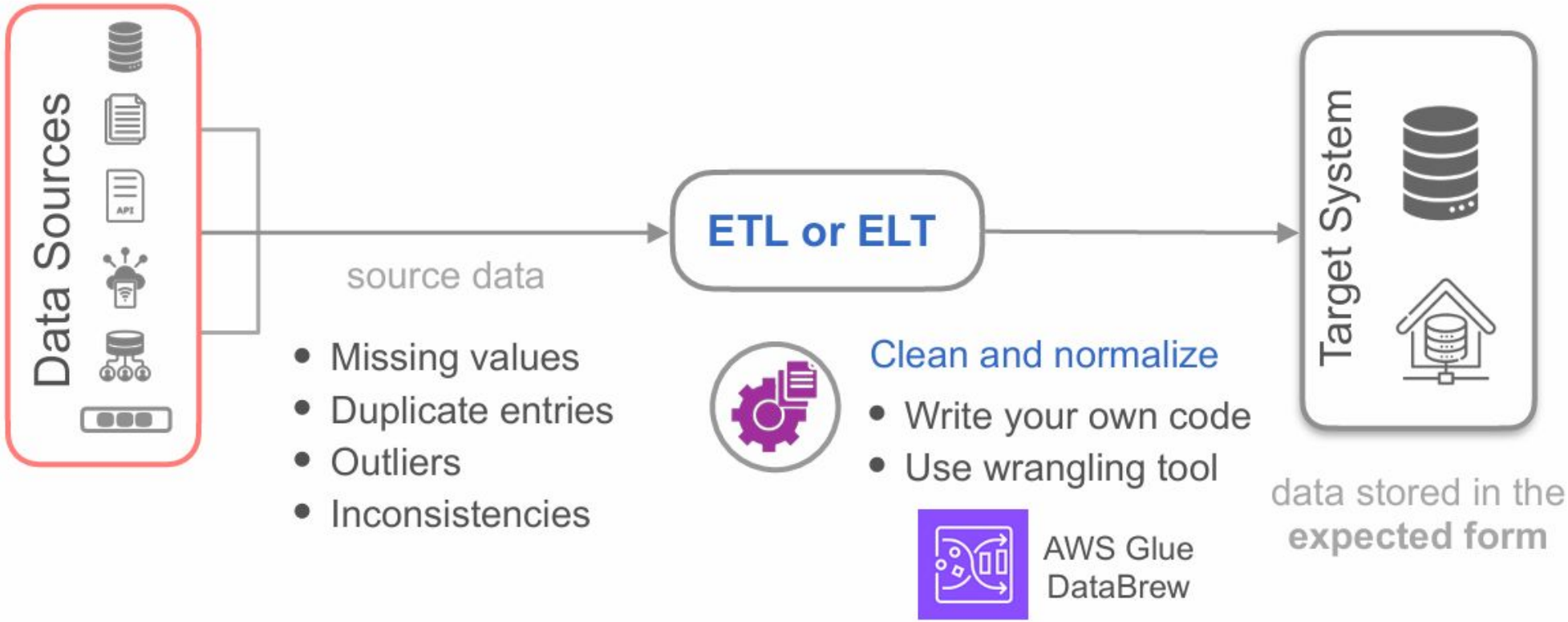
Course 4



- dbt is not an execution tool like Spark
- SQL-tool that facilitates transformation within the database or data warehouse

Transformations for Data Cleaning

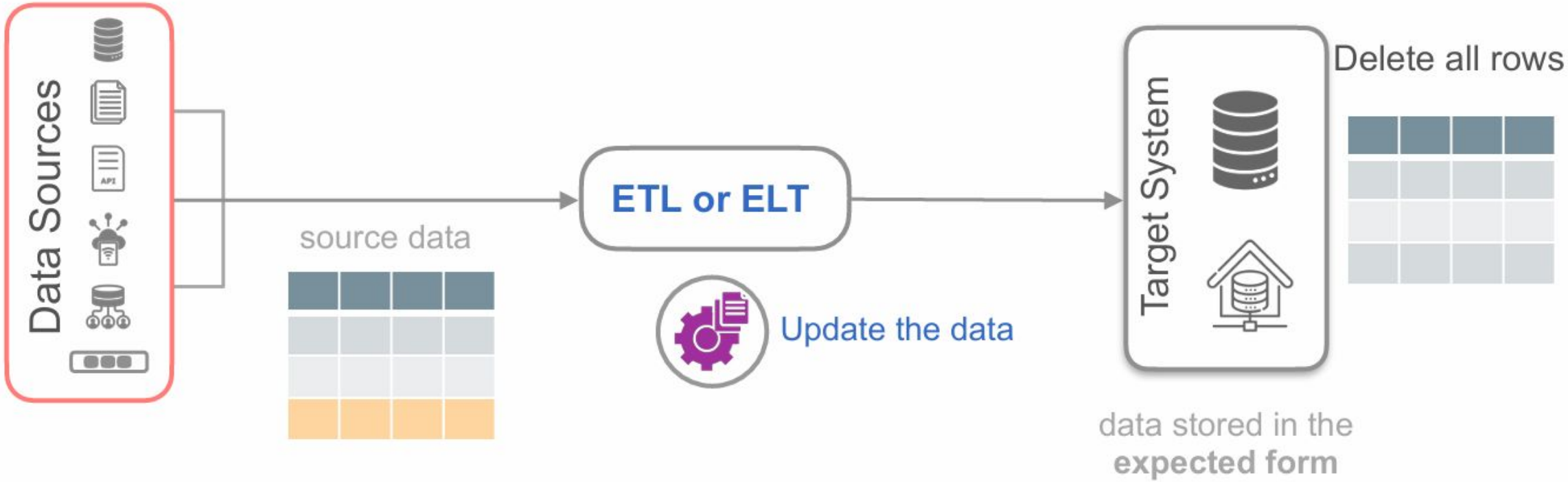
Data Wrangling Take messy, malformed data and turn it into clean data



Transformations for Data Updating

Truncate and Reload

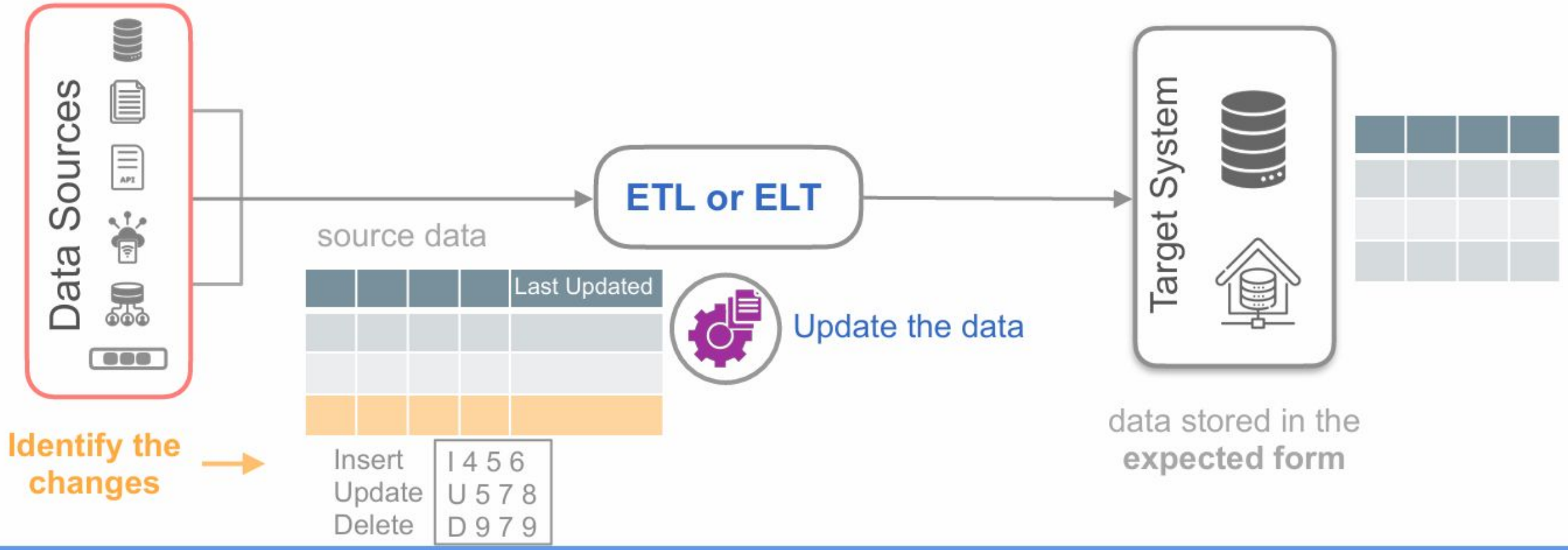
- Delete all records in target system and reload the data from the source
- Have a small dataset
- Update the data once in a while



Transformations for Data Updating

Truncate and Reload

Change Data Capture (CDC)



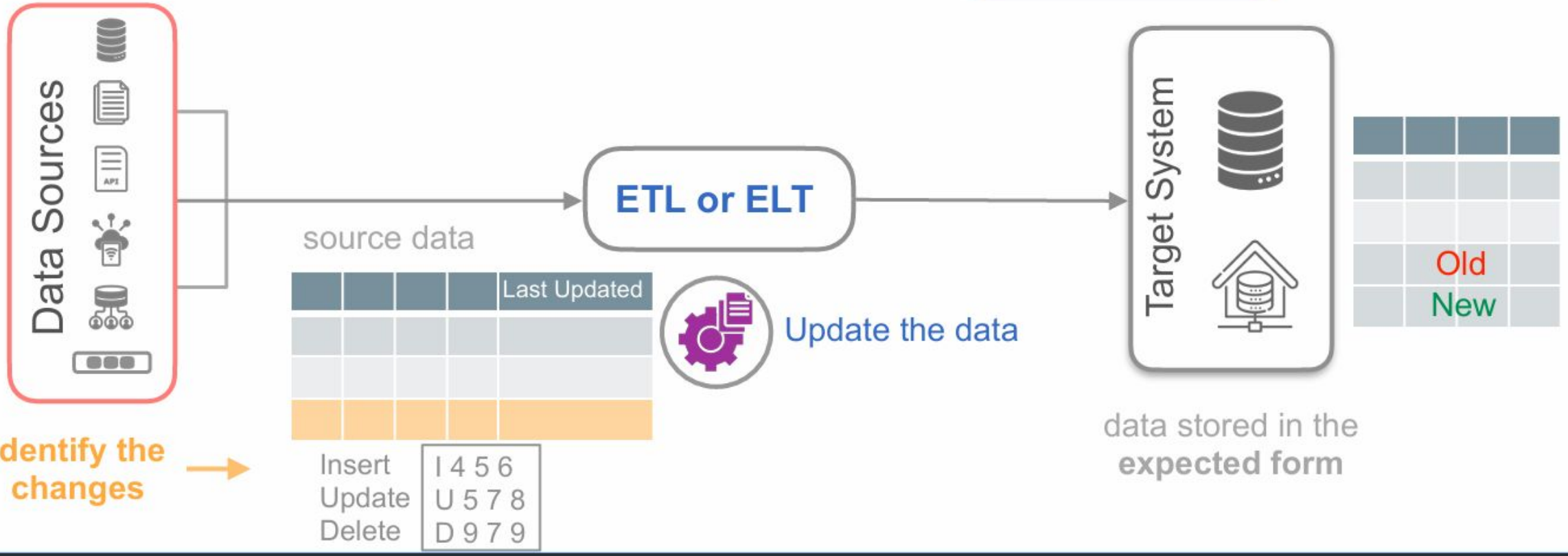
Transformations for Data Updating

Truncate and Reload

Change Data Capture (CDC)

Capture updates

- Insert-only
- Upsert / Merge



Transformations for Data Updating

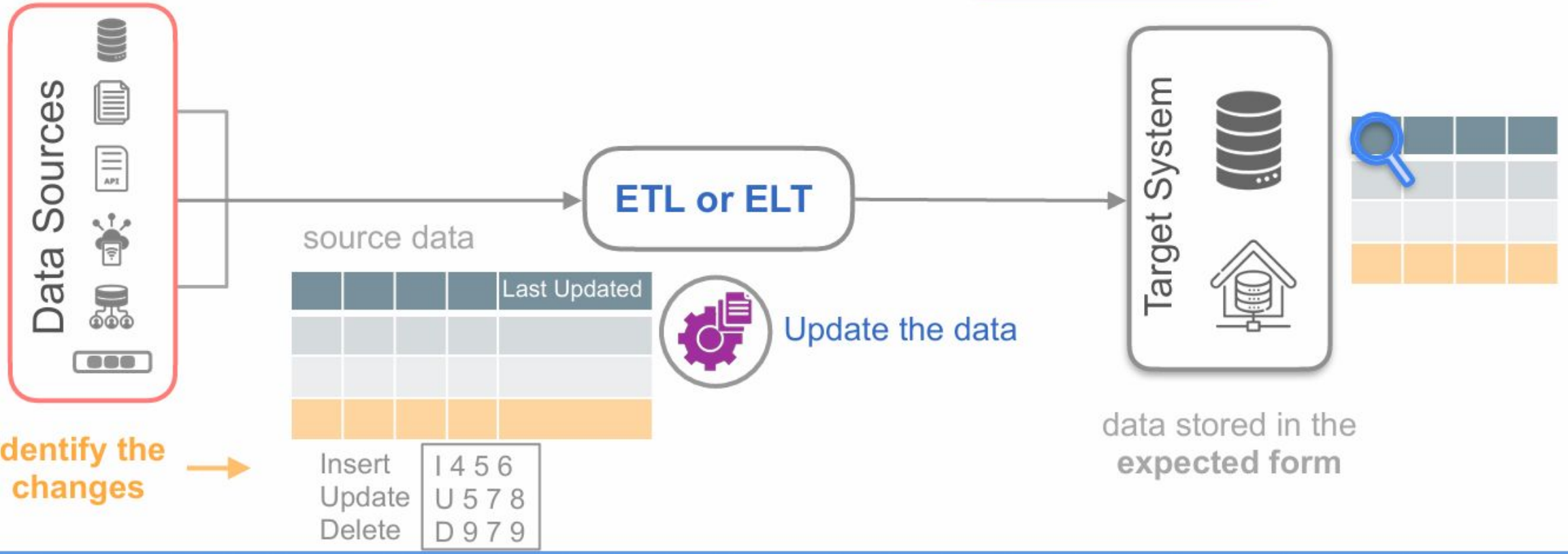
Truncate and Reload

Change Data Capture (CDC)

Capture updates

Insert-only

Upsert / Merge



Transformations for Data Updating

Truncate and Reload

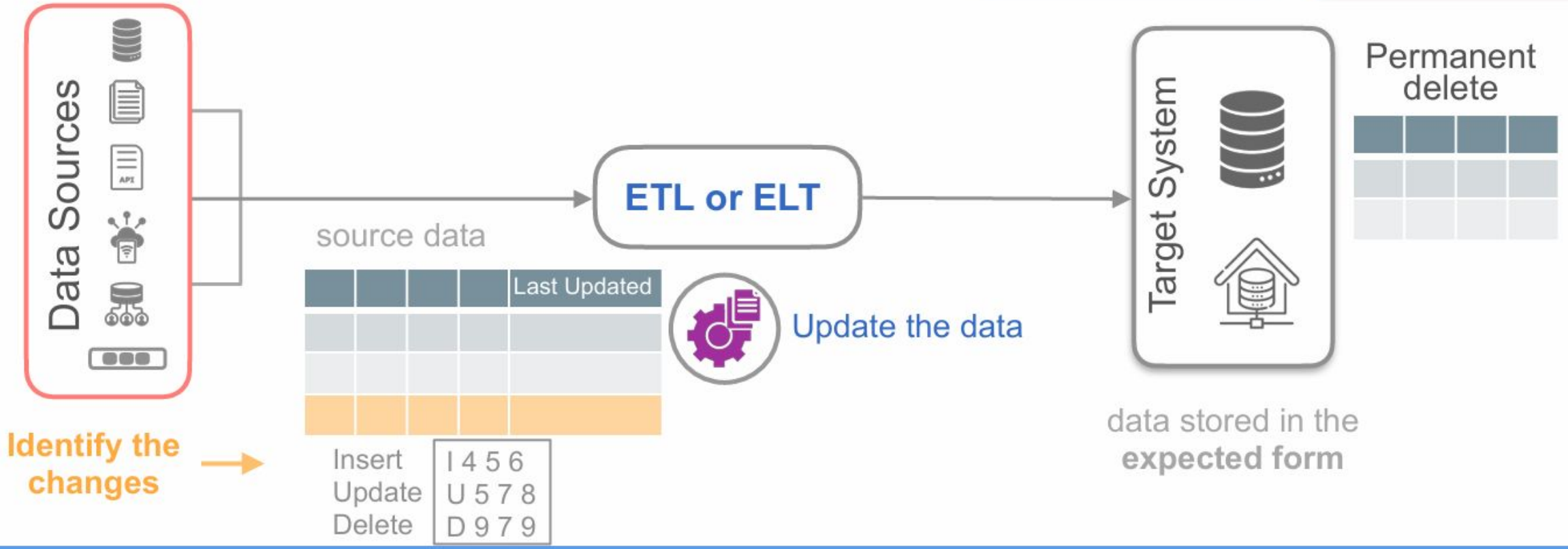
Change Data Capture (CDC)

Capture updates

- Insert-only
- Upsert / Merge

Capture deletes

- Hard
- Soft



Transformations for Data Updating

Truncate and Reload

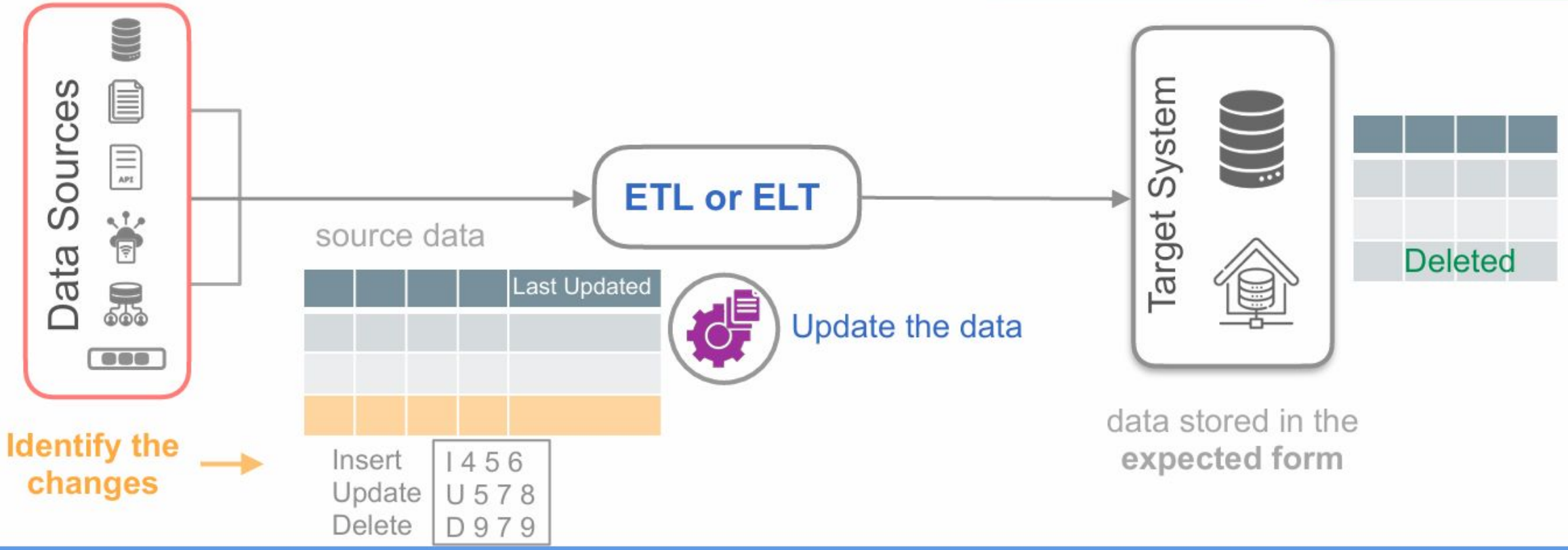
Change Data Capture (CDC)

Capture updates

- Insert-only
- Upsert / Merge

Capture deletes

- Hard
- Soft



Transformations for Data Updating

Truncate and Reload

Change Data Capture (CDC)

Capture updates

Insert-only

Upsert / Merge

Capture deletes

Hard

Soft

Keep all records but be able to filter them out

ETL or ELT

source data

				Last Updated

Update the data

data stored in the **expected form**

Identify the changes

Insert	I 4 5 6
Update	U 5 7 8
Delete	D 9 7 9

Transformations for Data Updating

Truncate and Reload

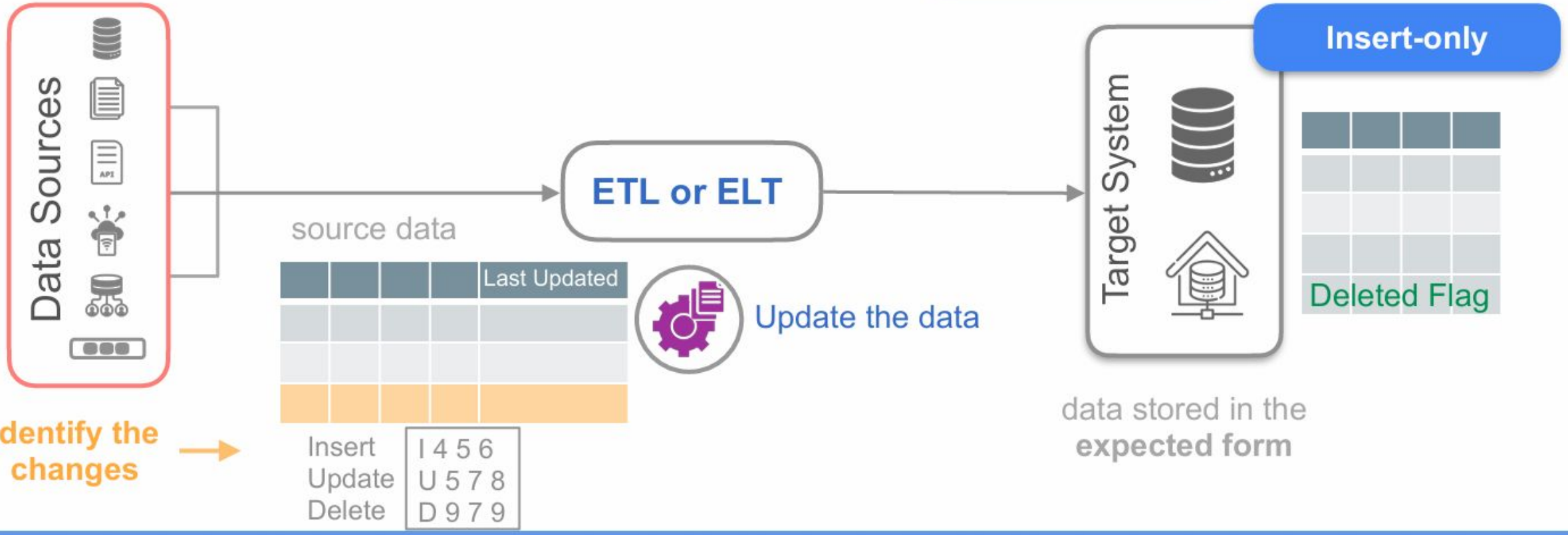
Change Data Capture (CDC)

Capture updates

- Insert-only
- Upsert / Merge

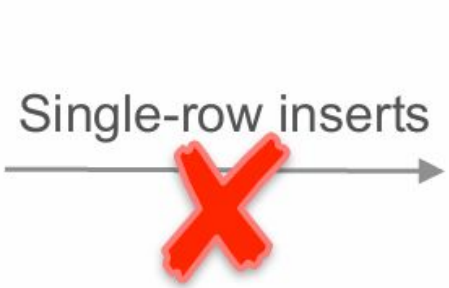
Capture deletes

- Hard
- Soft



Transformations for Data Updating

Row-oriented database



OLAP Column-oriented database

- Puts a massive load on the OLAP system
- Extremely inefficient for subsequent reads

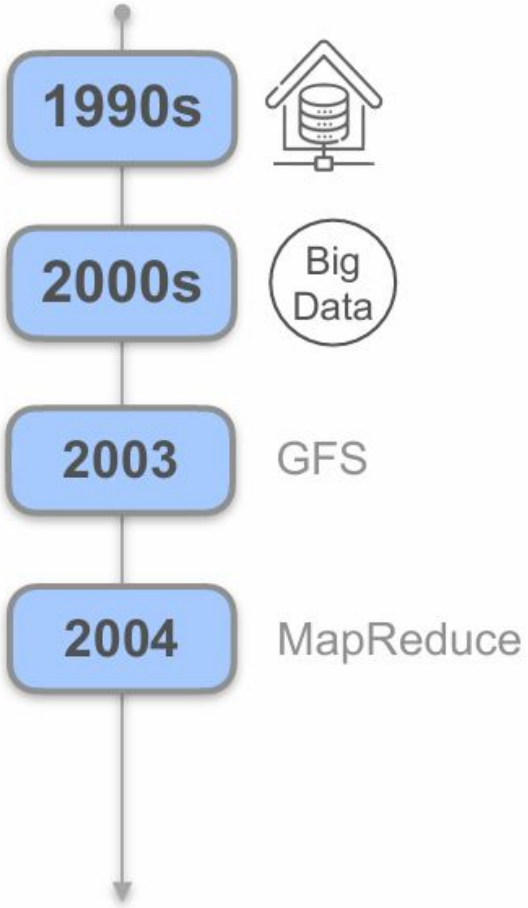
- Organizes data more efficiently
- Leverages distributed parallel processing to speed up data loading

RV UNIVERSITY
Go, change the world
an initiative of RV EDUCATIONAL INSTITUTIONS

Batch Transformations

Distributed Processing Framework — Hadoop MapReduce

Big Data Era



- Traditional monolithic data warehouses were not able to handle massive amounts of data effectively
- Commodity hardware became cheap and ubiquitous
- Several innovations in large-scale distributed storage and computing:

2003

Google

The Google File System

GFS

2004

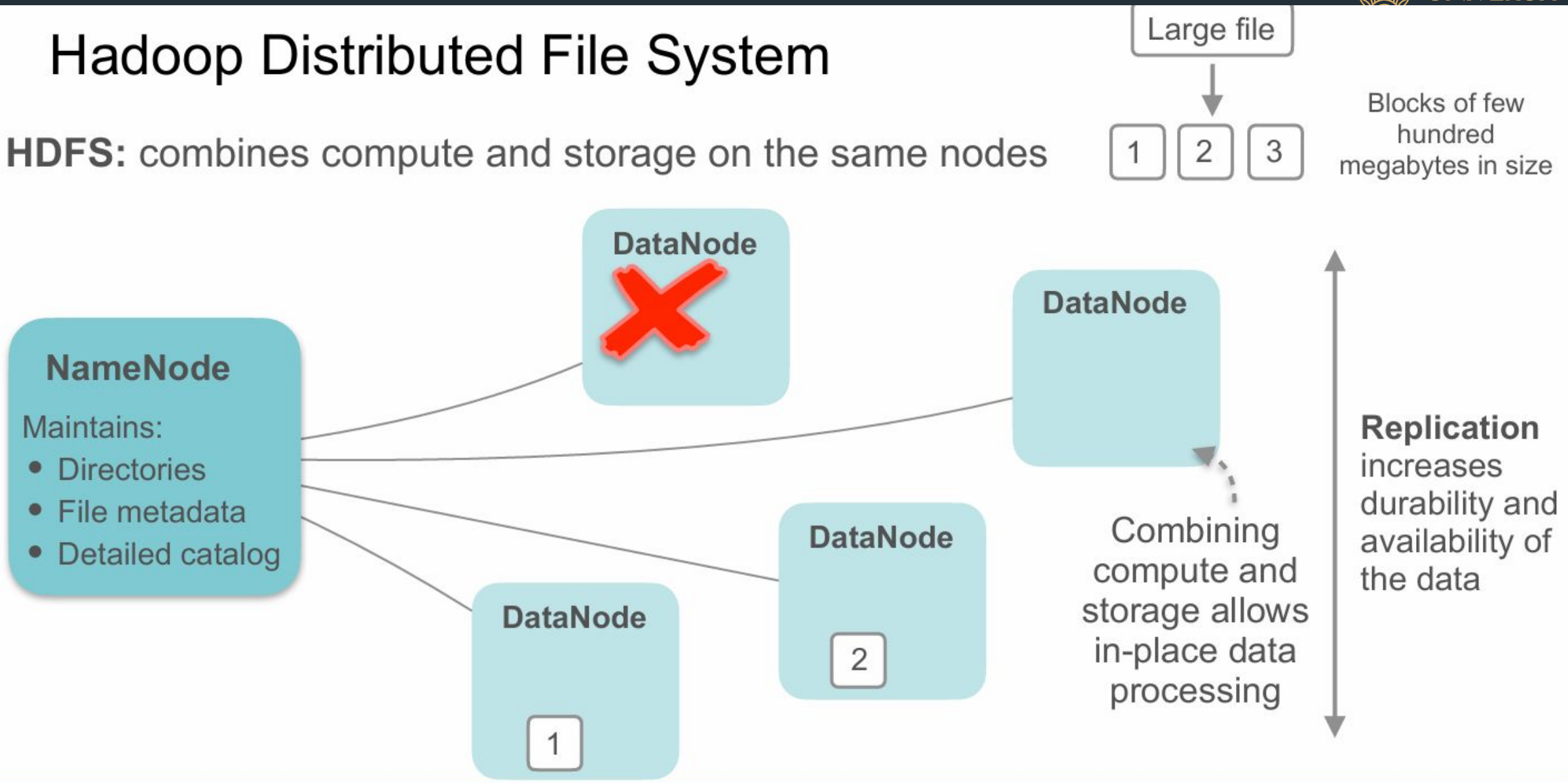
Google

MapReduce

Parallel programming paradigm for processing data distributed over GFS

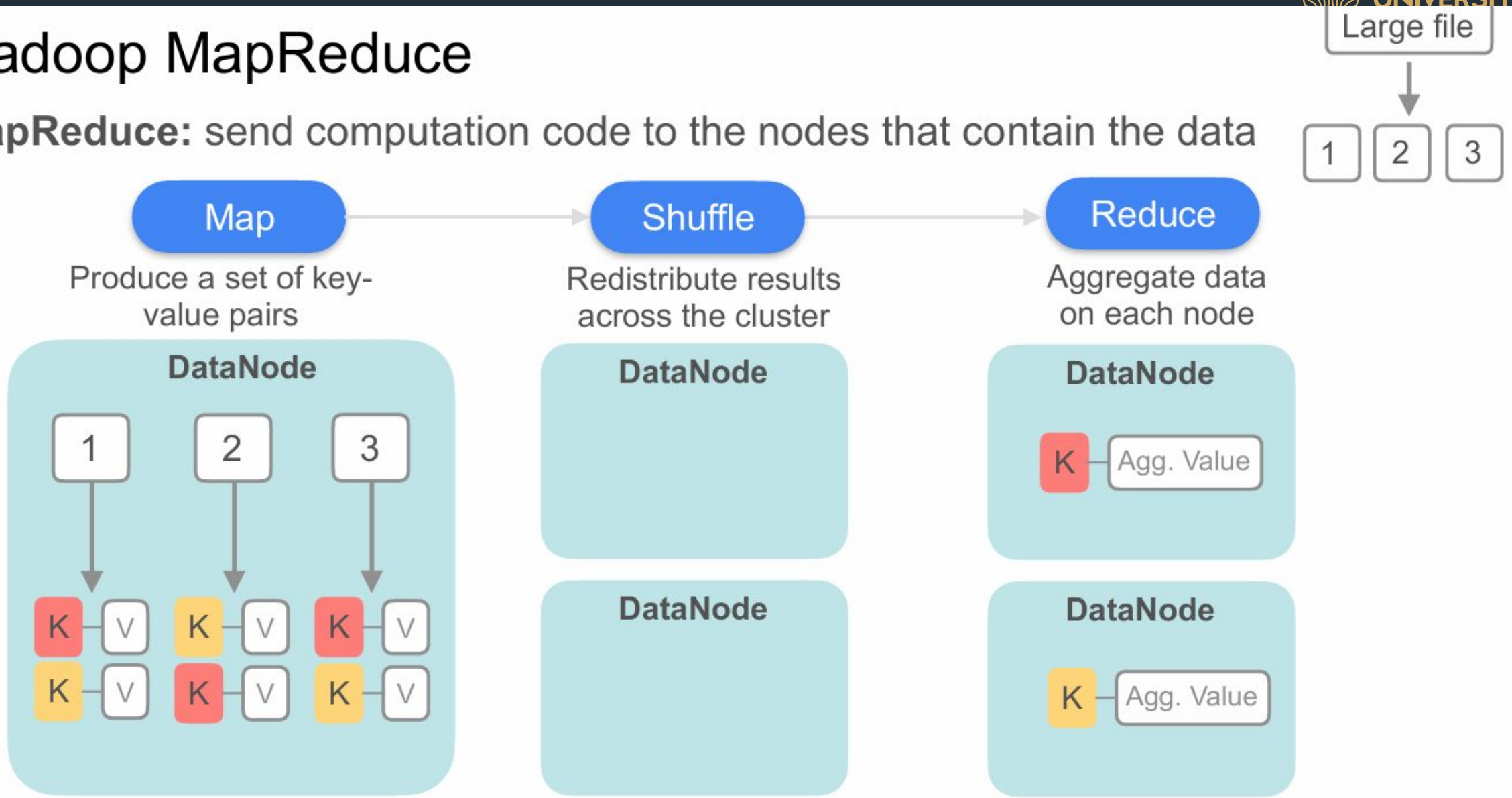
Hadoop Distributed File System

HDFS: combines compute and storage on the same nodes



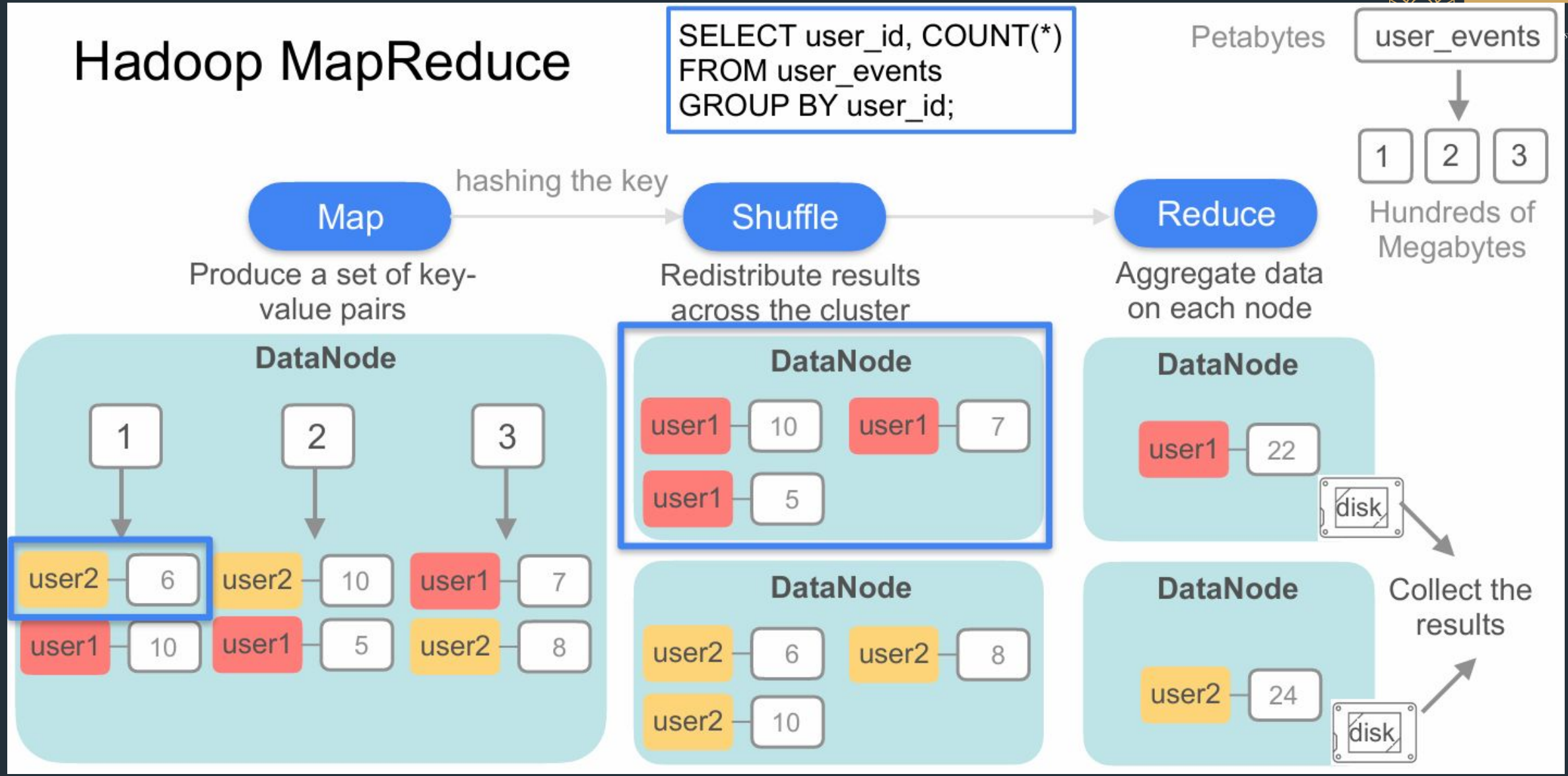
Hadoop MapReduce

MapReduce: send computation code to the nodes that contain the data



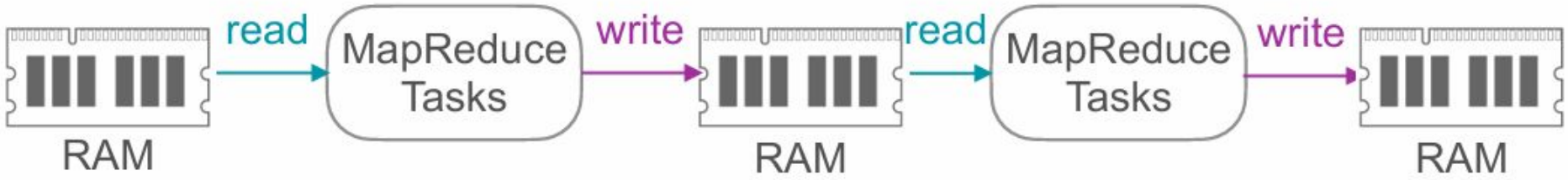
Hadoop MapReduce

```
SELECT user_id, COUNT(*)
FROM user_events
GROUP BY user_id;
```



Hadoop MapReduce - Shortcomings

Pros	Cons
• Simplifies state and workflow management • Minimizes memory consumption	• Drives high-disk bandwidth utilization • Increases processing time



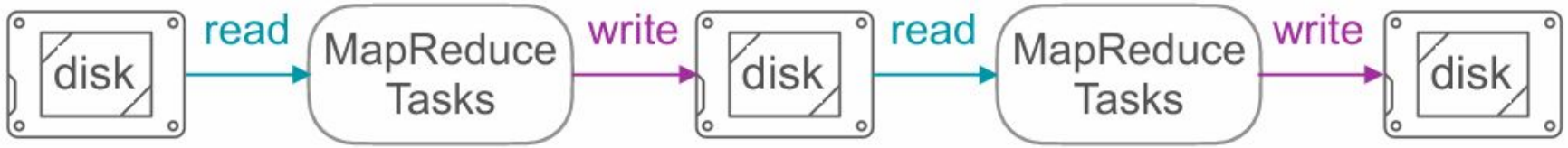
- In-memory processing speeds up data processing tasks
- Disk is treated as a second-class data storage layer

Batch Transformations

Distributed Processing Framework — Spark

Distributed processing framework

Integrate compute and persistent storage via the HDFS.



Computing engine designed for processing large datasets.

- Retains intermediate results in memory
- Limits disk I/O interactions, enabling faster computation

Separate persistent storage systems

Locally

Data centers

Cloud

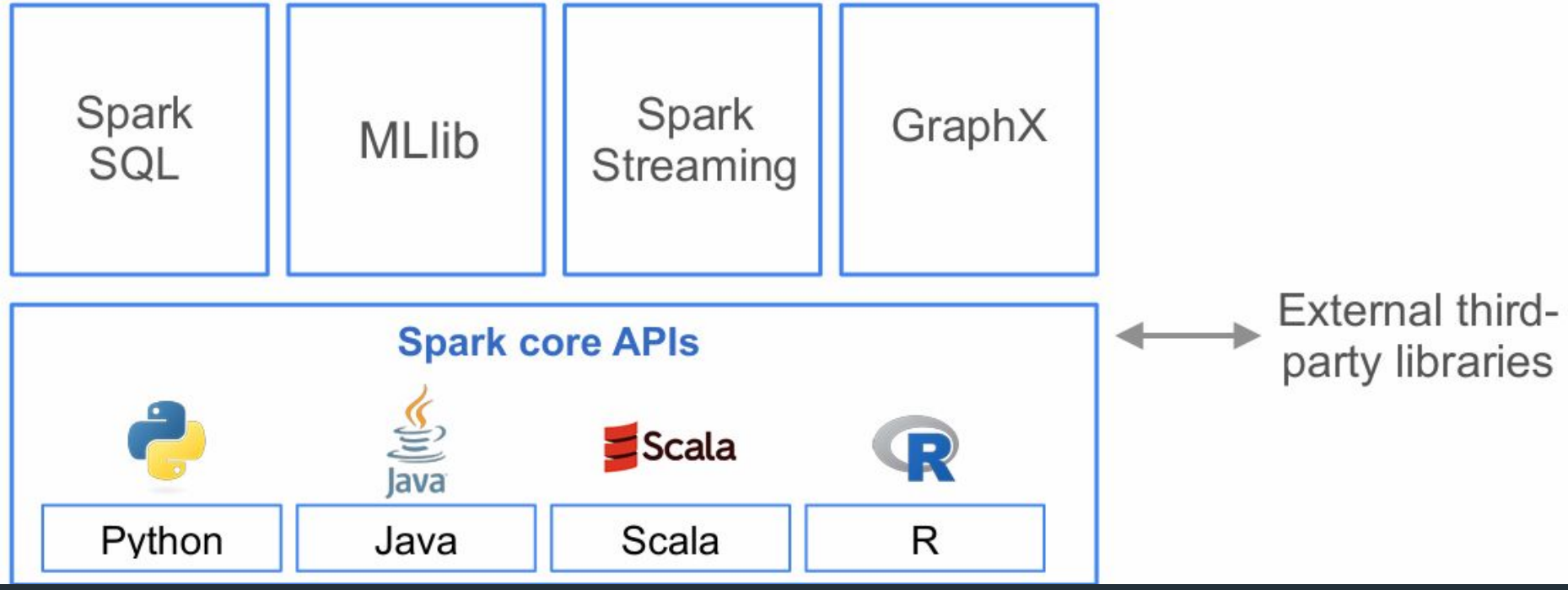
HDFS

Spark

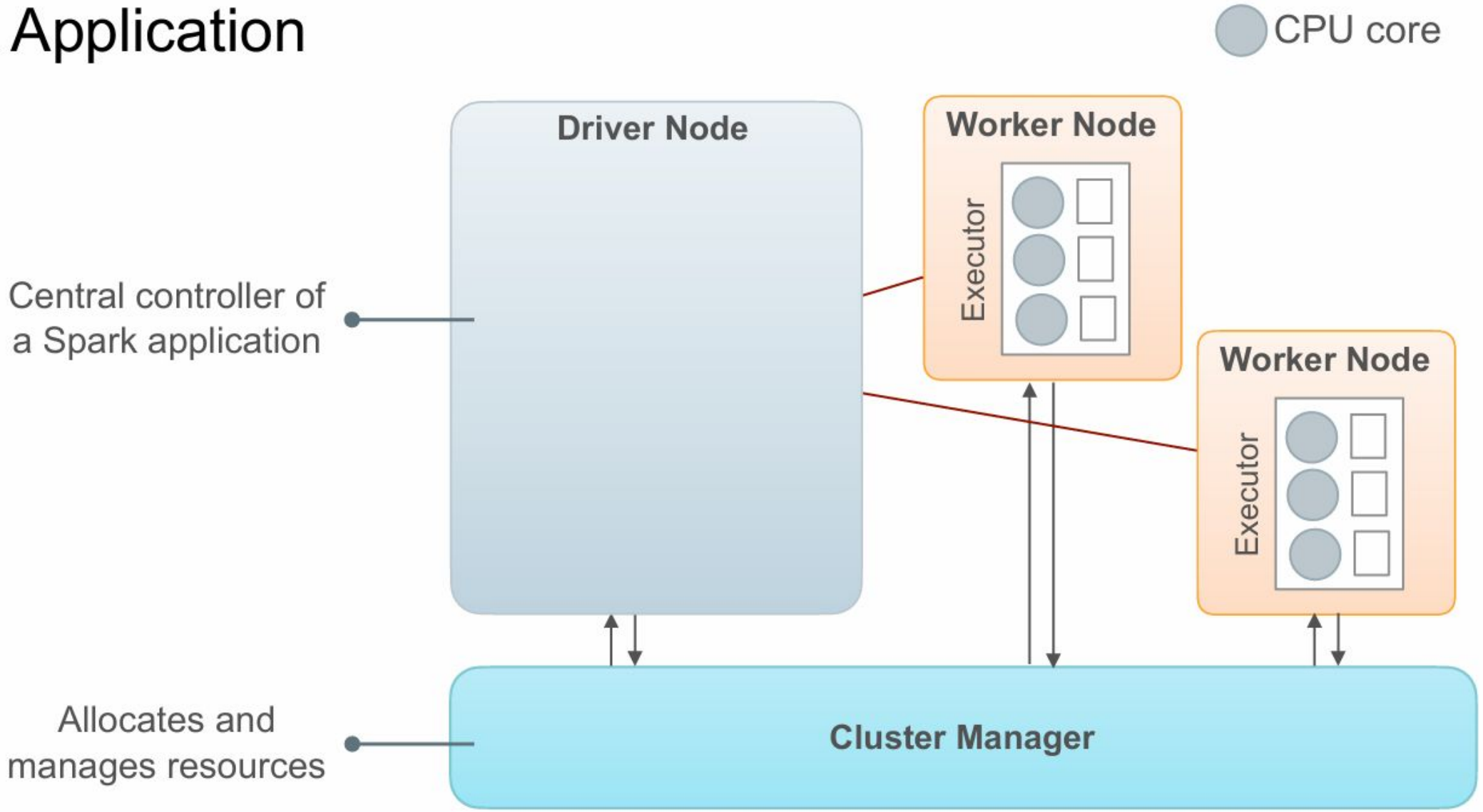
Unified Platform

You can run different types of analytical workloads:

- Perform SQL queries
- Train and test ML algorithms
- Apply streaming transformations



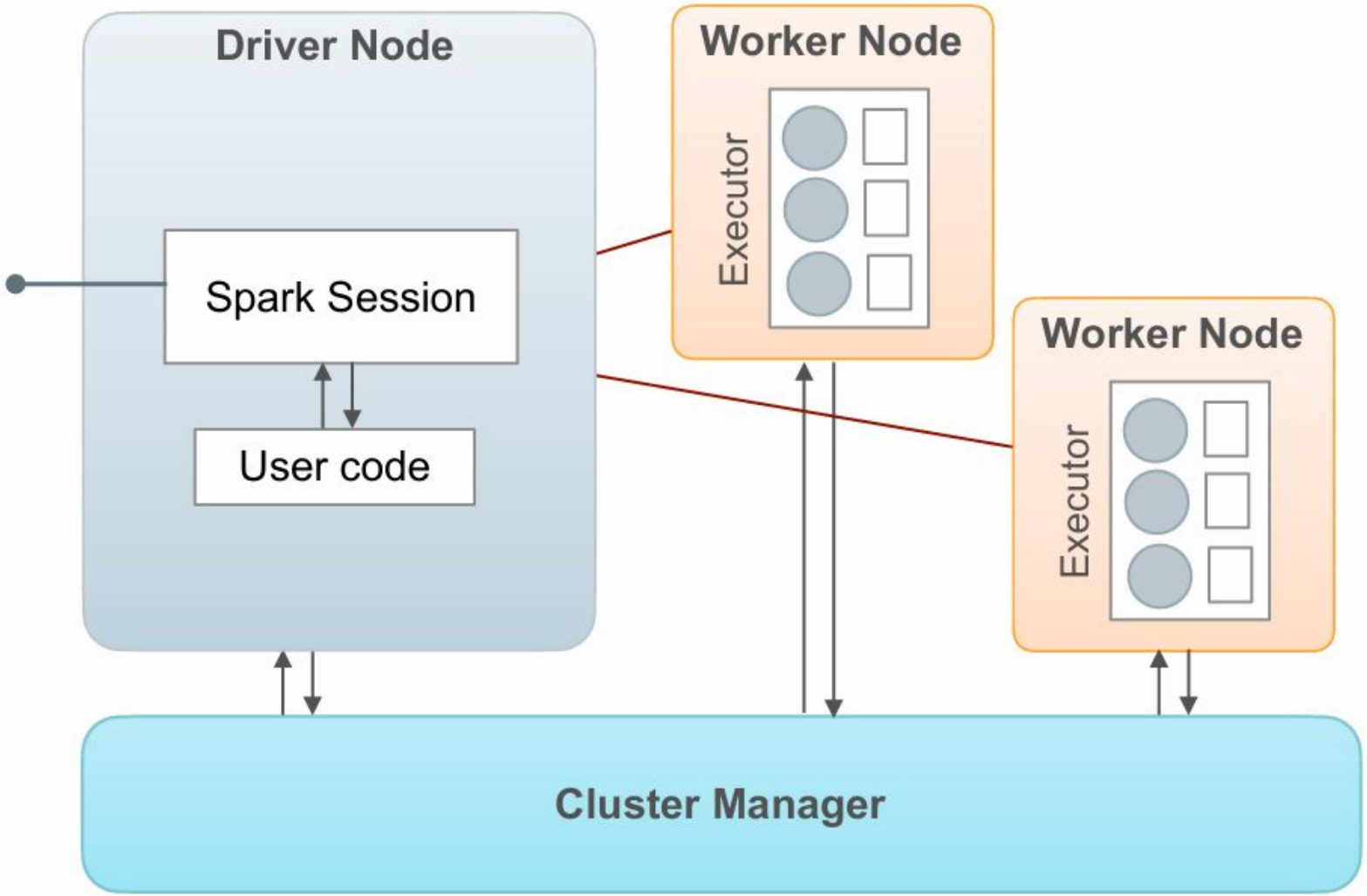
Spark Application



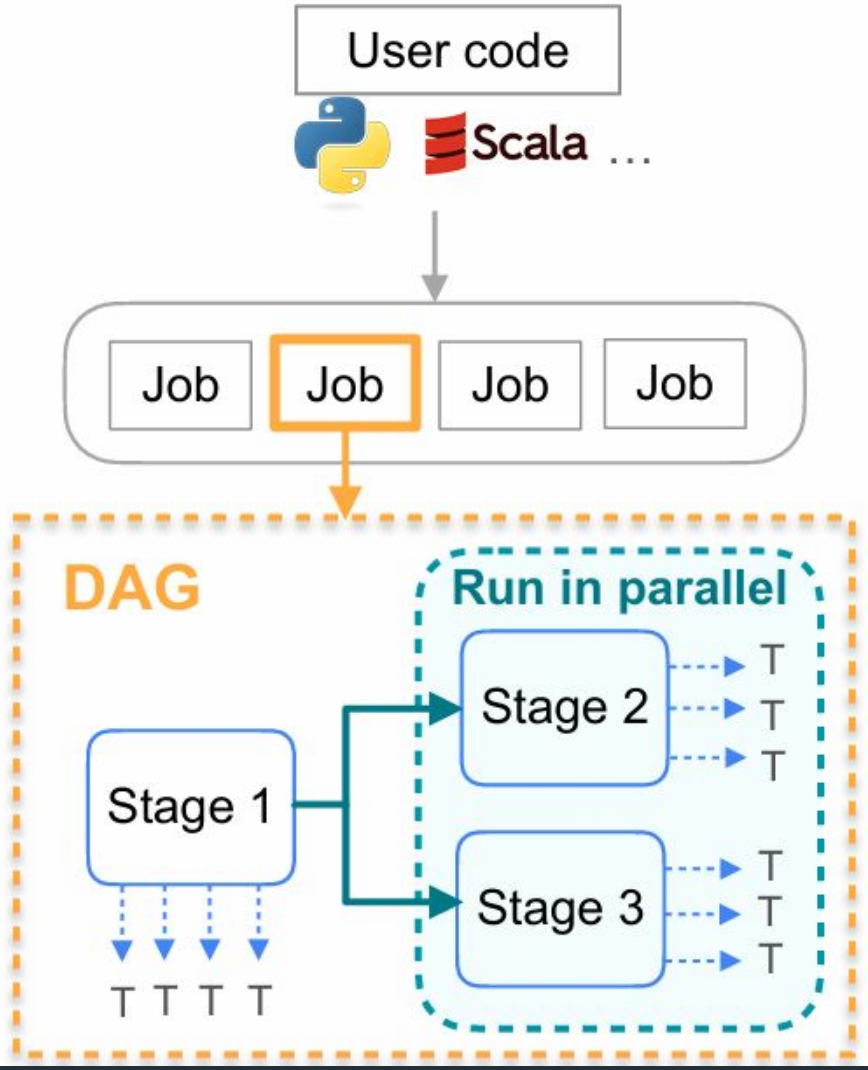
Spark Application

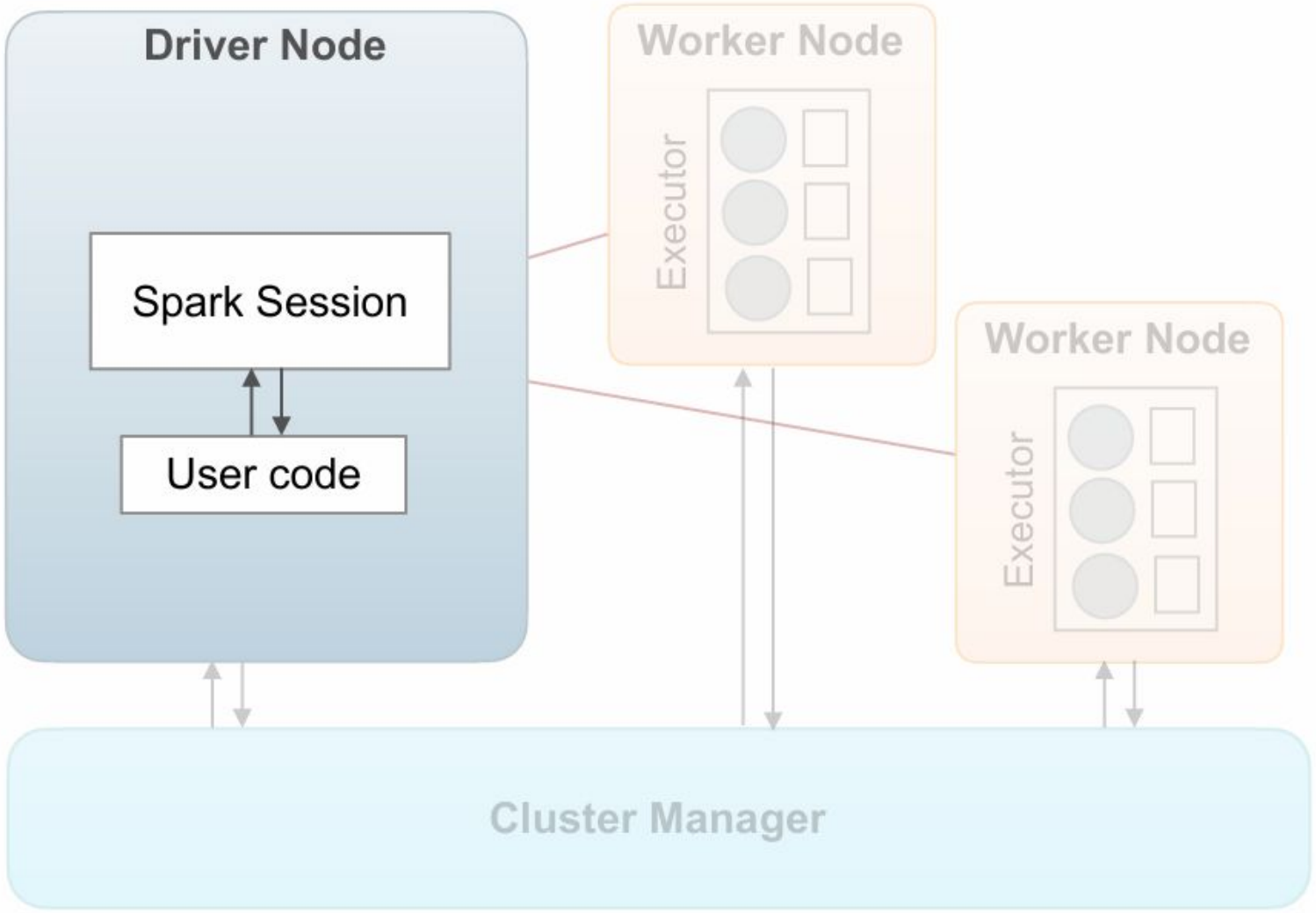
Single unified entry point to Spark's functionality:

- Define Dataframes
- Read data from sources
- Perform SQL queries

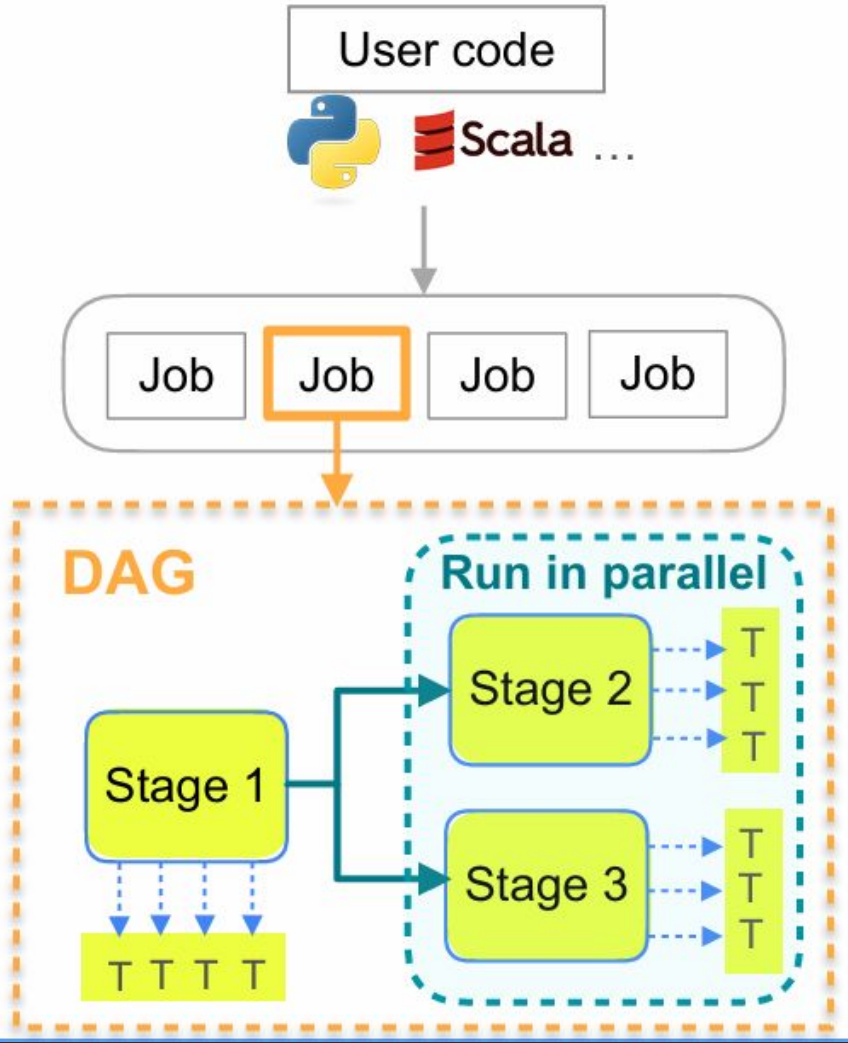


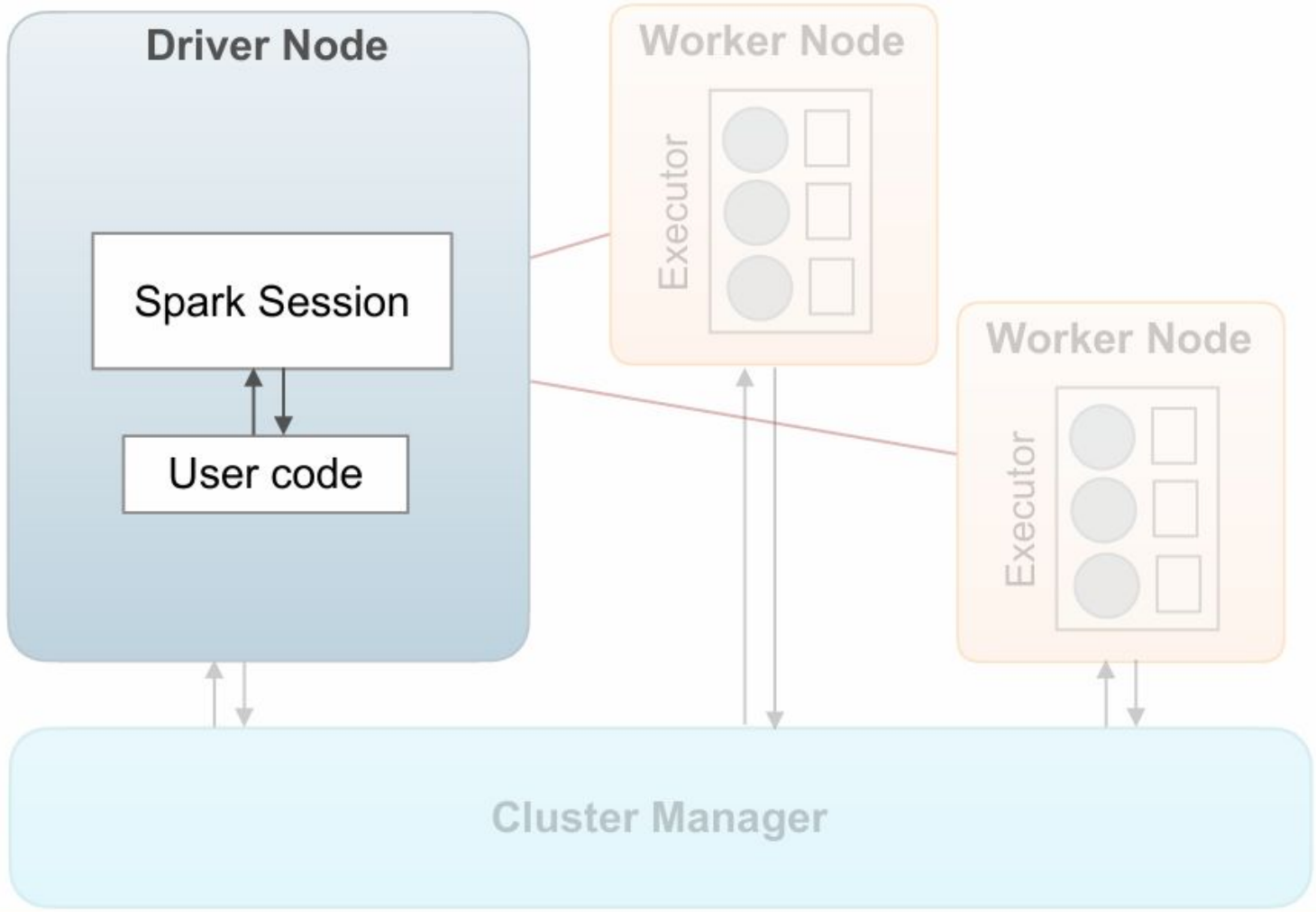
Spark Application



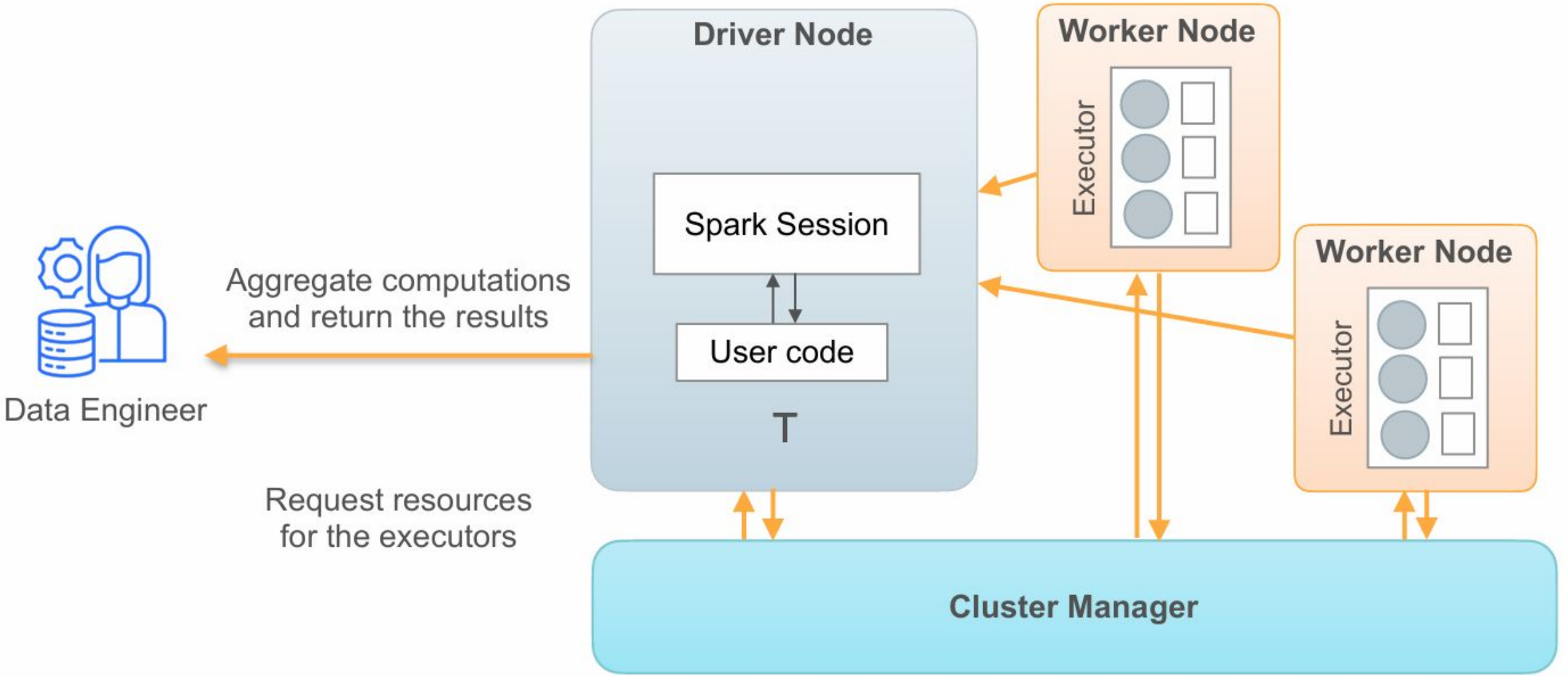


Spark Application





Spark Application



Batch Transformations

Spark DataFrames

Spark Dataframes

High-level Structured API

Spark DataFrames

← Use simpler and more expressive operations:
Filtering, selecting, counting, aggregating and grouping

Immutable data structures

Low-level API

Resilient Distributed Dataset (RDD)

actual partitioned collection of records

← Need to manually define and optimize operations

Spark Dataframes

High-level Structured API

Spark DataFrames

Use simpler and more expressive operations:
Filtering, selecting, counting, aggregating and grouping

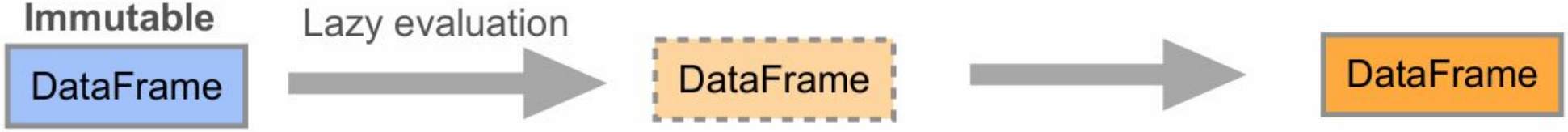
Transformation

Filtering, selecting, joining and grouping

Action

Counting, showing, and saving

Immutable



- Transformations: recorded as a lineage, executed when an action is invoked
- Allows Spark to optimize the execution plan
- Lineage & immutability: fault-tolerance of DataFrames

When to use Spark DataFrames?

Distributed Framework: load the entire data into a cluster of nodes

Use Spark: if data doesn't fit entirely into memory or you want to leverage distributed computations

Non-distributed Framework: load the entire data into the memory

Use pandas: if data can fit into the memory

Best Practices

- Extract only the data you need from the source
- Apply transformation inside the source database to reduce the size of the ingested data

RV UNIVERSITY
Go, change the world
an initiative of RV EDUCATIONAL INSTITUTIONS

Serving Data for Analytics and Machine Learning

Serving Data — Analytics Use Cases

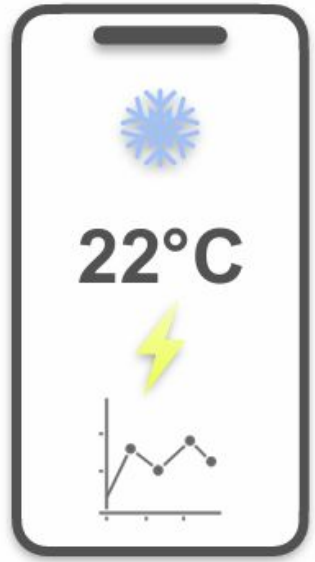
Business Intelligence

Operational Analytics

Monitor data to inform immediate action

Serve data within the required latency

Embedded Analytics



Serving Data — Machine Learning Use Cases

Becoming widely adopted in many companies

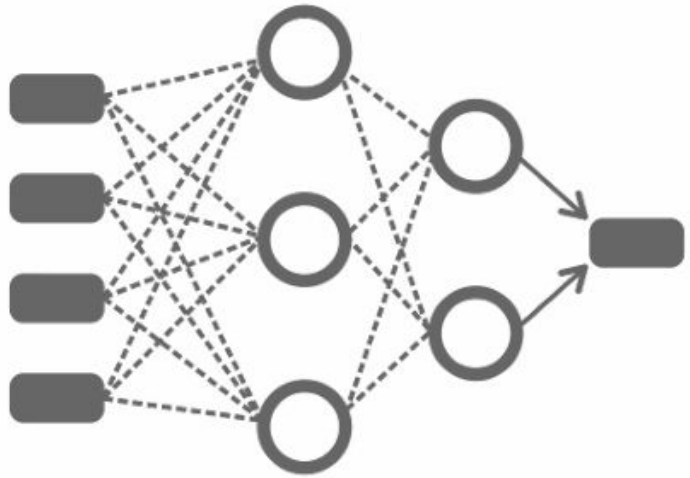
Data Engineer

Data Scientist

Machine Learning Engineer

Model training

- FEATURE 1
- FEATURE 2
- FEATURE 3
- FEATURE 4



Example:

- Customer churn model
- Recommendation system

Serving Data

Stakeholders

Semantic Layer

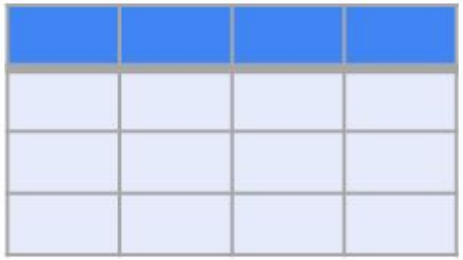
- Document definitions (e.g. meaning of an active user)
- Derive business metrics (e.g. revenue)
- Create a common language for data

Data Sources

Data Model

You can serve data as…

A table



View

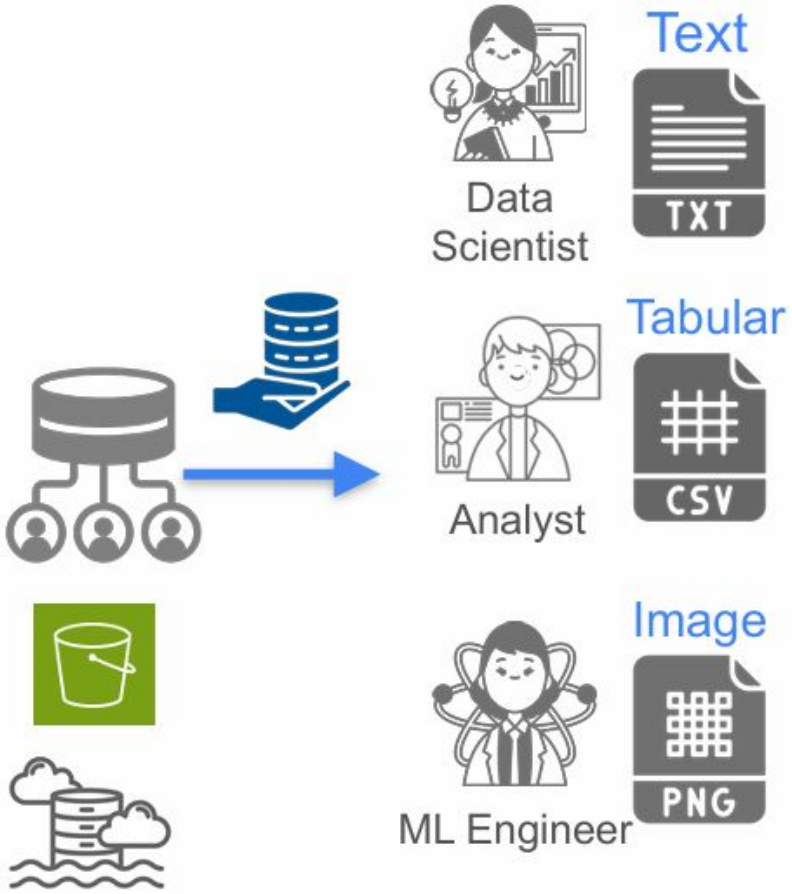
Materialized View

RV UNIVERSITY
Go, change the world
an initiative of RV EDUCATIONAL INSTITUTIONS

Serving Data

As Files

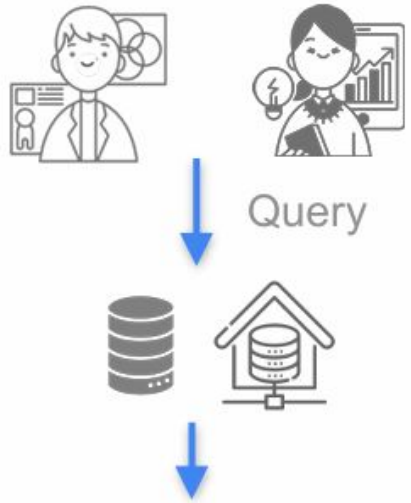
For ad hoc requests



As Files

For ad hoc requests

From Databases and Data Warehouses

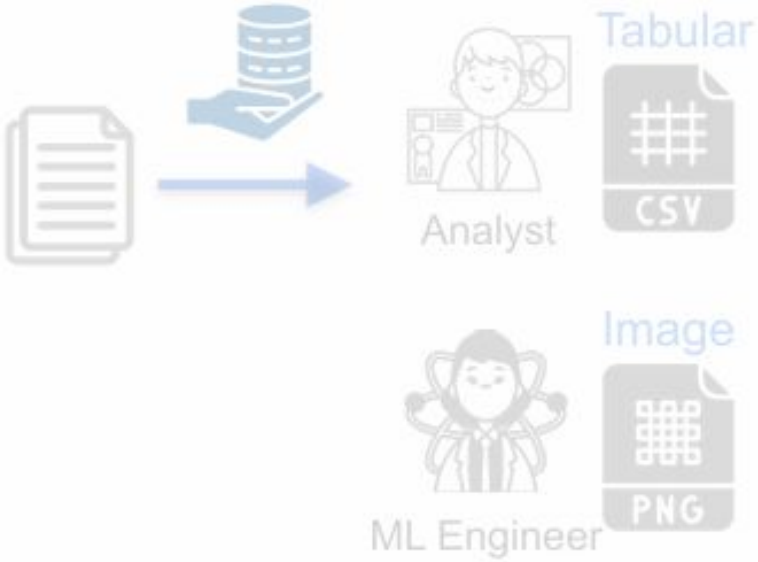


Benefits:

- Imposes order & structure through schema
- Gives you fine-grained permissions controls
- Offers high performance for queries

As Files

For ad hoc requests

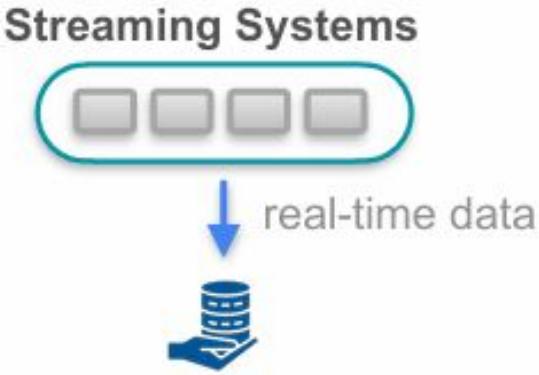


From Databases and Data Warehouses

Benefits:

- Imposes order & structure through a schema
- Gives you fine-grained permissions controls
- Offers high performance for queries

From Streaming Systems



Example

Operational analytics database

- Enables low latency analytical queries across historical and current data
- Effectively combining the features of an OLAP database with a stream-processing system

Data Management

Data Engineer

Trust the data

Can interpret the data correctly

Stakeholders

Ensuring data correctness, consistency, trustworthiness

Data Definition

Meaning of data as it is understood throughout the organization
e.g. document and make available the definition of “Customer”

Data Logic

Consists of formulas for deriving metrics from data
e.g. gross sales or customer lifetime value

Customer Churn Metric → Understand the definition of “Customer" → Write SQL query → Define this metric so it can be reused

Semantic Layer

- Translate the underlying data elements and structures into more intuitive business terms
- Ensures consistent definitions for business terms
- Helps end users more easily navigate the data

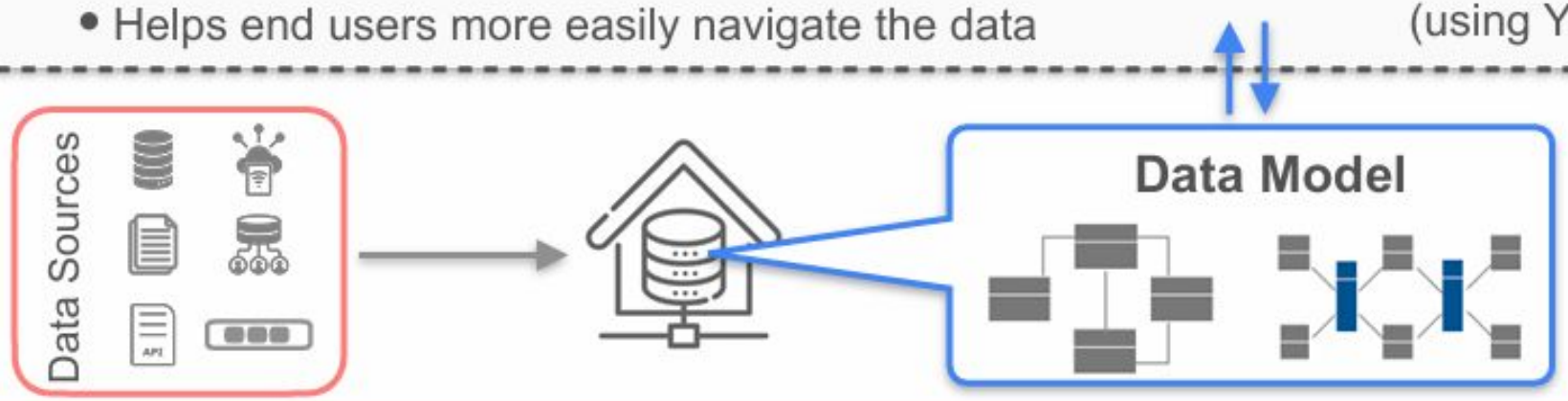
Created in a BI tool

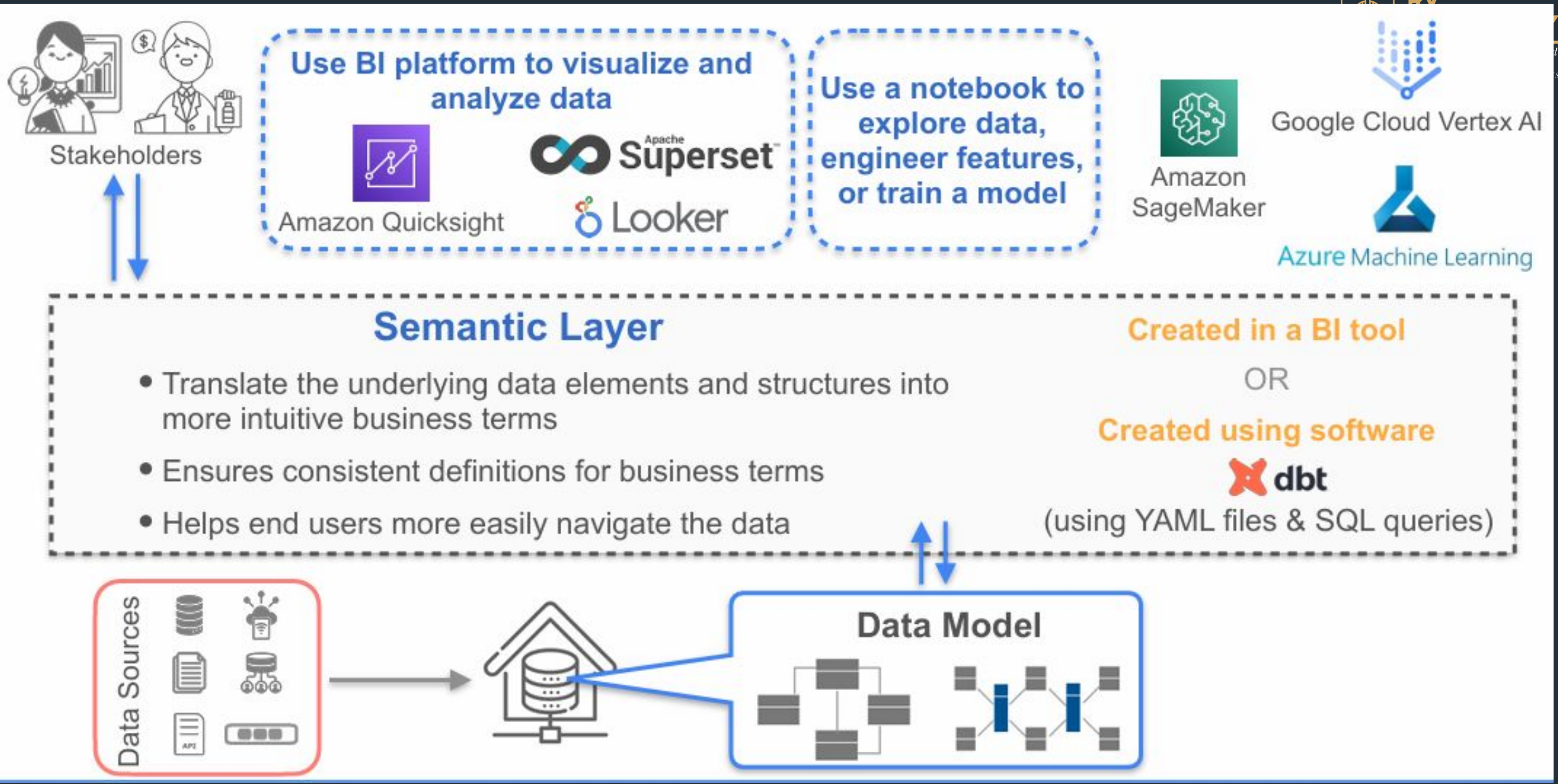
OR

Created using software

dbt

(using YAML files & SQL queries)




Stakeholders
Use BI platform to visualize and analyze data
Amazon Quicksight
Apache Superset
Looker
Use a notebook to explore data, engineer features, or train a model
Amazon SageMaker
Google Cloud Vertex AI
Azure Machine Learning
Semantic Layer
Translate the underlying data elements and structures into more intuitive business terms
Ensures consistent definitions for business terms
Helps end users more easily navigate the data
Created in a BI tool
OR
Created using software
dbt
(using YAML files & SQL queries)
Data Sources
Data Model

RV UNIVERSITY
Go, change the world
an initiative of RV EDUCATIONAL INSTITUTIONS

Views and Materialized Views

View

- Stored in the database to provide easier access to common queries
- Represents a virtual table, not a physical one
- When selecting from a view:
  - The database creates a new query
  - The query optimizer optimizes and runs the query

```
CREATE VIEW  customer_info AS
SELECT
      first_name,
      last_name,
      email,
      Phone,
      city,
      postal_code,
      country
FROM
      customer
      JOIN address ON address.id = customer.address_id
      JOIN city ON city.id = address.city_id
      JOIN country ON country.id = address.country_id
```

Effectively restricting data access to only the data needed

Marketing Analyst

query

Wide Table:
customer, address, city, country

View

- Stored in the database to provide easier access to common queries
- Represents a virtual table, not a physical one
- When selecting from a view:
  - The database creates a new query
  - The query optimizer optimizes and runs the query
- Database object
- Stored and persists on disk

Common Table Expressions (CTE)

- Represents temporary results that you can reference
- Only exists within scope of the main query where they are referenced

```
WITH name of the CTE AS (
Query),

Subsequent query
```

View

- Stored in the database to provide easier access to common queries
- Represents a virtual table, not a physical one
- When selecting from a view:
  - The database creates a new query
  - The query optimizer optimizes and runs the query
- Database object
- Stored and persists on disk

Can be expensive for frequently-run complex queries

Common Table Expressions (CTE)

- Represents temporary results that you can reference
- Only exists within scope of the main query where they are referenced

Materialized Views

- Does some or all of the view computations in advance
- Caches the query results and allows you to refresh the data
- Some latency between refreshes

```
CREATE MATERIALIZED VIEW  rental_by_category AS
SELECT category.name AS category,
       sum(payment.amount) AS total_sales
FROM payment
   JOIN ... rental, inventory, film, film_category, category ...
GROUP BY category.name
```