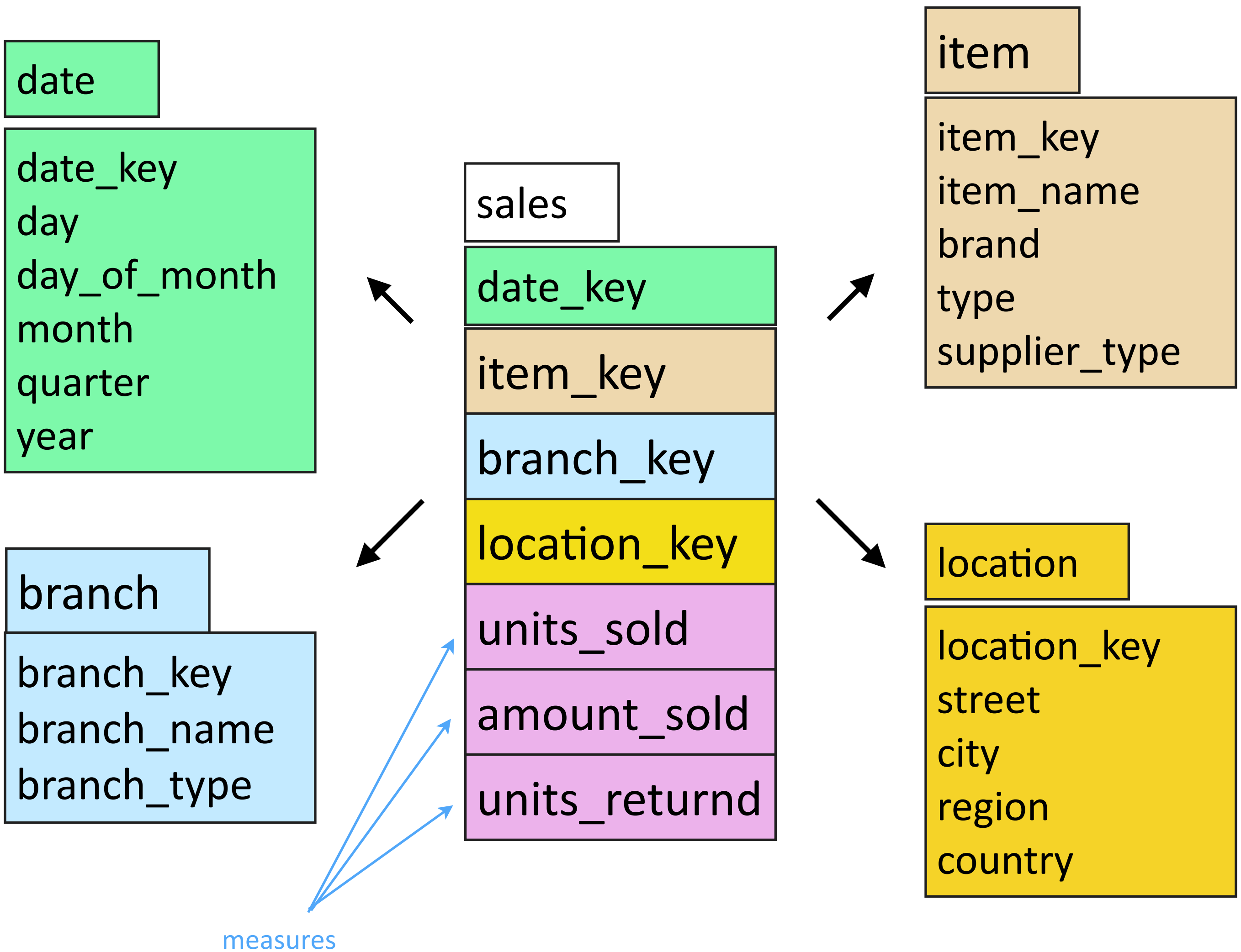
TÉCNICO LISBOA

Data Analysis and Integration

Data warehouse design

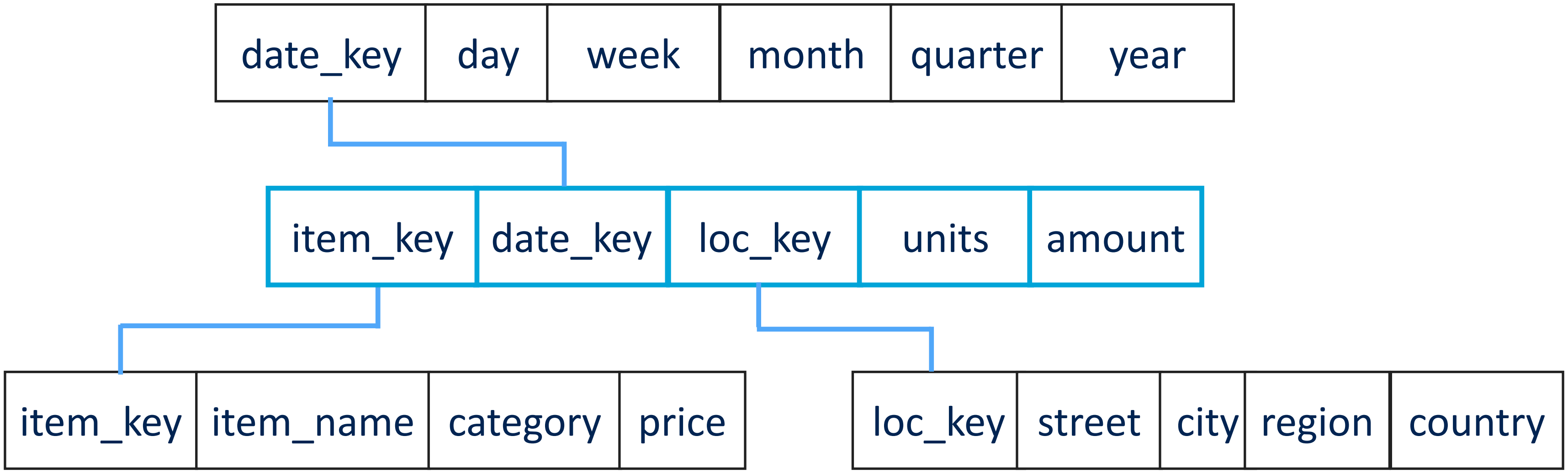
Star Schema

Star Schema



date
date_key
day
day_of_month
month
quarter
year
sales
date_key
item_key
branch_key
location_key
units_sold
amount_sold
units_returnd
item
item_key
item_name
brand
type
supplier_type
branch
branch_key
branch_name
branch_type
location
location_key
street
city
region
country
measures

Star Schema

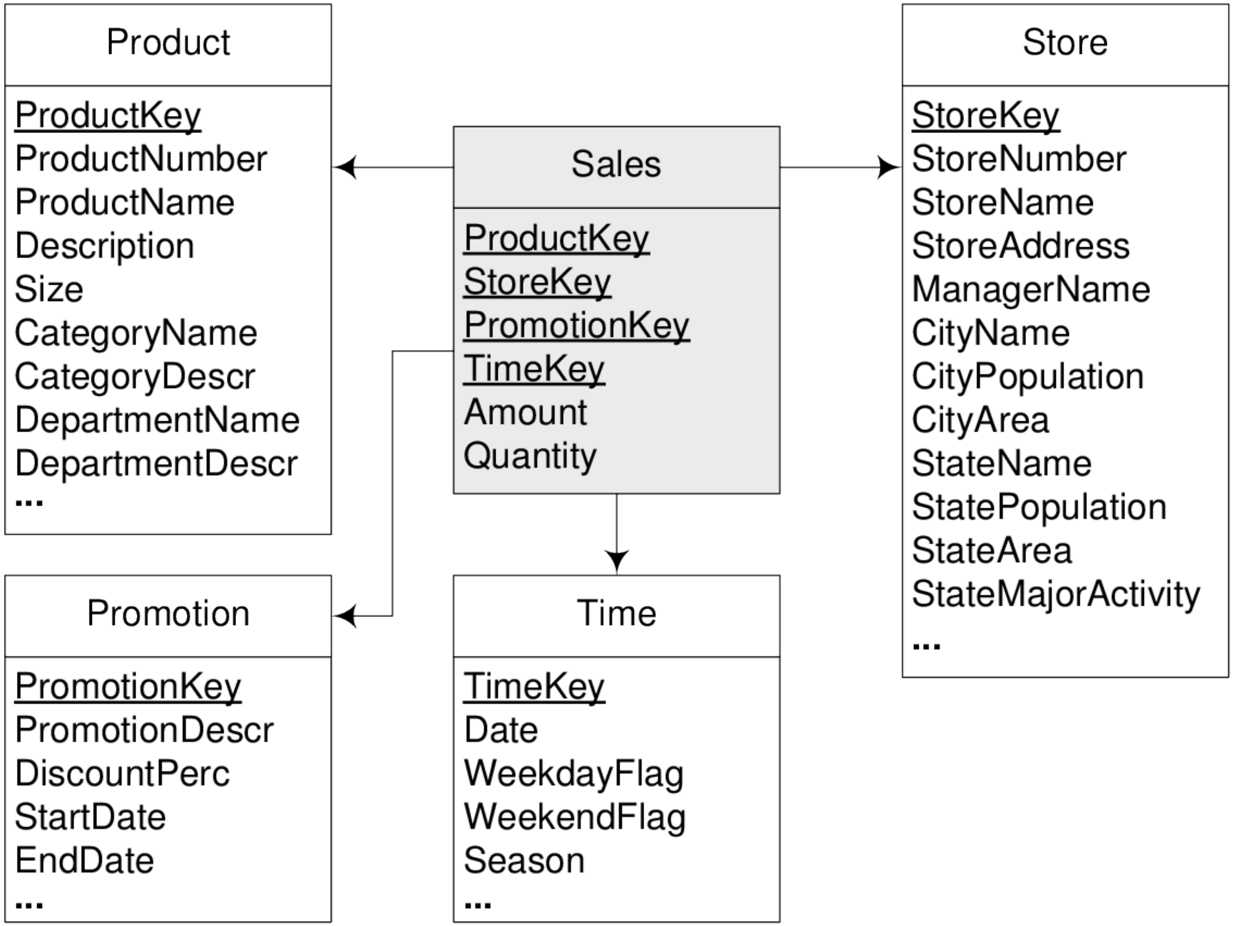


- Fact table is KEY → FACTS
- Fact table size dominates the database size; dimension tables have relatively few rows
- Dimension tables have redundant information; but redundancy (by design) is less important that efficiency
- Update and Delete operations rarely occur

Fact Tables

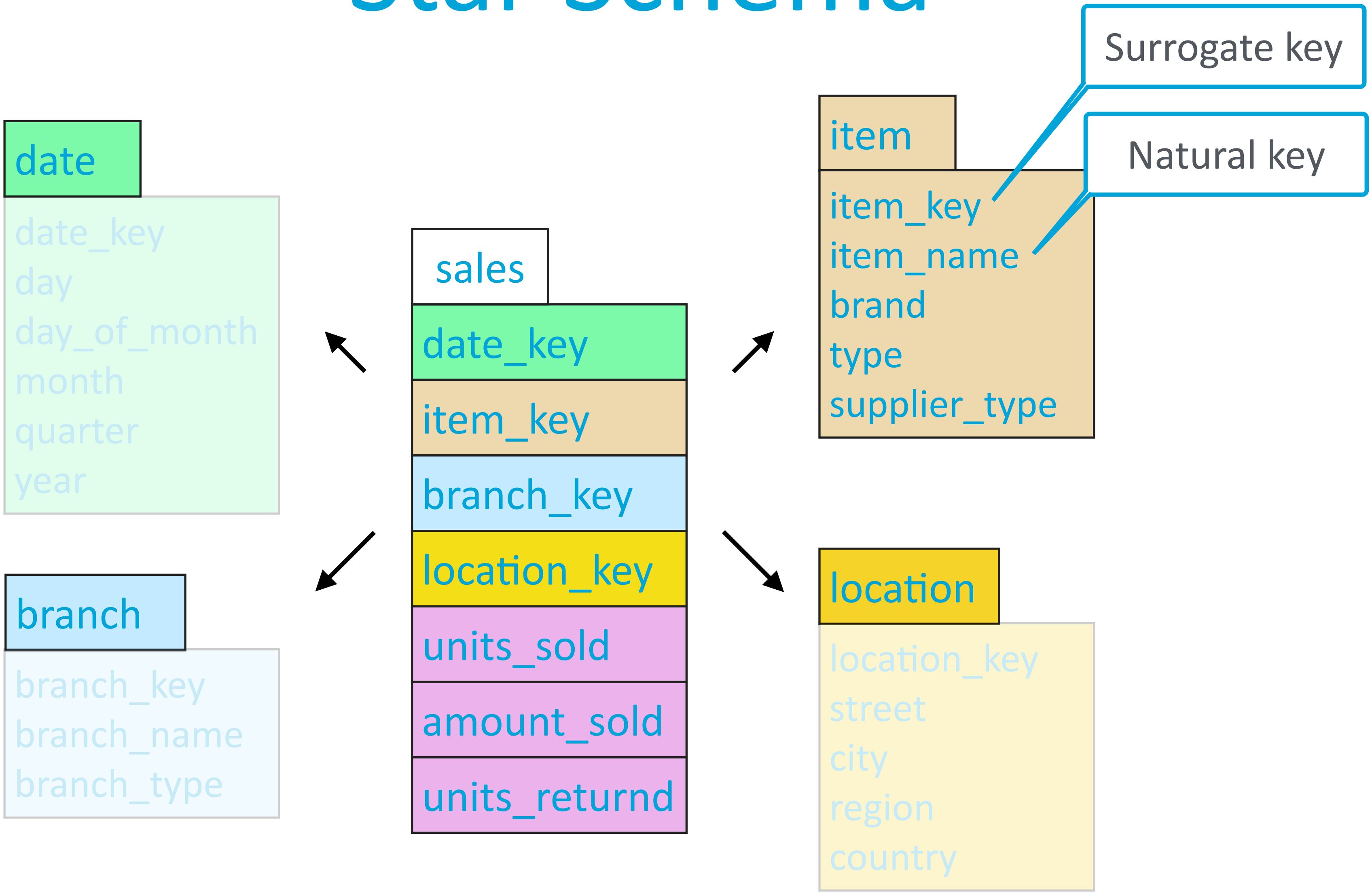
- They stand at the center of a star schema
- Contain numerical measurements (i.e. observations) related to a certain business process that can be aggregated through an aggregation function
  - E.g.: Sales at the Lisbon store on 24-12-2017
- The key of each fact is combination of keys to the distinct dimension tables

Star schema (running example)



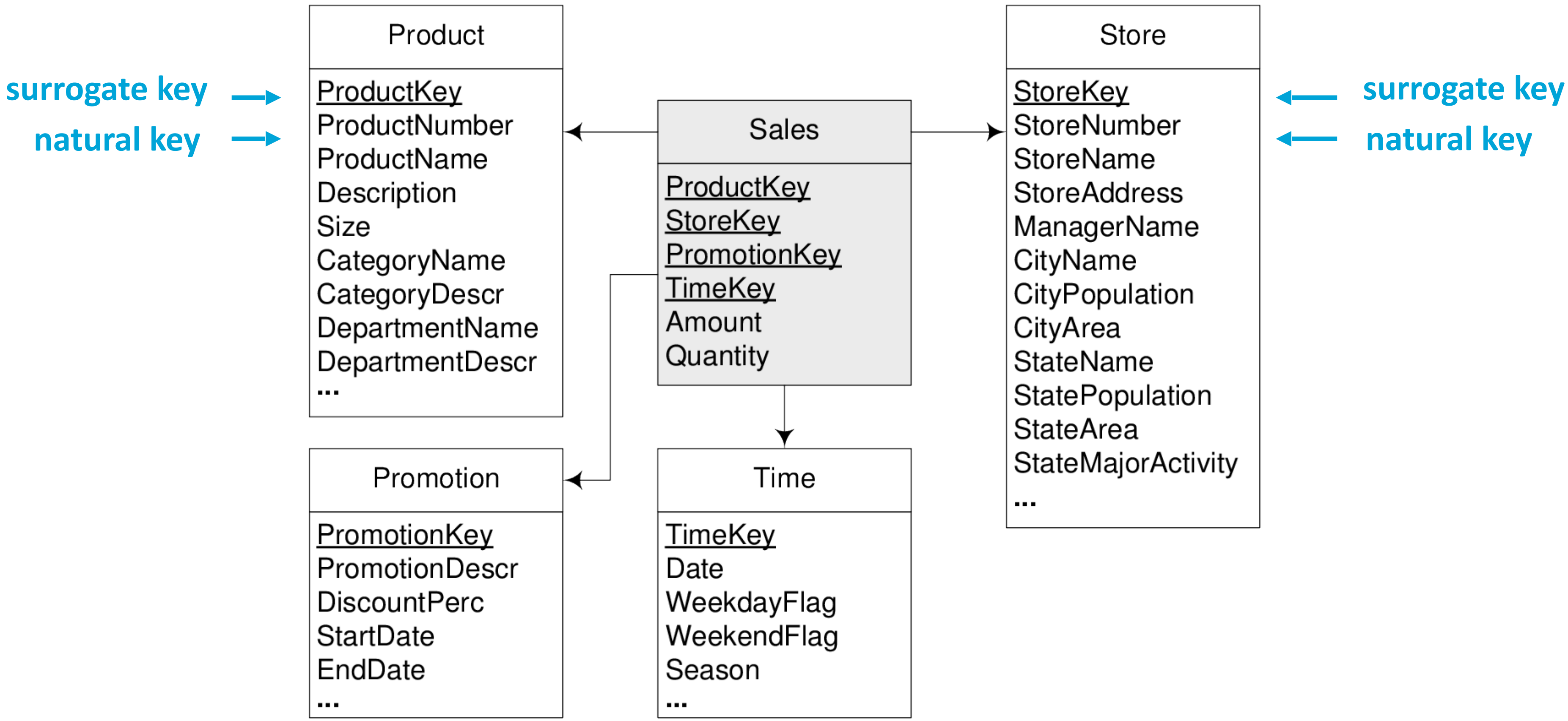
Surrogate Keys

Star Schema


date
date_key
day
day_of_month
month
quarter
year
branch
branch_key
branch_name
branch_type
sales
date_key
item_key
branch_key
location_key
units_sold
amount_sold
units_returnd
item
item_key
item_name
brand
type
supplier_type
Surrogate key
Natural key
location
location_key
street
city
region
country

Surrogate keys

- Each **dimension** has its own key



Surrogate (Technical) Keys

- A data warehouse has its own primary keys
  - these are called **surrogate keys** or **technical keys**
    - **ProductNumber** identifies products in the original database
    - **ProductKey** identifies products in the data warehouse
  - **Surrogate keys** replace the original primary keys (**natural keys**)
    - Provide independence from keys in the original data sources
    - Solve inconsistencies between keys from multiple sources

Independence and efficiency

Keys are represented as **integers** to improve efficiency

avoid less efficient data types, such as strings

Semantic vs. Technical Keys

- A **semantic key** (or **natural key**, or **business key**) is based on the entity's attributes. They allow for a very simple way to understand and compare entities.
- A **technical key** (or **surrogate key**) has values that are often unrelated to the fields of the entity. Technical keys are typically constructed when the entity is inserted in the DB and are immutable.

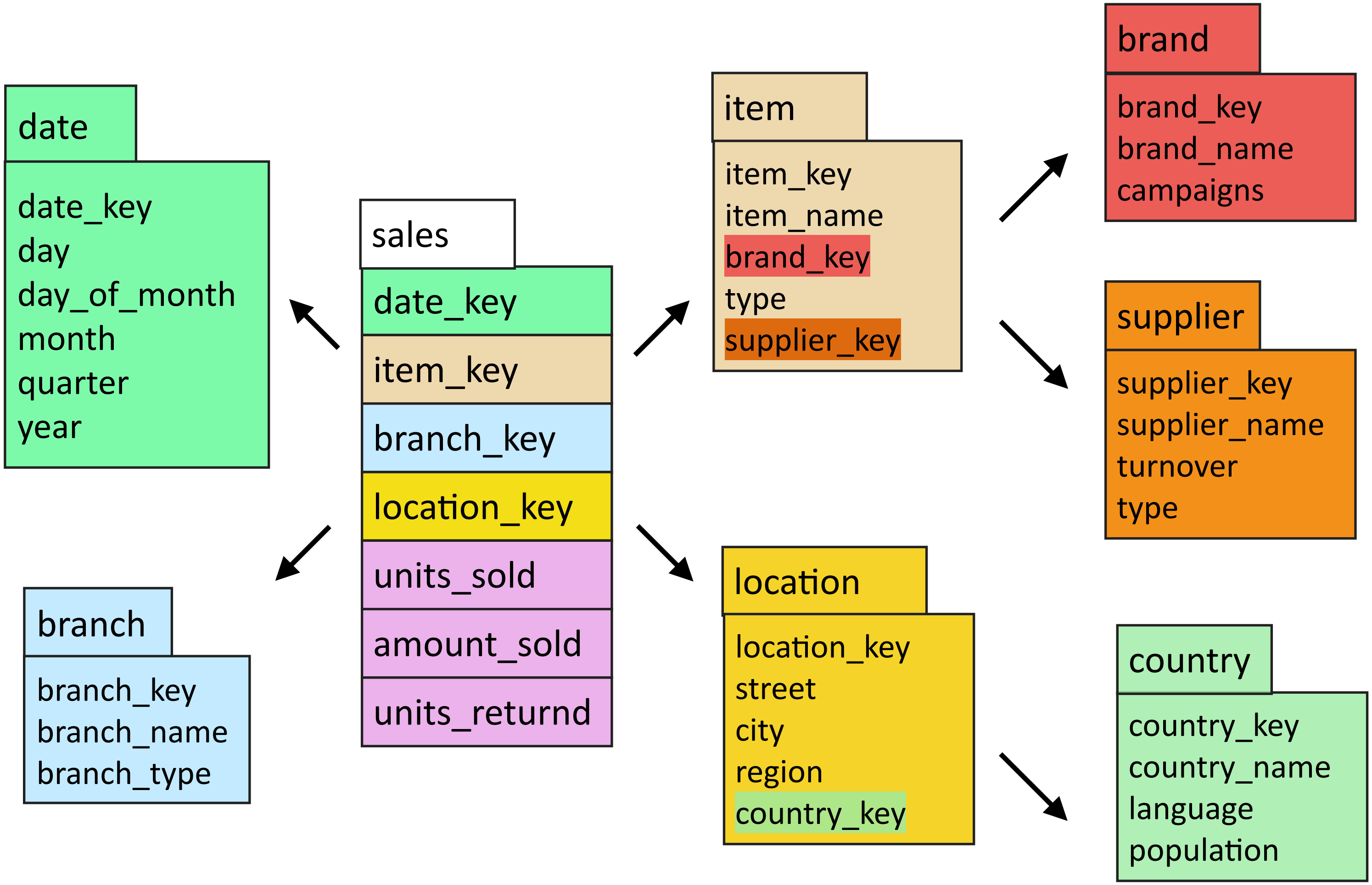
Simplifies combining (joining) tables

Avoids cascading (millions) of updates when natural keys change

Multidimensional Schema Patterns

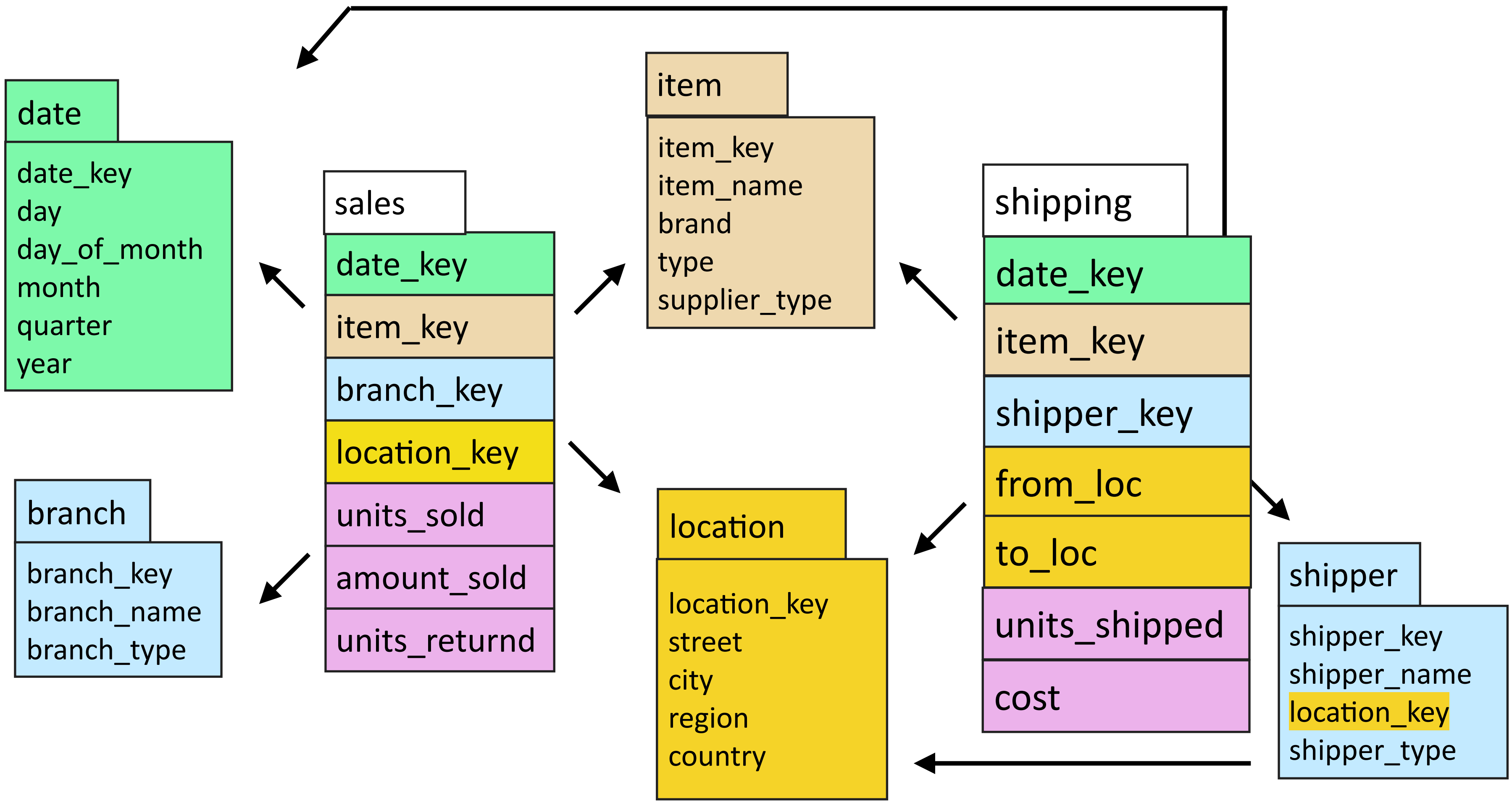
Snowflakes and Constellations

Snowflake Schema



Dimension tables are **connected** to a set of **smaller** (normalised) dimension tables

Fact Constellation Schema



Multiple fact tables **sharing** the same dimension tables

Conceptual Modelling of Data Warehouses

- Star Schema: A schema with a fact table in the middle connected to a set of dimension tables
- Snowflake Schema: A star schema where some dimensional hierarchy is normalised into a set of smaller dimensions tables (thus forming a shape similar to a snowflake)
- Fact Constellation: Multiple fact tables sharing the same dimension tables (viewed as a collection of starts and therefore, called a fact constellation or galaxy schema)

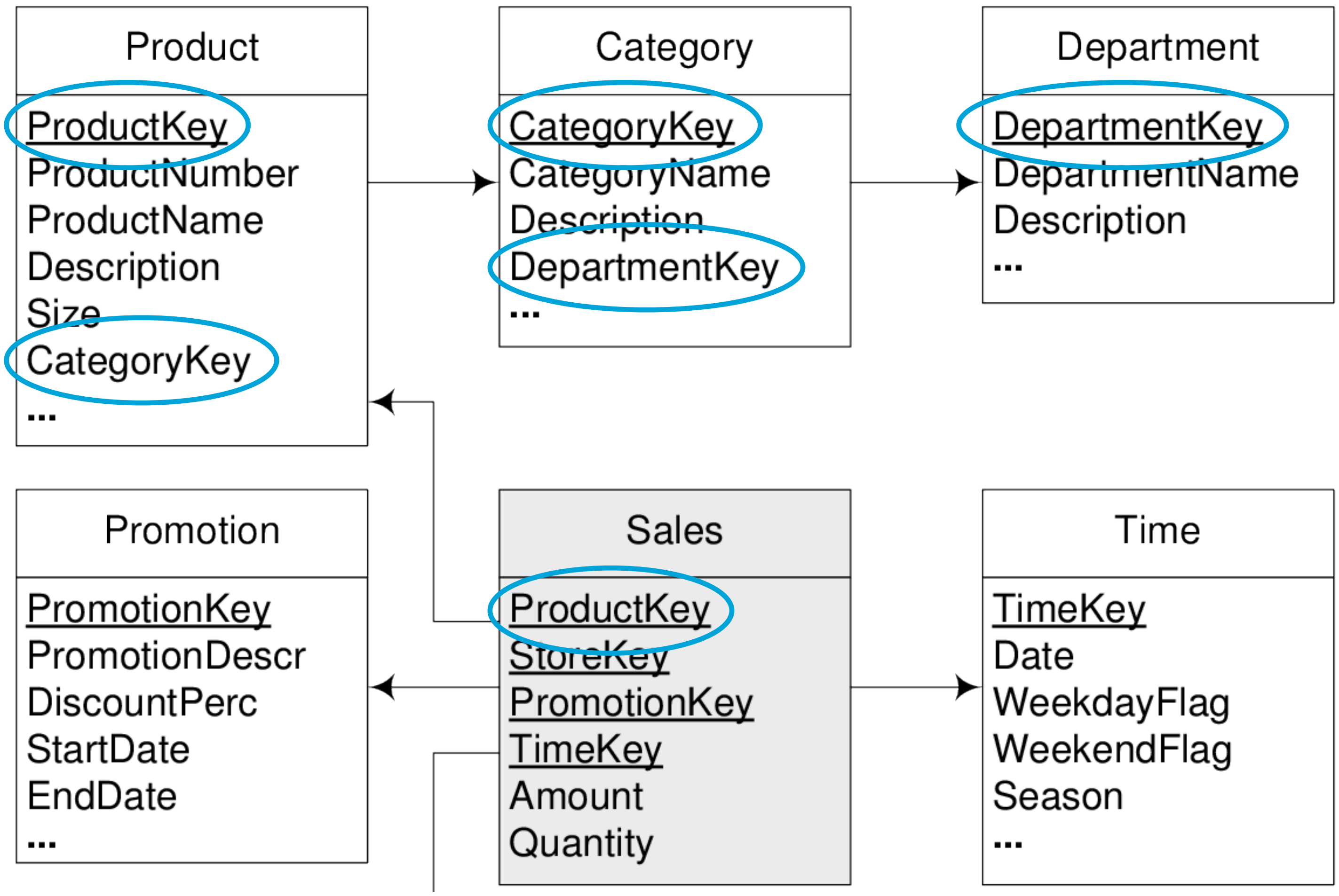
Snowflake

Snowflake schema

- When dimensions that store redundant data (are normalized)
  - **CategoryName**, **CategoryDescr** are the same fo multiple products
    - replace by **CategoryKey** and move them to another table
  - **DepartmentName**, **DepartmentDescr** are the same for multiple categories
    - replace by **DepartmentKey** and move them to another table

Product
ProductKey
ProductNumber
ProductName
Description
Size
CategoryName
CategoryDescr
DepartmentName
DepartmentDescr
...

Snowflake schema

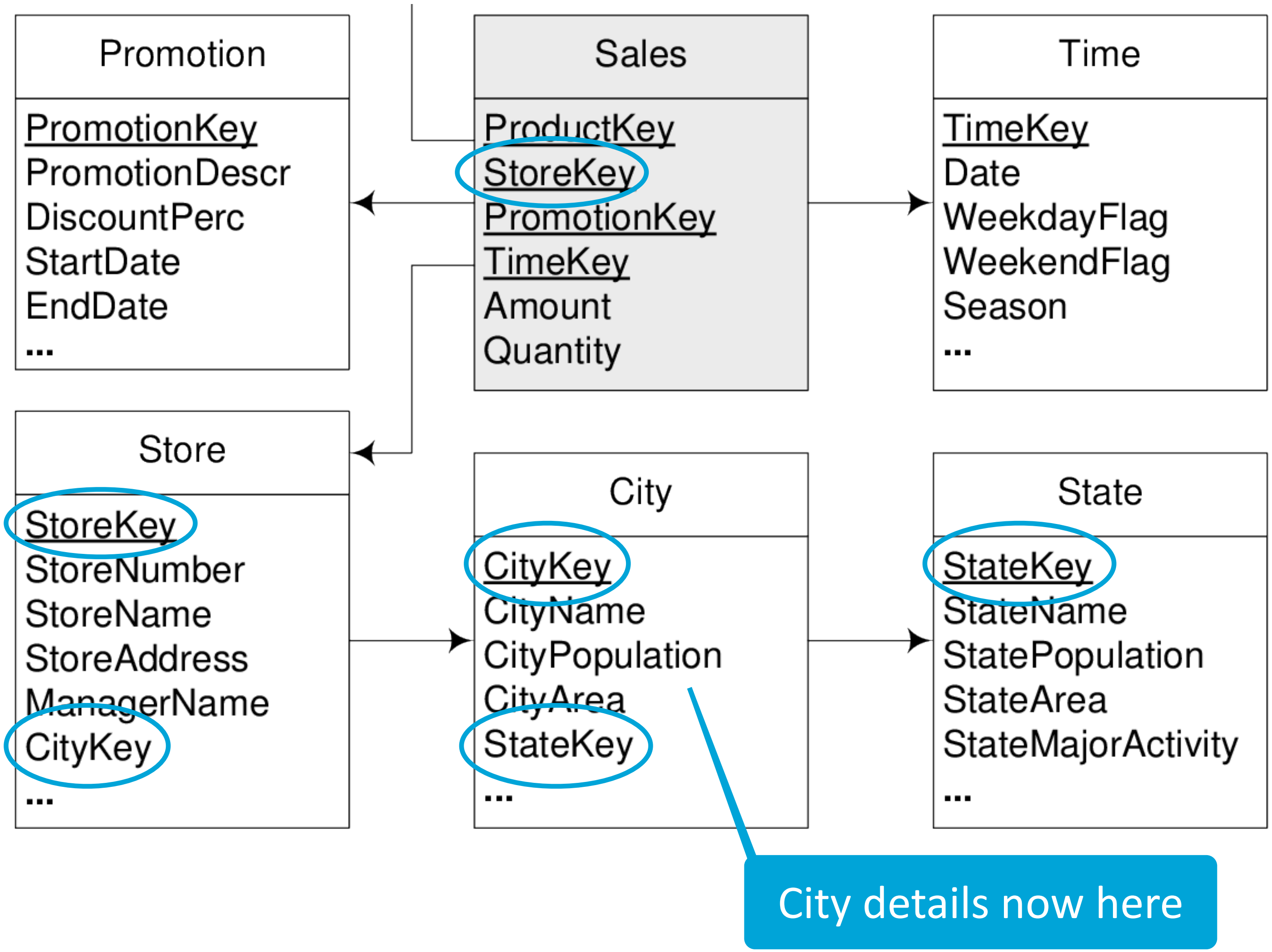


Snowflake schema

- Another example
  - **CityName**, **CityPopulation**, **CityArea** are the same for multiple stores
    - replace by **CityKey** and move these details to another table
  - **StateName**, **StatePopulation**, **StateArea**, etc. are the same for multiple products
    - replace by **StateKey** and move them to another table

Store
<u>StoreKey</u> StoreNumber StoreName StoreAddress ManagerName CityName CityPopulation CityArea StateName StatePopulation StateArea StateMajorActivity ...

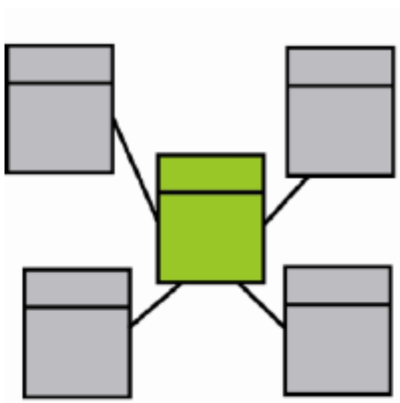
Snowflake schema



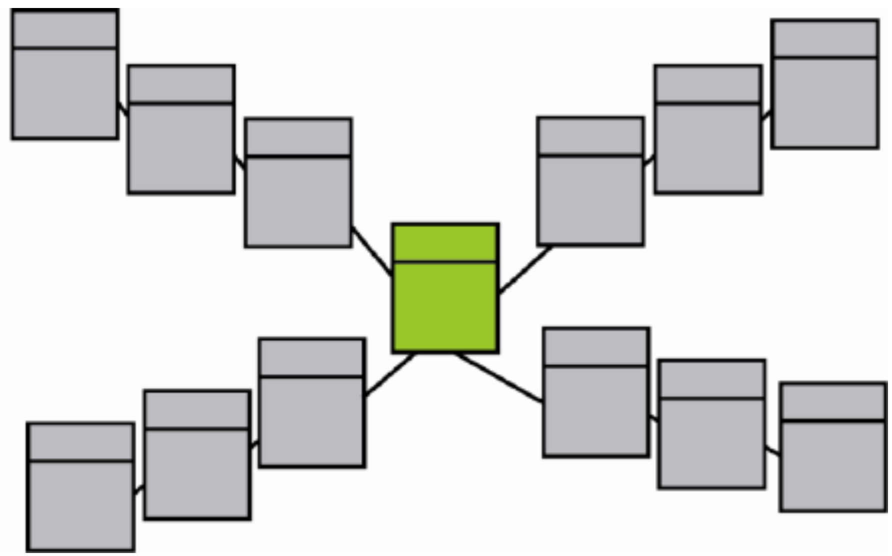
Snowflake schema

Snowflake schema

- Star Schema



- Snowflake Schema



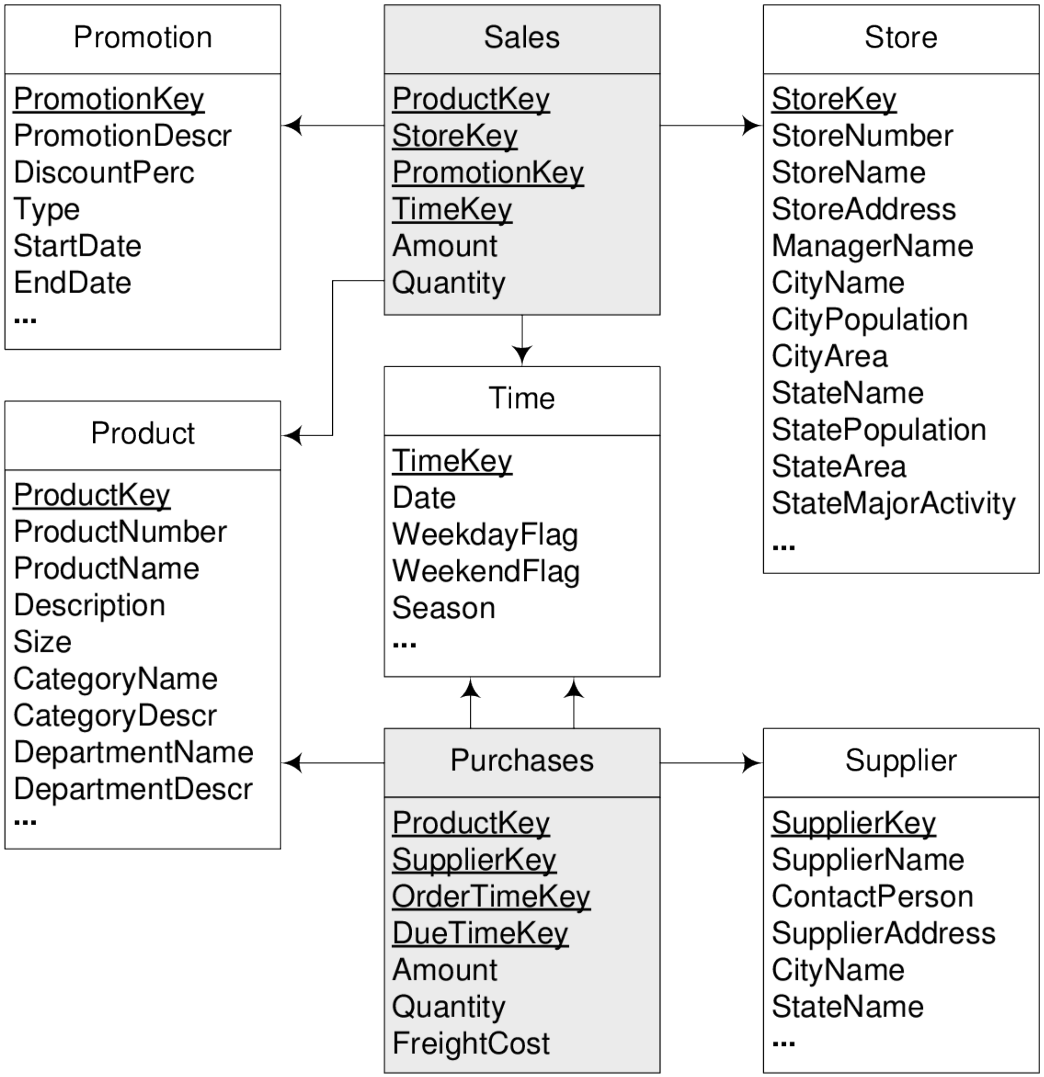
All dimensions are normalised

Constellation

Constellation schema

- Multiple fact tables
  - e.g. sales facts, purchase facts
- Some dimension tables may be shared
  - e.g. product, time

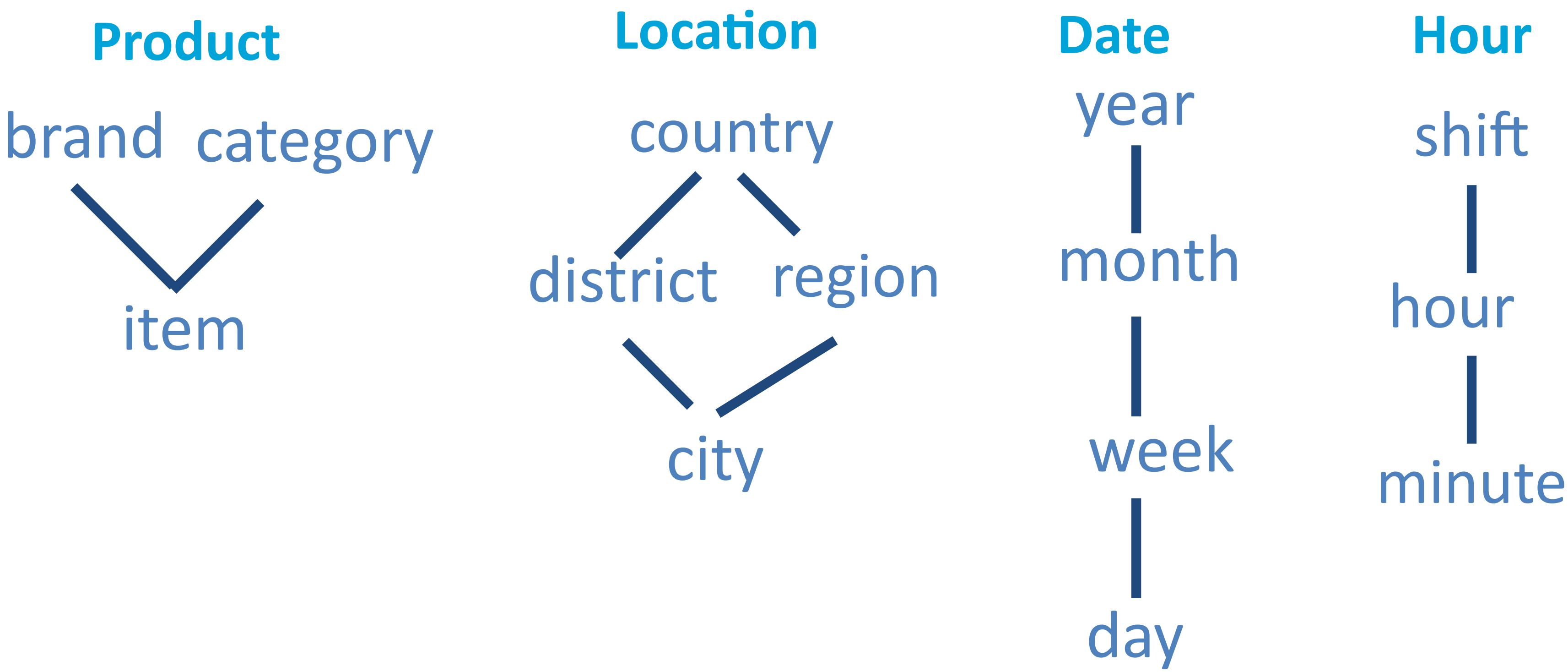
Constellation schema



Hierarchies

Hierarchies

For each dimension, the set of values can be organised into an hierarchy

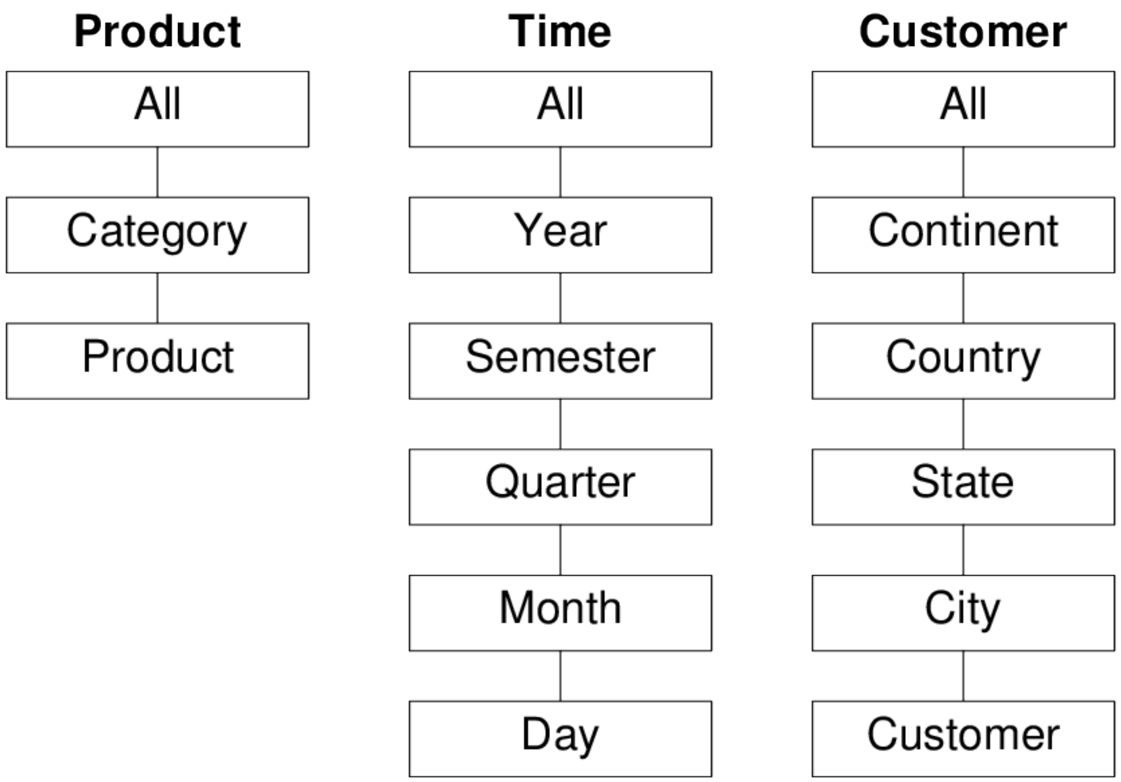


Hierarchies

- Dimension hierarchies are essential to enable analysis at different levels of detail
  - define a hierarchical structure of levels relating lower-level members to higher-level ones
- In real-world applications, users must deal with complex hierarchies of various kinds
  - however, current DW and OLAP systems support only a limited set of hierarchies

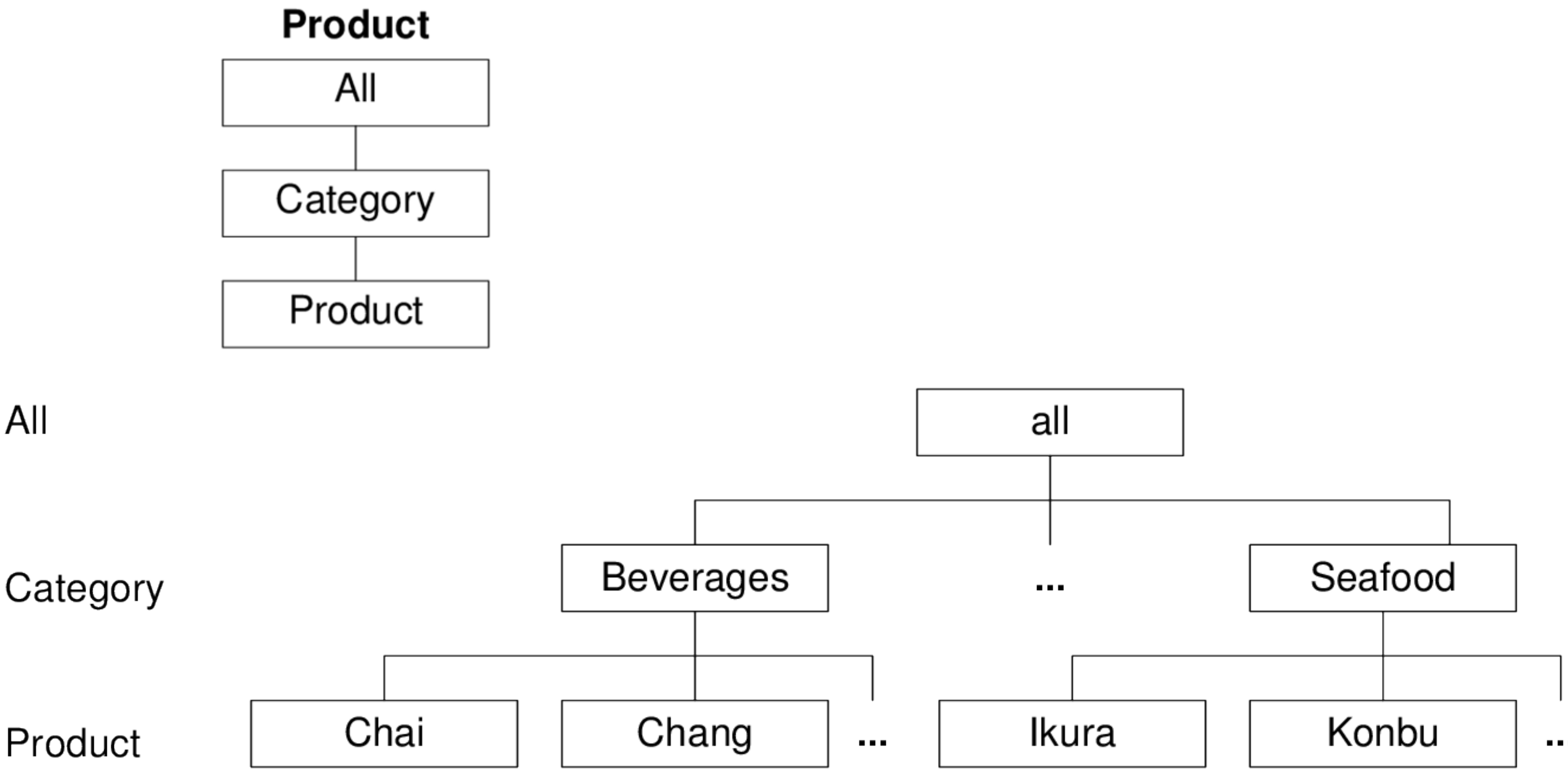
Hierarchies

- Product, time and customer dimensions



Hierarchy members

- Example of a hierarchy and an instance with its members

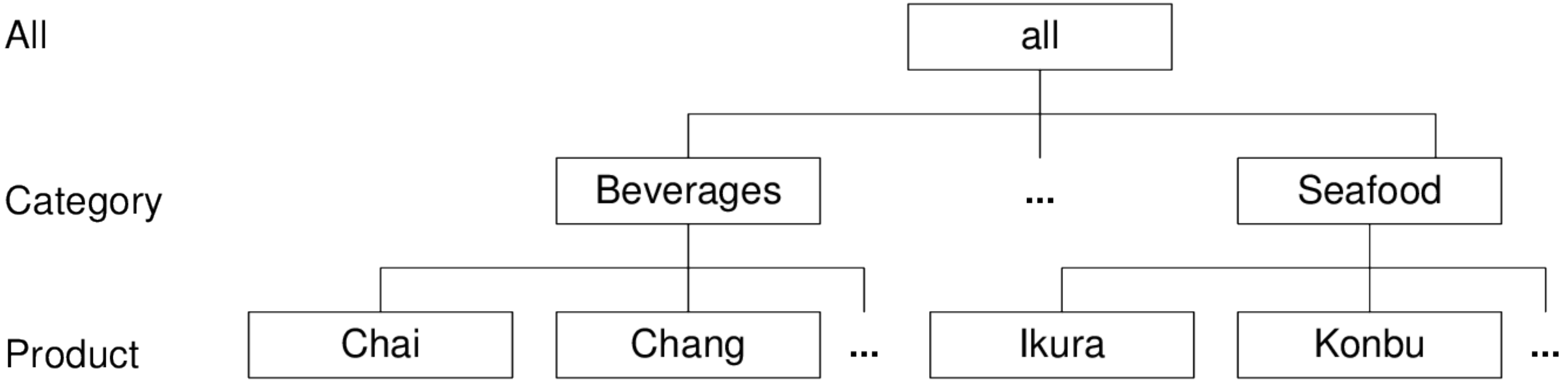


Types of hierarchy

- Balanced hierarchy
- Unbalanced hierarchy
- Recursive hierarchy
- Generalized hierarchy
- Ragged hierarchy
- Alternative hierarchy
- Non-strict hierarchy

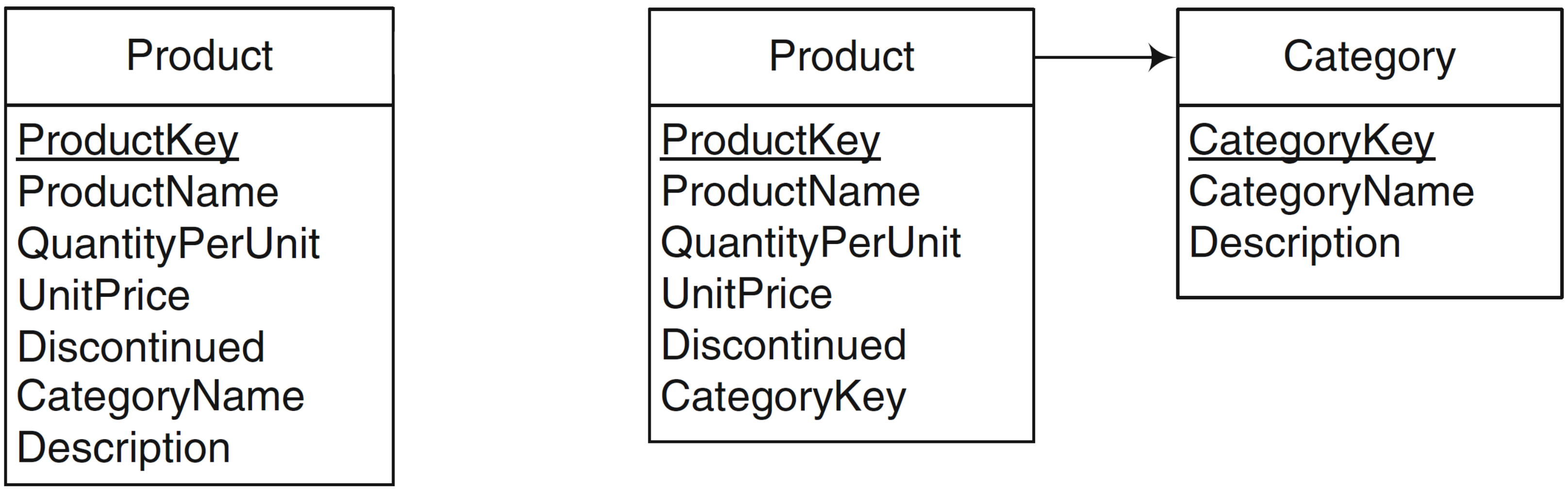
Types of hierarchy

- <u>**Balanced**</u> hierarchy
  - all levels are mandatory
  - all branches have the same length
  - a child member belongs to only one parent



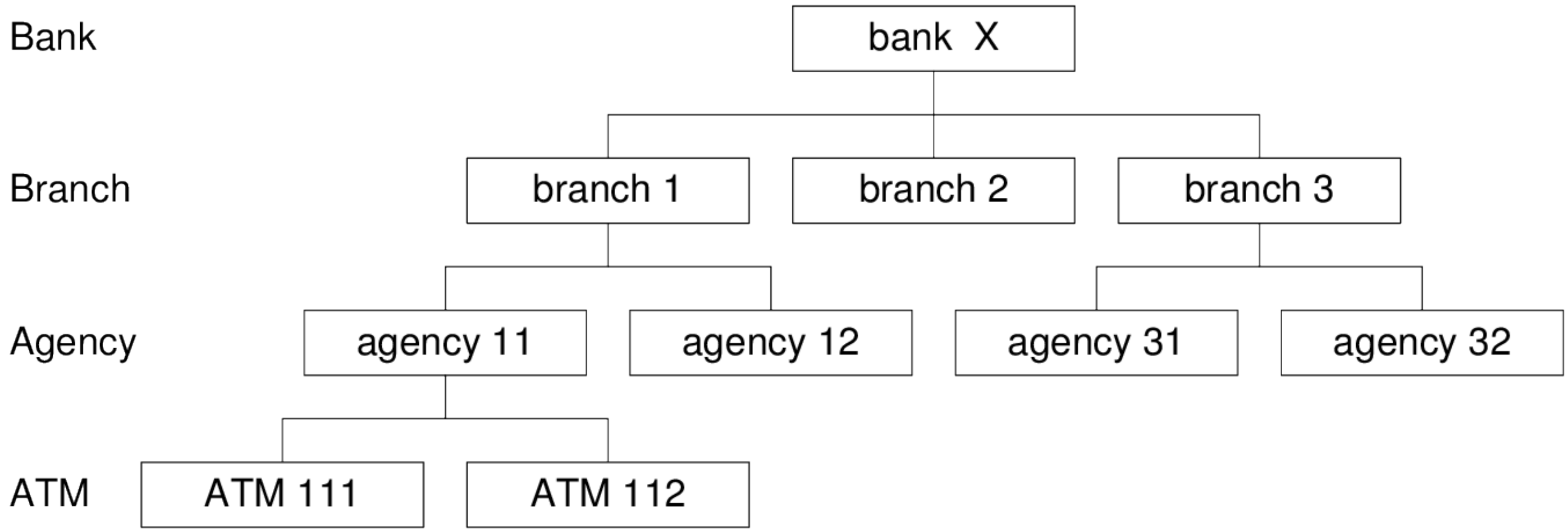
Types of hierarchy

- Encoding a balanced hierarchy
  - flat table (as in star schema) or snowflake structure



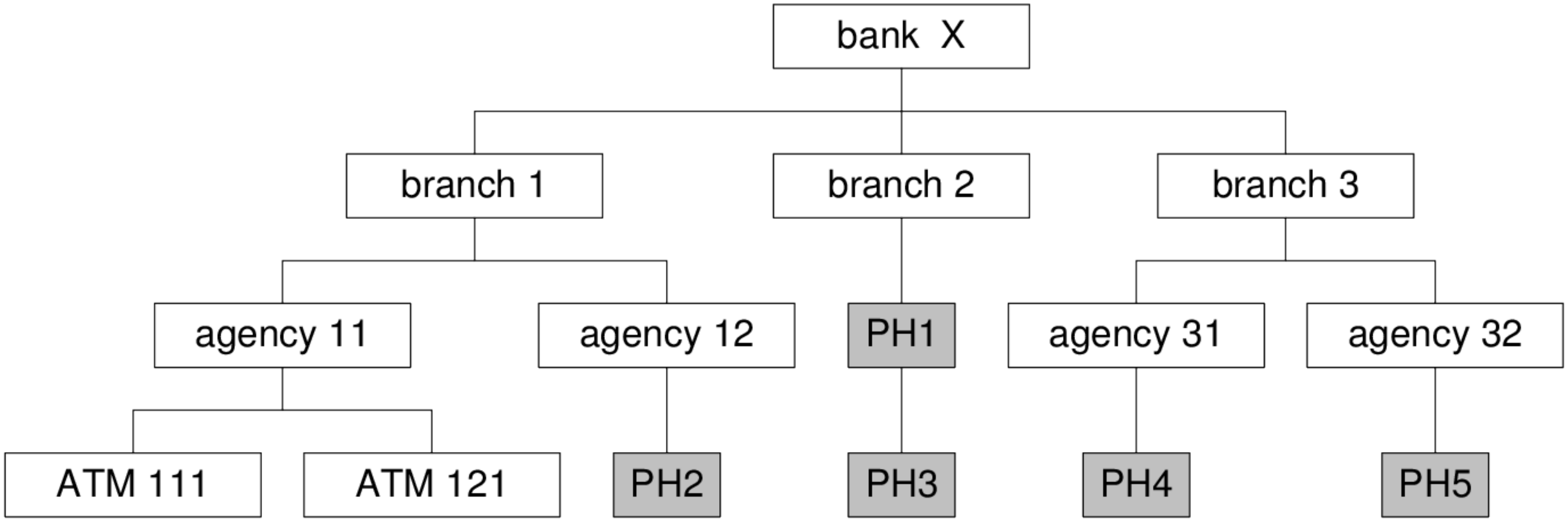
Types of hierarchy

- **<u>Unbalanced</u>** hierarchy
  - some levels are optional
  - branches may have different lengths
  - a child member belongs to only one parent



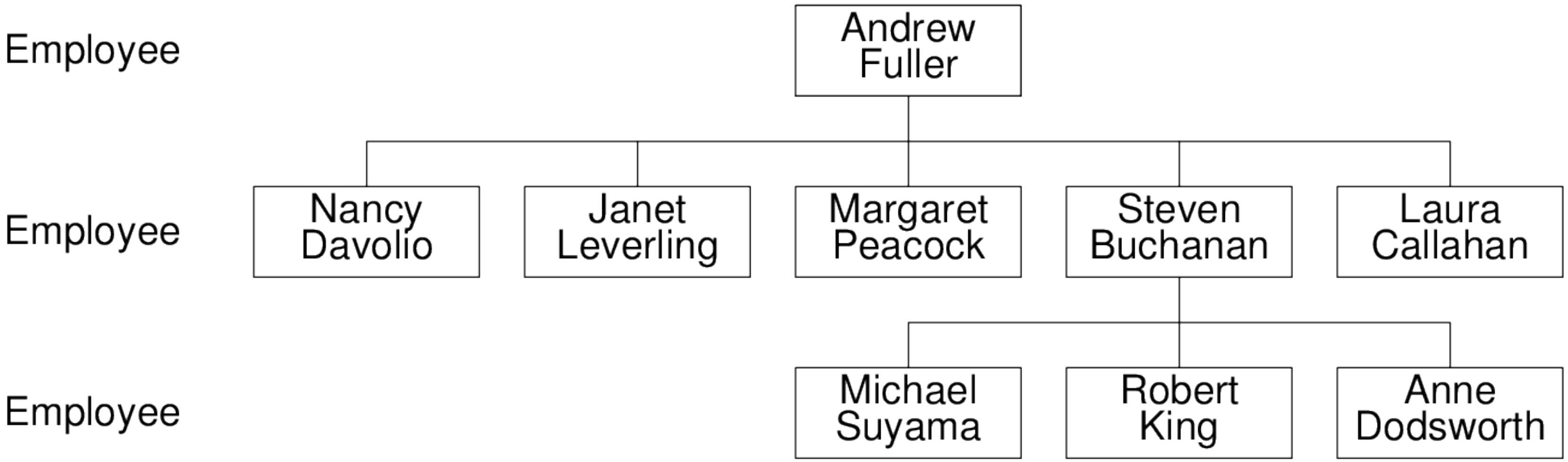
Types of hierarchy

- Encoding an unbalanced hierarchy
  - transform into balanced hierarchy by using placeholders
  - then use flat table or snowflake structure



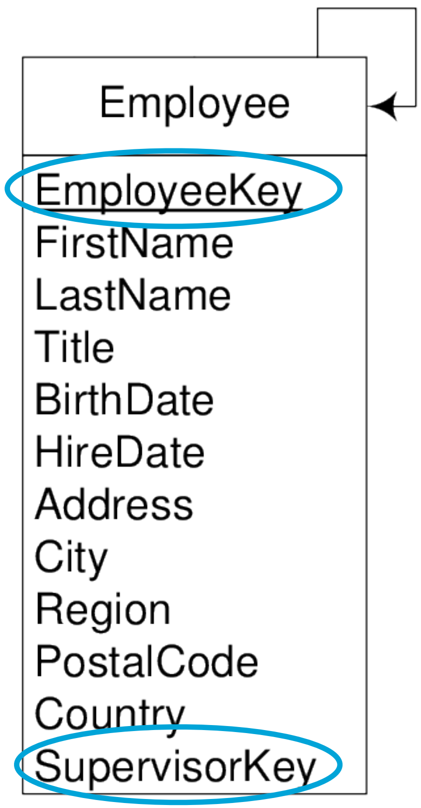
Types of hierarchy

- **Recursive** hierarchy
  - all levels are of the same type
  - can easily become an unbalanced hierarchy
  - requires recursive queries to traverse the hierarchy



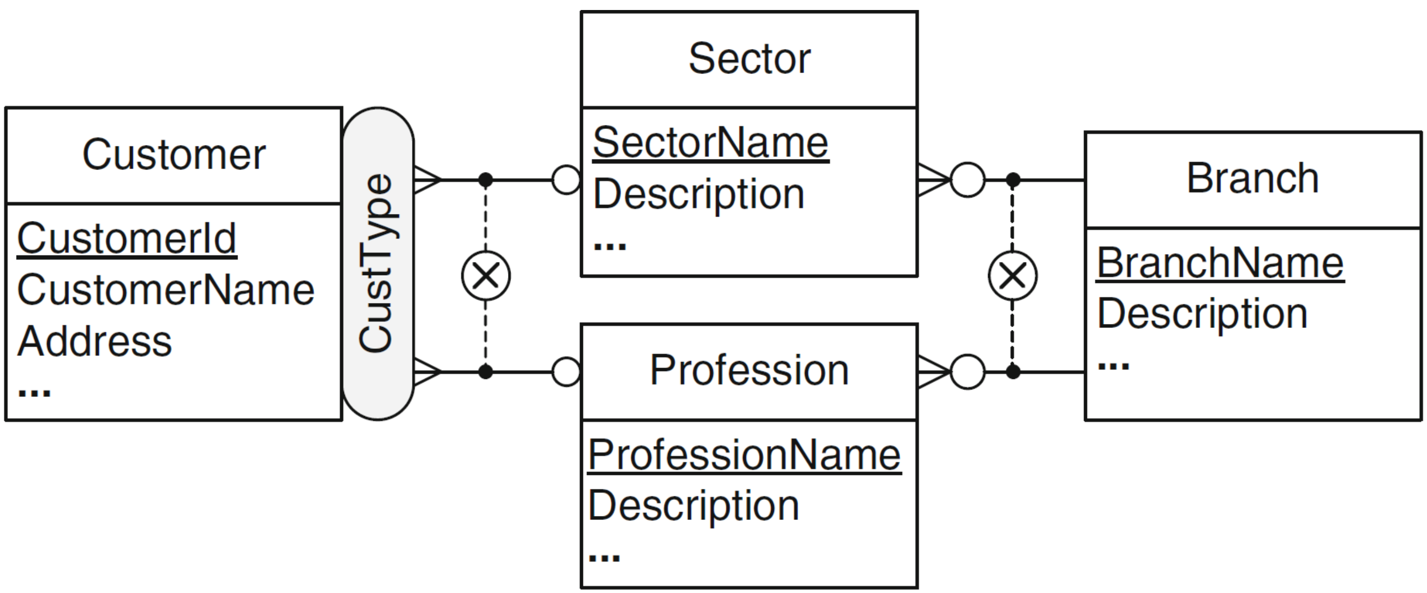
Types of hierarchy

- Encoding a recursive hierarchy
  - The very common scenario



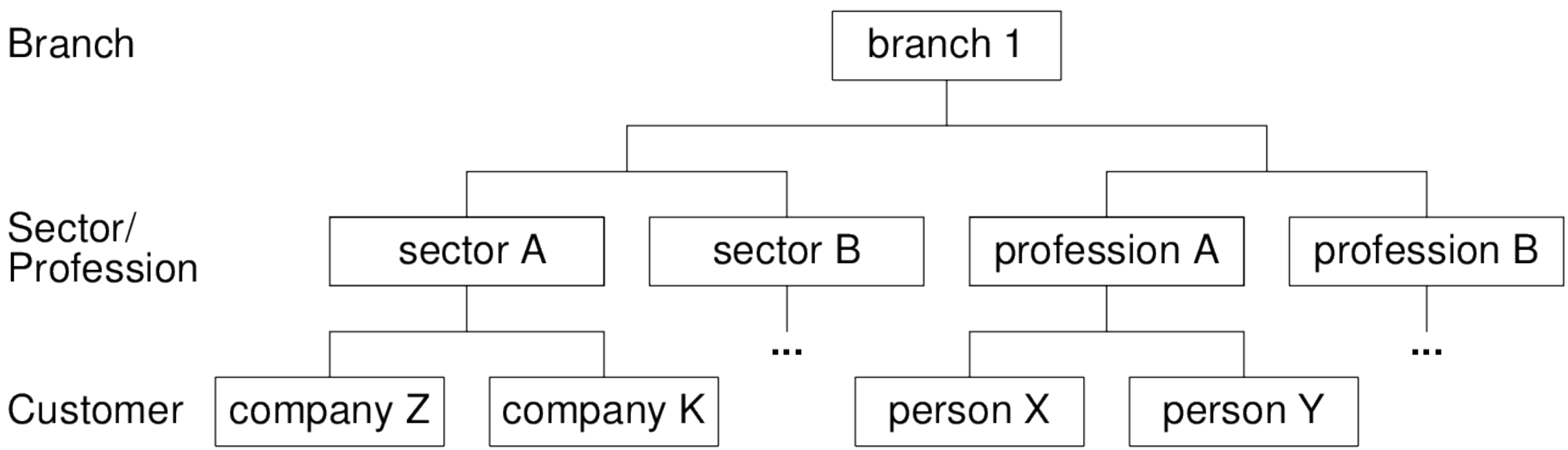
Types of hierarchy

- **Generalized** hierarchy
  - one same level may have distinct types of elements
    - e.g. customers of a bank may be companies (with an industry **sector**) or individual persons (with a **profession**)



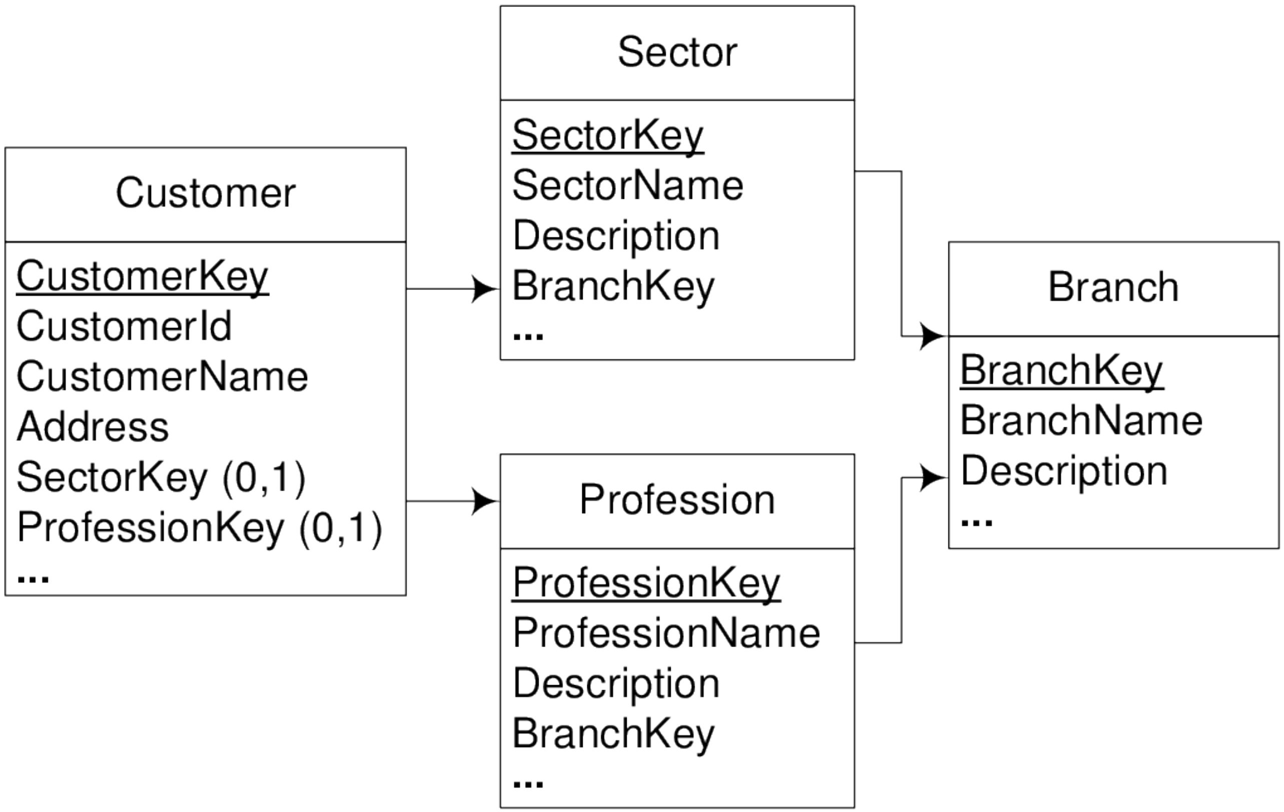
Types of hierarchy

- **<u>Generalized</u>** hierarchy
  - the same level may have different types
    - e.g. customers of a bank may be companies (with an industry **sector**) or individual persons (with a **profession**)



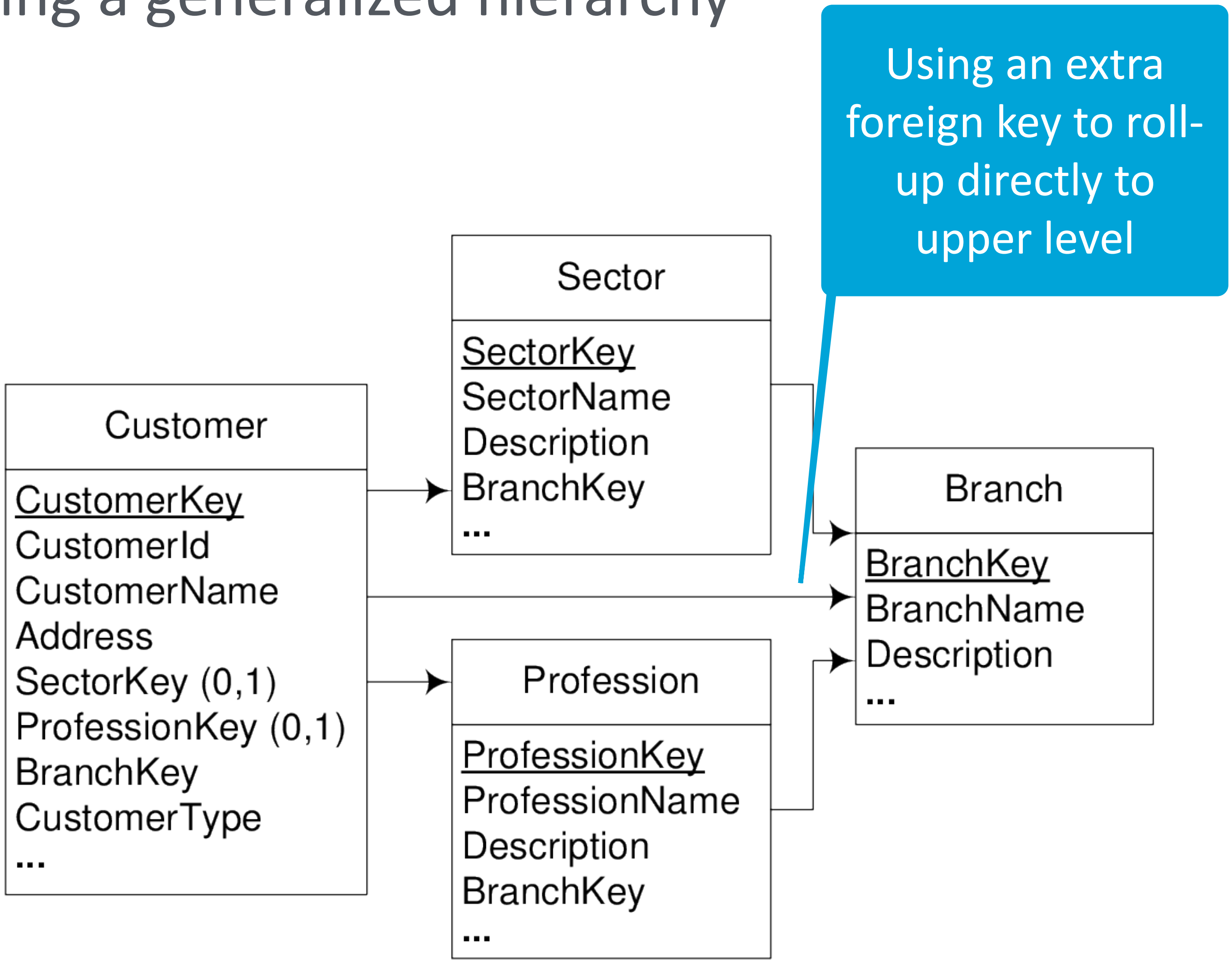
Types of hierarchy

- Encoding a generalized hierarchy
  - Flat table with NULLs or snowflake structure (preferred)
    - different aggregation paths for different types of customer



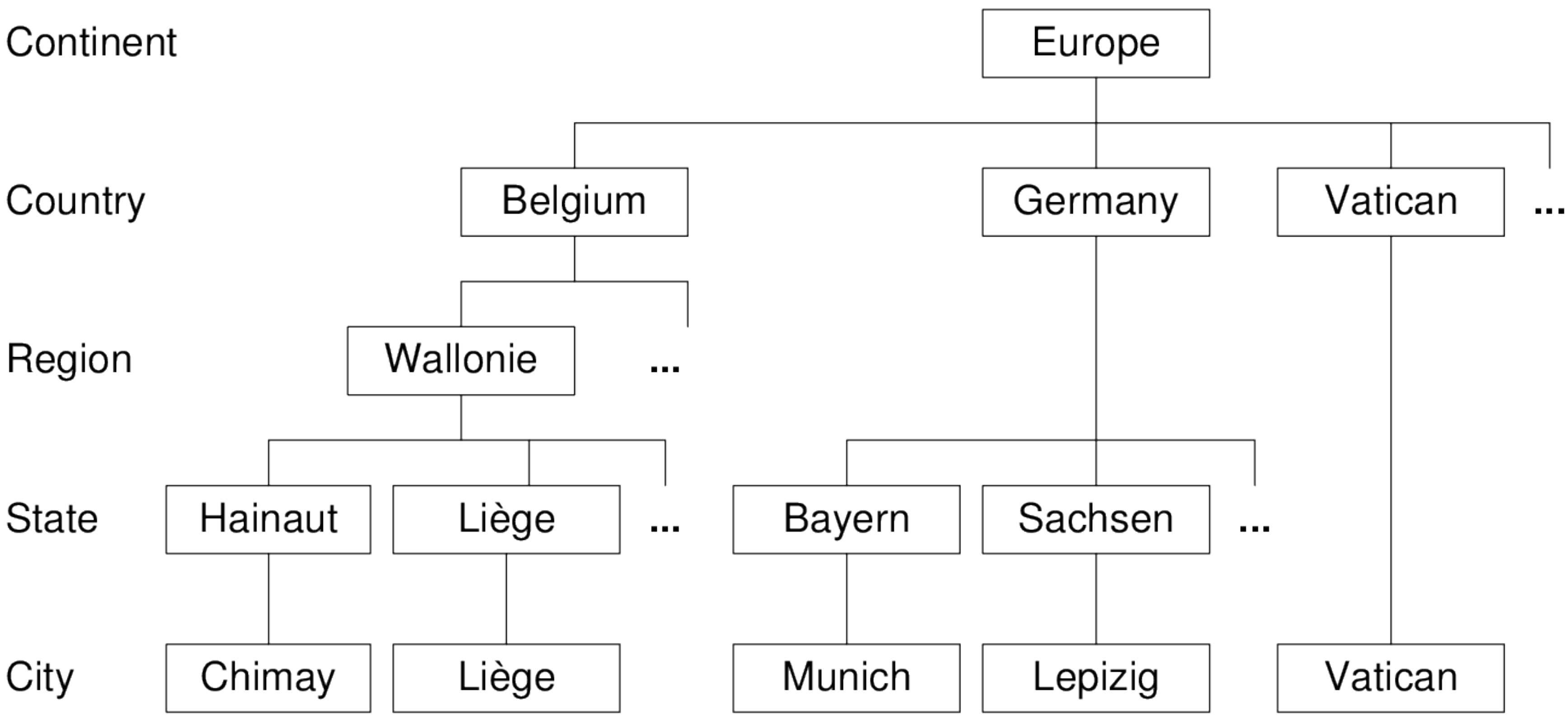
Types of hierarchy

- Encoding a generalized hierarchy



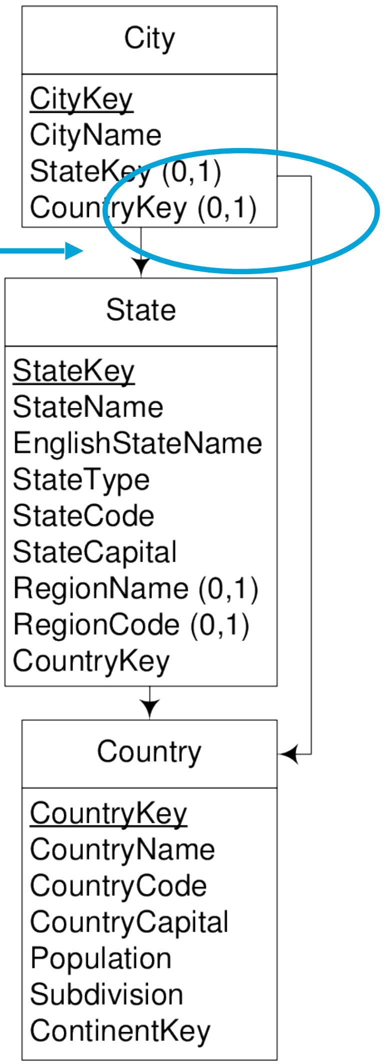
Types of hierarchy

- **Ragged** hierarchy
  – one or more levels can be skipped



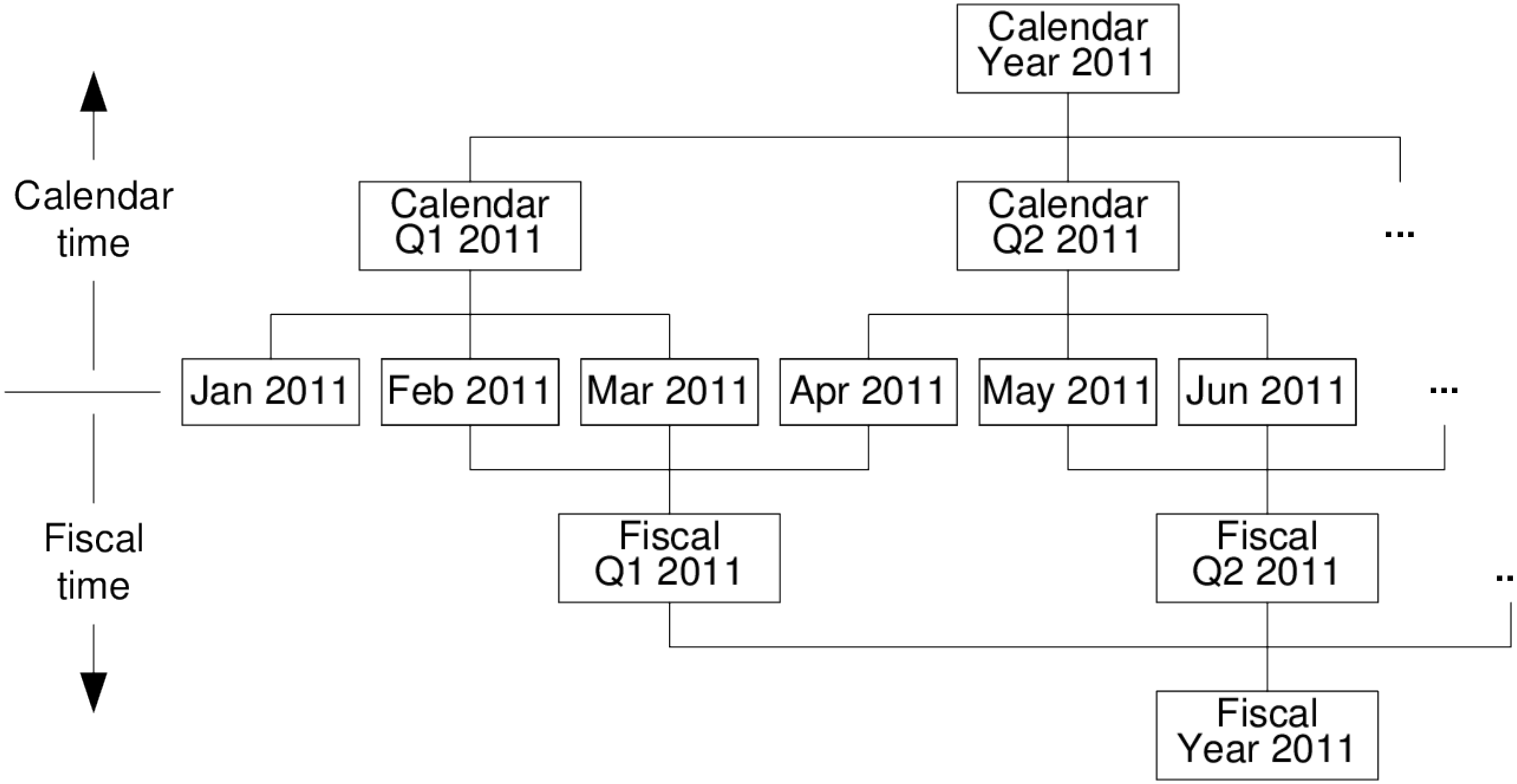
Types of hierarchy

- Encoding a ragged hierarchy
  - Several implementations
    - add extra foreign keys to skip levels, or
    - use placeholders



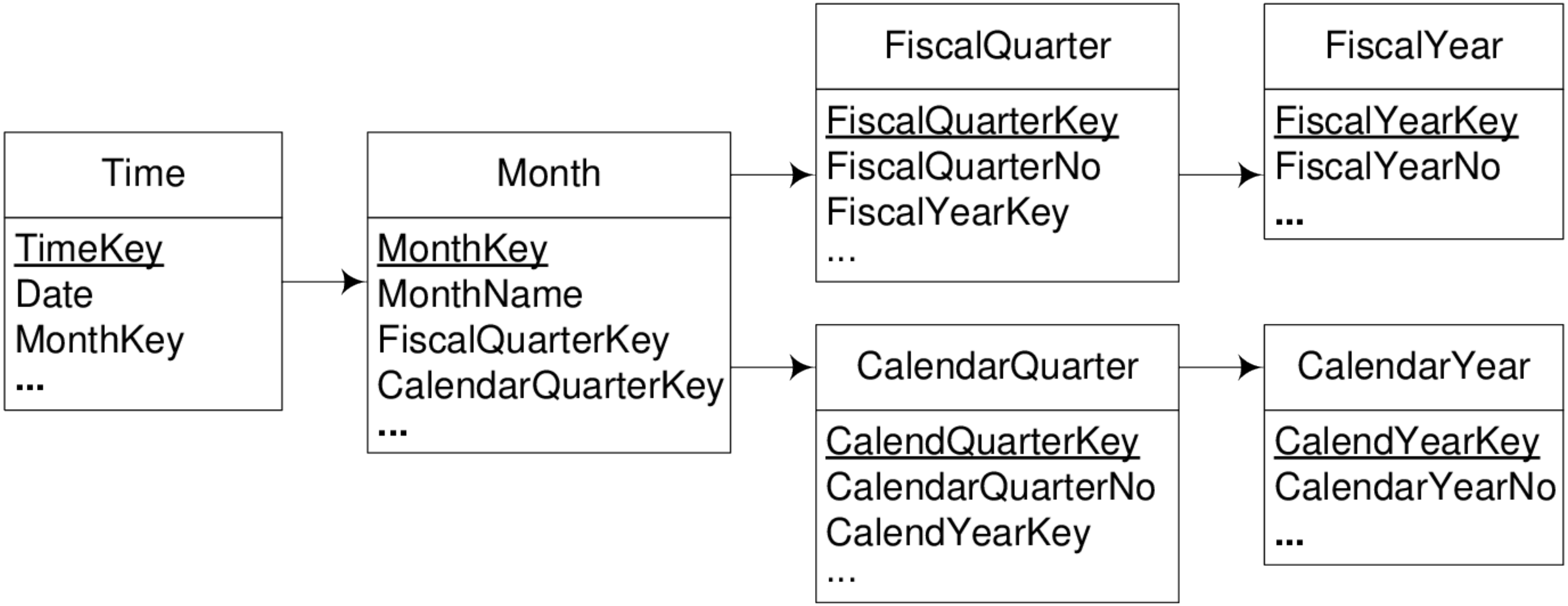
Types of hierarchy

- **Alternative** hierarchy
  - the same level has alternative aggregation paths



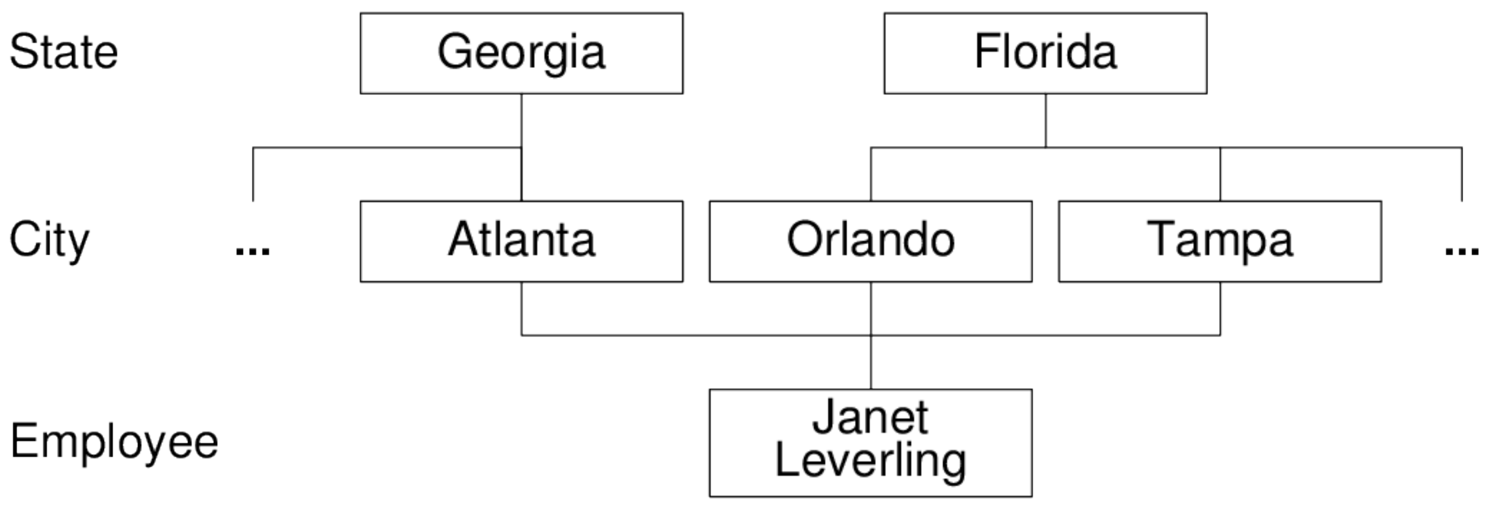
Types of hierarchy

- Encoding an alternative hierarchy
  - use snowflake structure



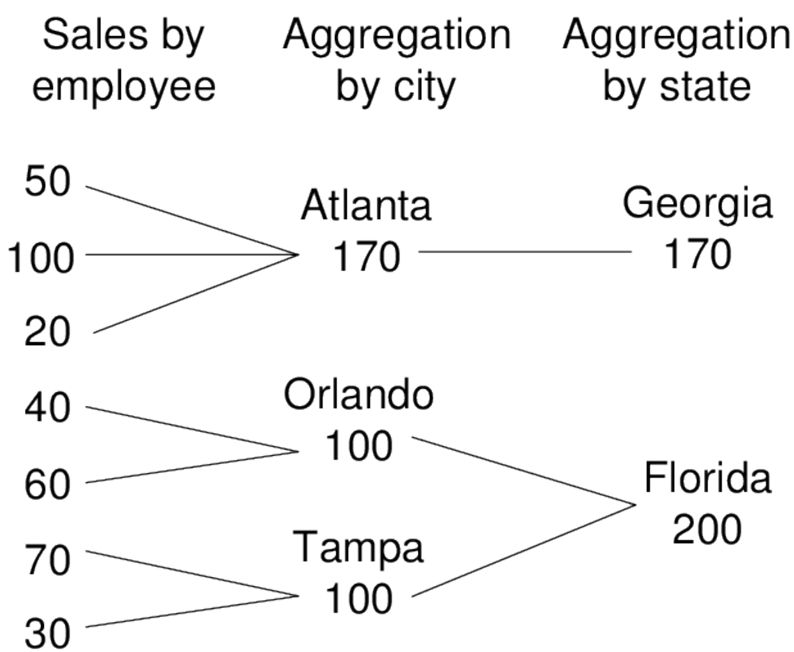
Types of hierarchy

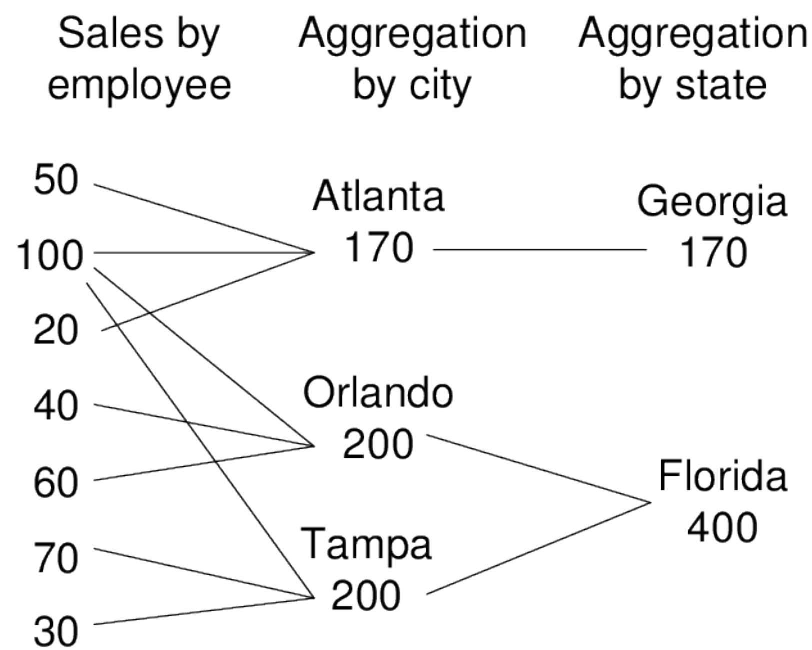
- **Non-strict** hierarchy
  - When member may have several parents
    - e.g. an employee that works in multiple cities
    - e.g. a week that belongs to two months



Types of hierarchy

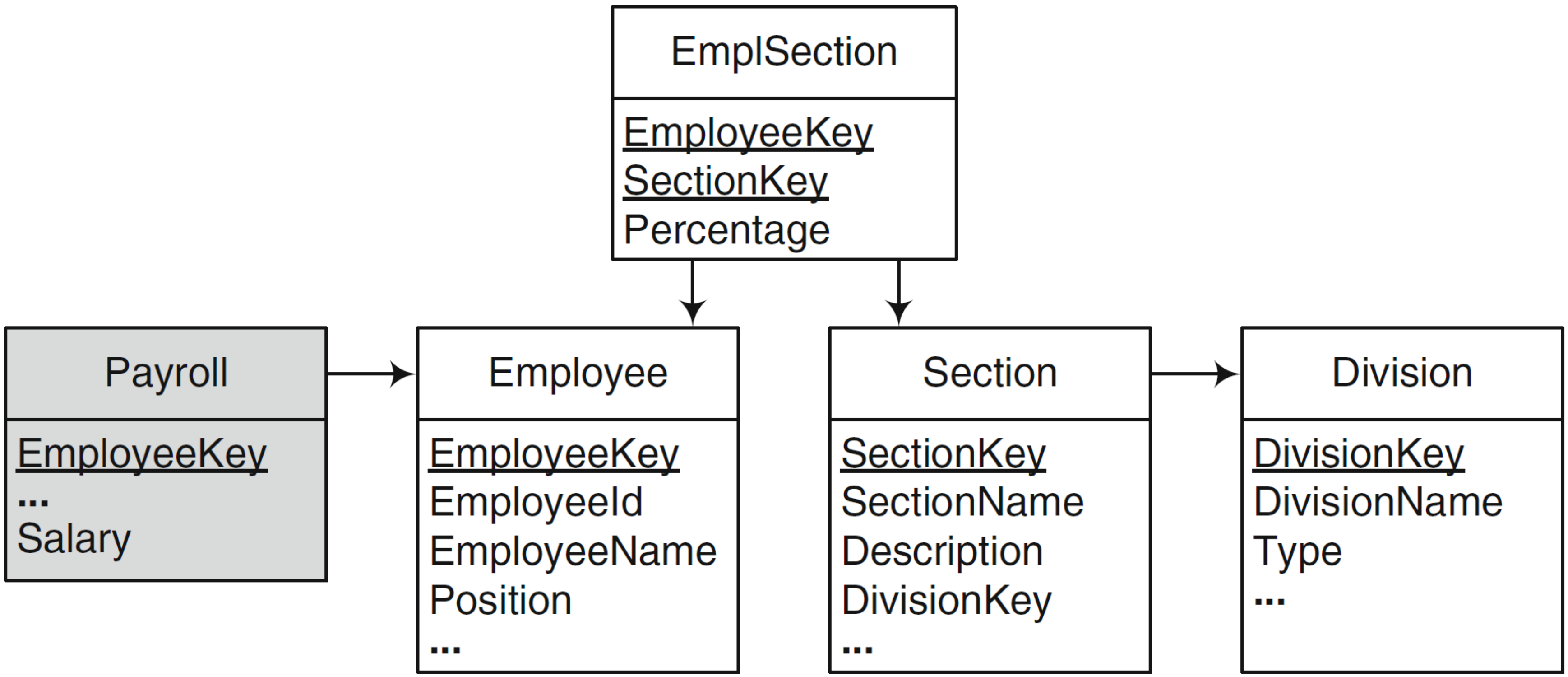
- Encoding a non-strict hierarchy
  - Queries must be created with care to avoid **double counting** on roll-up





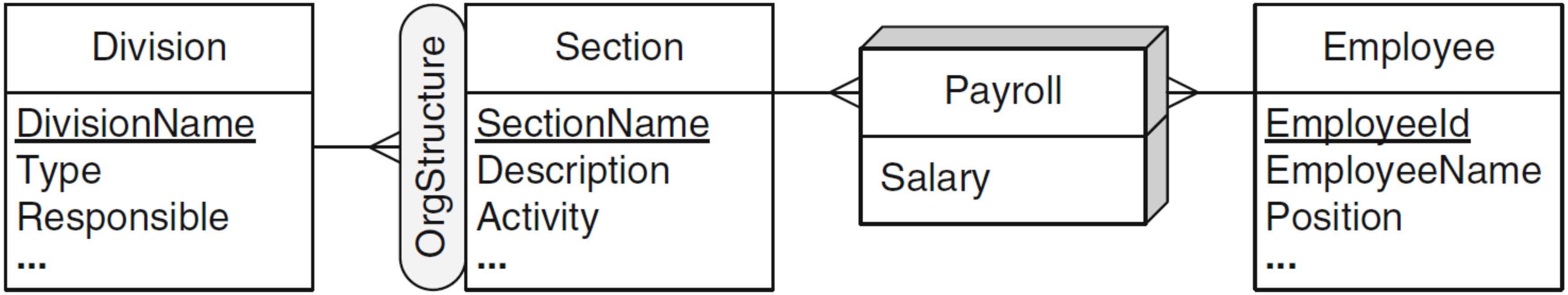
Types of hierarchy

- Encoding a non-strict hierarchy
  - use **bridge table** with percentage (%)
    - distributing attribute



Types of hierarchy

- Encoding non-strict hierarchy
  - Another possible solution is to re-design the DW schema using two separate dimensions



Measures

Measures

- Each measure is associated to an **aggregation function** that combines several values into a single one
  - the aggregation takes place whenever we change to a different level in a dimension hierarchy
- When defining a measure we must decide the associated aggregation function
  - **SUM** is the most typical, but it may not always apply
  - some aggregation functions may not apply to a measure, or to a measure on a certain dimension

Measures

- Additive measures
  - Facts can be aggregated along all dimensions using addition (sum)
    - e.g. sales amount along customer, product and time
- Semi-additive measures
  - Facts can be added along some, but not all dimensions
    - e.g. inventory level cannot be summed along time
- Non-additive measures
  - Facts cannot be added along any dimension
    - e.g. unit price, exchange rate

Measures

- What to do about semi- or non-additive measures
  - use other forms of aggregation
    - average (e.g. average inventory level over time)
    - minimum (e.g. minimum exchange rate over space or time)
    - maximum (e.g. maximum unit price over space or time)

Measures

- Derived measures
  - Can be computed from other measures or attributes
    - e.g. given two measures: **sales amount** and **tax amount**
    - then **net amount** can be derived as a third measure
      net amount = **sales amount** – **tax amount**

Time Dimension

Time and Date Dimension Tables

- **Date** and **Time** dimensions: Typically created separately in the very beginning of data warehouse project
  - Represent the time axis of the events being captured
  - Referenced by multiple fact tables
  - SKs can be created by a formula because the values of the Natural Keys do not “change"

Will be used to rollup and filter facts

⚠ Never load the date dimension from of operational tables!

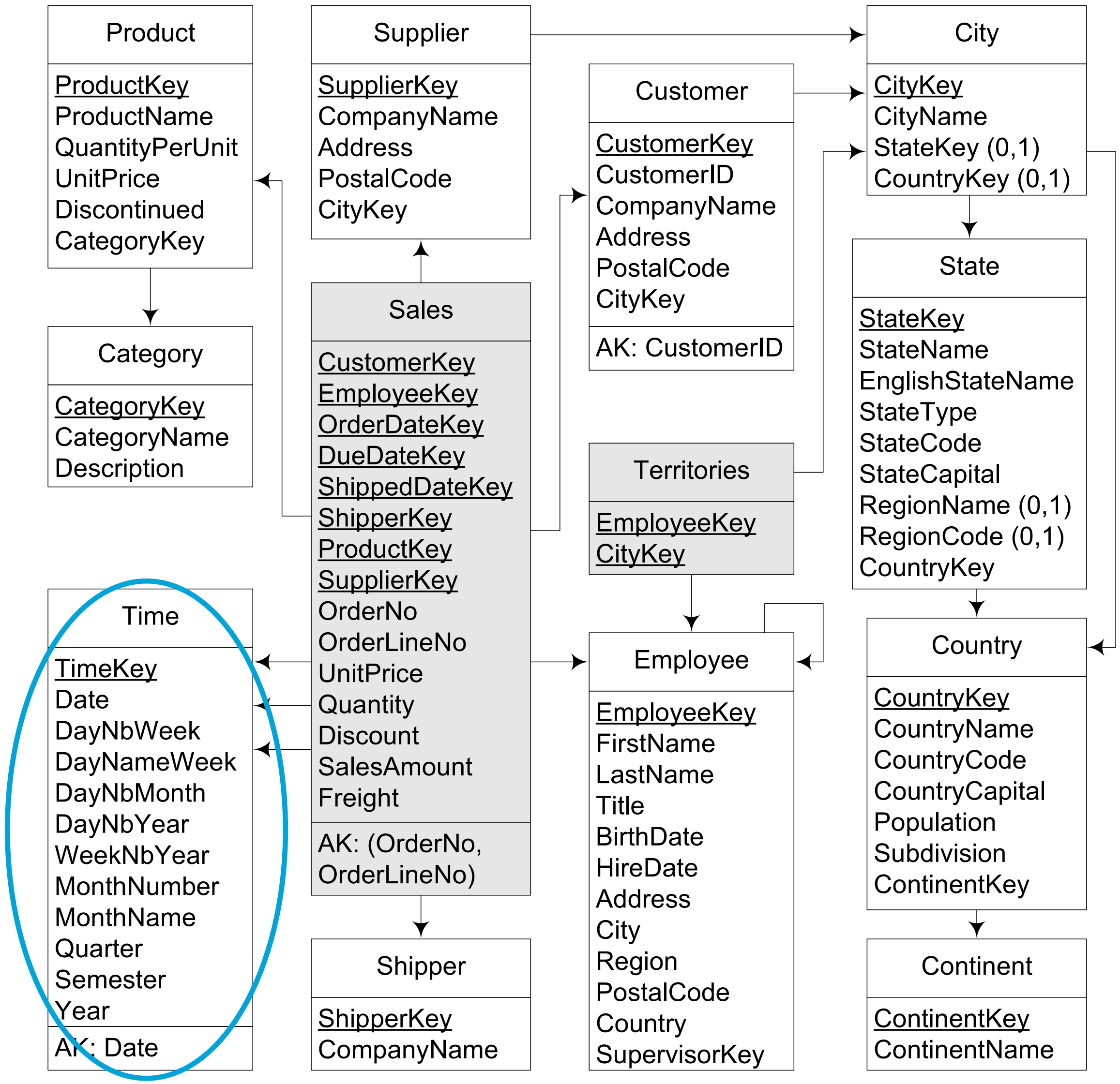
Time dimension

- A data warehouse is a historical database so the time dimension is present in almost all DWs
  - in star/snowflake schema, time is included both as a foreign key in the fact table and as a time dimension containing the associated hierarchy levels

- In transactional databases, time information is stored in attributes of a DATE data type
  - e.g. weekend is computed on-the-fly using appropriate functions

- In a data warehouse, time information is stored explicitly in multiple attributes in the time dimension
  - easier to compute queries, e.g. total sales during weekends

Time dimension



Time dimension

- Granularity of time dimension depends on use
  - If we are interested in monthly data only, define the time dimension with granularity of month
  - If the granularity is second, then the dimension time for 5 years will have: 5*12*30*24*60*60 = 155 520 000 tuples
    - To solve this we can have separate date and time dimensions
- Time dimension may have more than one hierarchy
  - e.g. fiscal and calendar year
- Time dimension can often be populated automatically