Data Warehouse

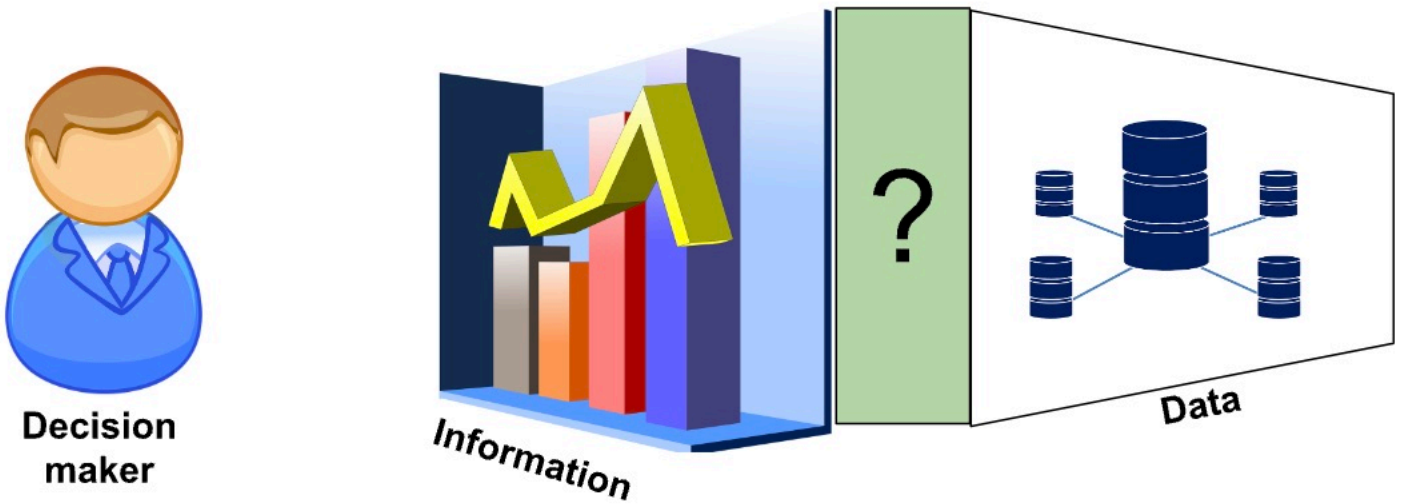
Danilo Montesi
Stefano Giovanni Rizzo

Operational Database

- The kind of database we have seen so far is "**operational**"
  - It's made to support **software applications**
  - It's made to be **updated** in real time (add, change or delete data), not just for viewing
  - It needs **complex** queries to make data analysis and exploration (and many computationally-expensive **joins** operations to join data in a single table)
  - It needs **technical** knowledge of the database schema
  - It may not contain **historical data**, because it doesn't need it for application to work!

Data-driven decision making

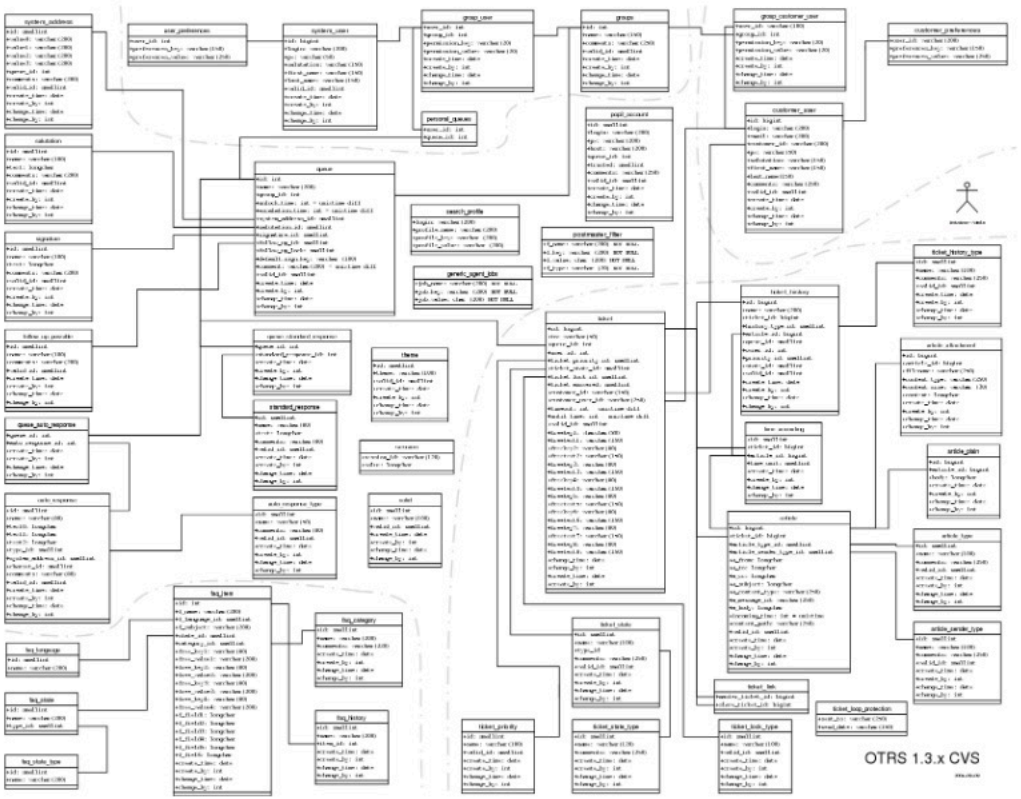
- Operational databases are more than fine for software data, but in a **business** context, data should also become a rich source of **information**:
  - It represents the business situation and behaviour
  - It can support **informed decisions** for the business decision makers



Decision making from a database

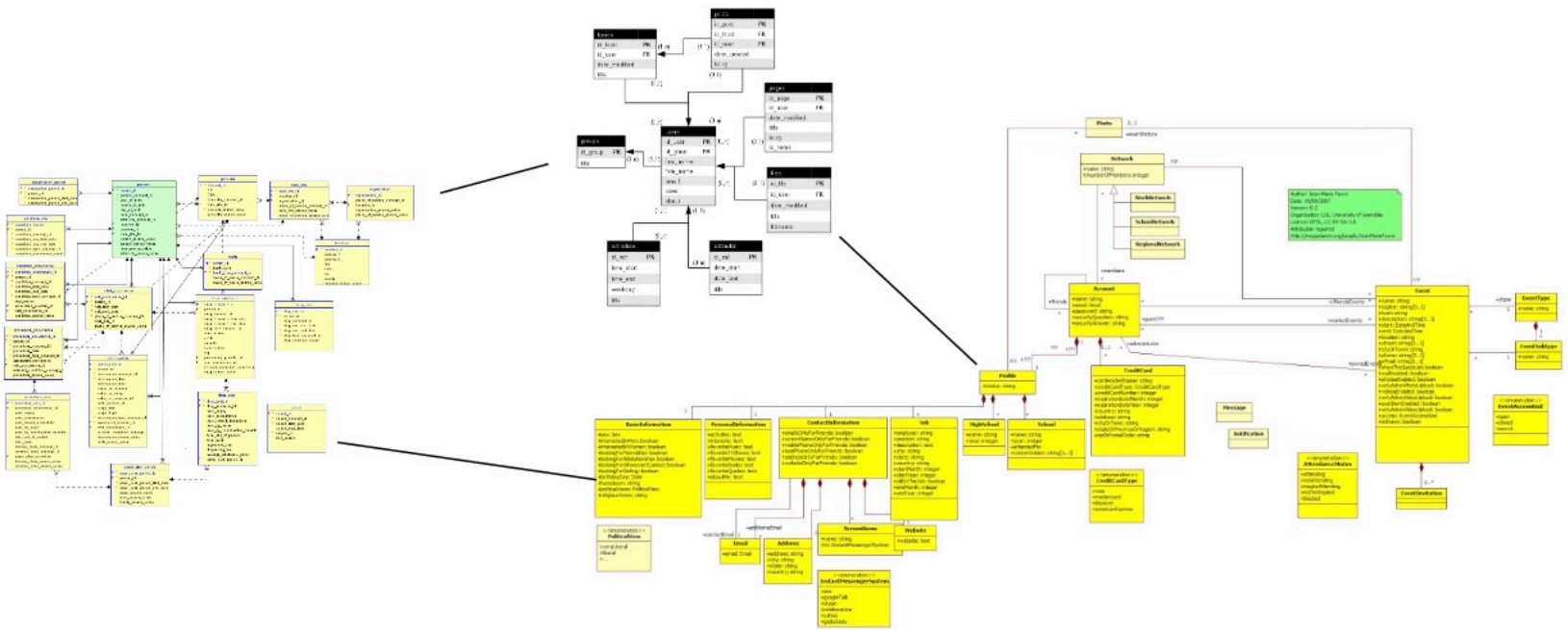
- Decision makers cannot directly **benefit** from a database's data without a deep and technical knowledge of the **data model**

Decision maker



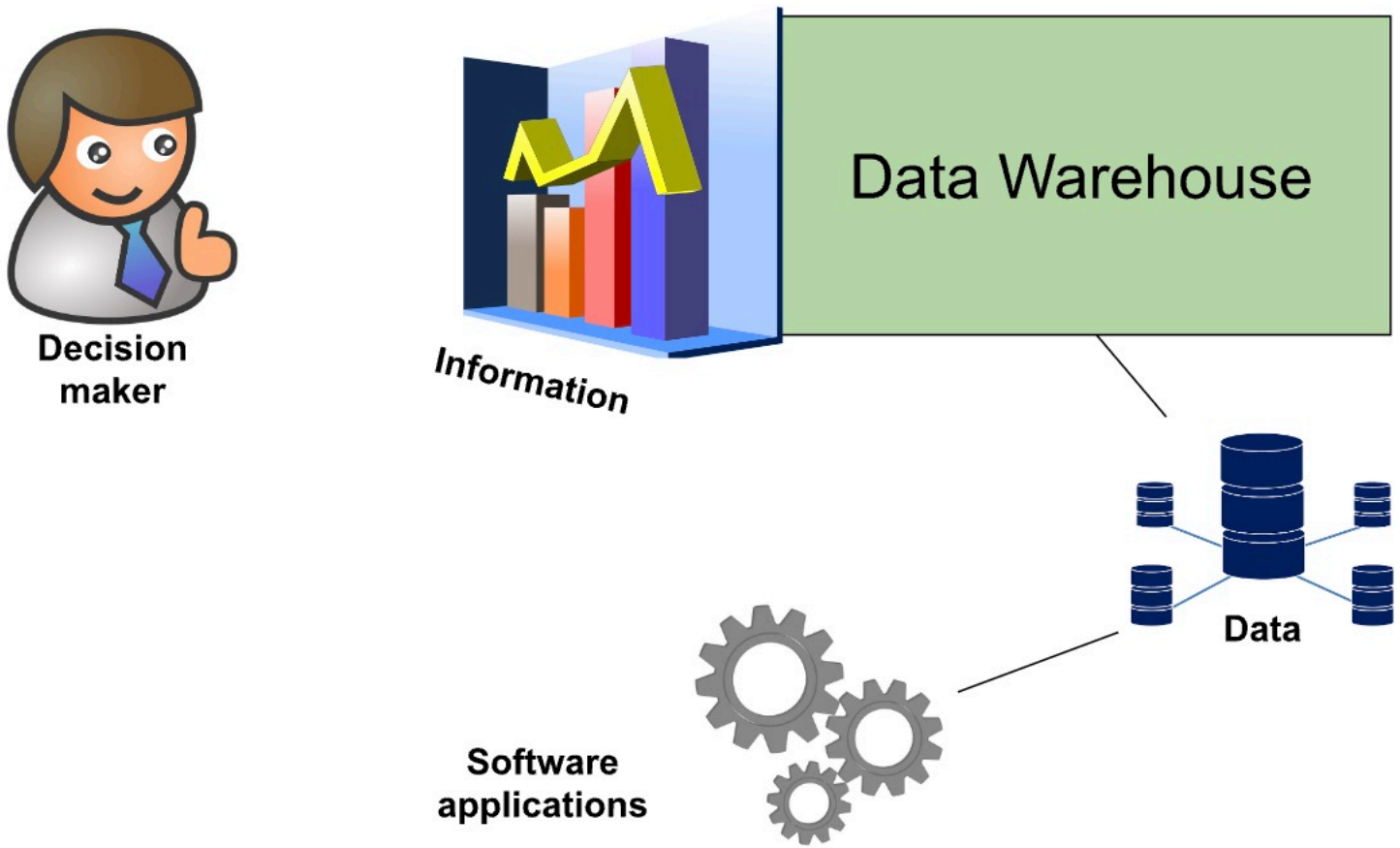
Data integration

- Often decision makers want to **put together** data from different operational databases
  - Example: Sales data with advertising data
- Integrating different operational databases is **extremely** difficult: they have different data definitions and content



Data warehouse

Data warehouse (DW): **collection** of **subject-oriented**, **integrated**, nonvolatile, and time-varying data to support management **decisions**



Data warehouse properties

More in details, a Data Warehouse is:

- **Subject-oriented**: a data warehouse targets one or several subjects of analysis according to the analytical requirements of managers at various levels of the decision-making process
- **Integrated**: the content of a data warehouse result from the integration of data from various operational and external systems
- **Nonvolatile**: a data warehouse **accumulates** data from operational systems for a long period of time → **modification and removal are not allowed** the only operation is the purging of no longer needed, obsolete data
- **Time-varying**: a data warehouse keeps track of how its data has **evolved over time**; for instance, it may allow one to know the evolution of sales or inventory over the last months/years

Design of databases

- Typically performed in four phases:
  - **Requirements specification:** needs of users are collected to create a database schema
  - **Conceptual design**: describes the data with entity-relationship (ER) model
  - **Logical design**: implementation paradigm for database applications→ relational model
  - **Physical design:** specific implementation over a DBMS

- The final result is a **relational database**: highly normalized to guarantee consistency under frequent updates, usually achieved at a higher cost of querying (normalization produces multiple tables)

Design of a Data Warehouse

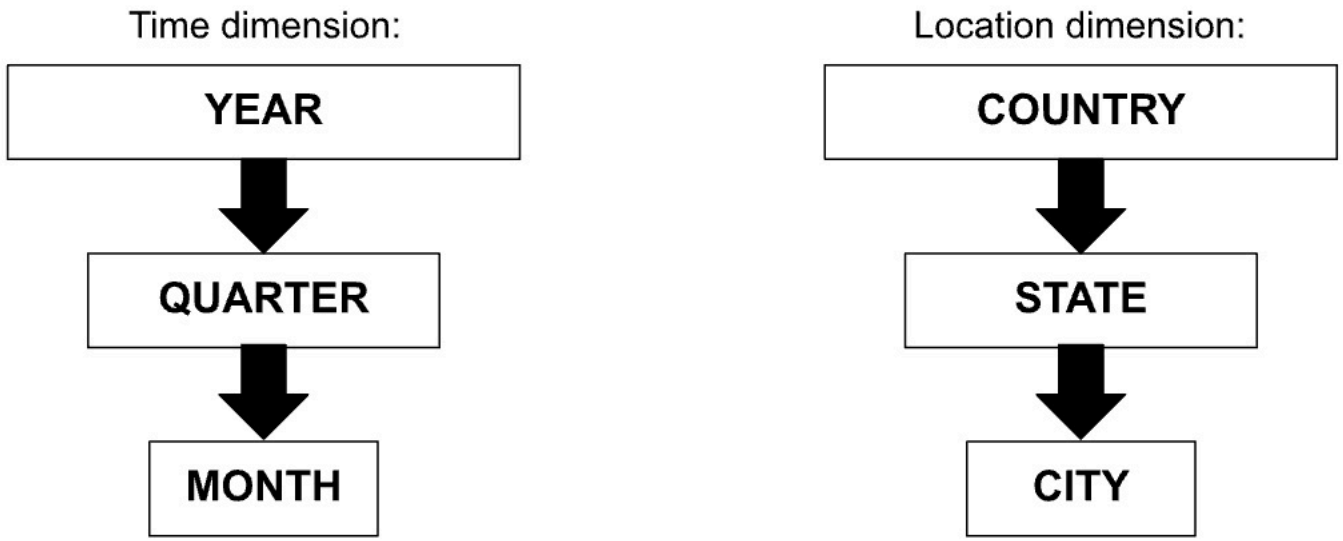
- The relational paradigm is **not appropriate** for data warehouses: we need good performance for complex query in data analysis.
- We need a design technique that:
  - It's intended to support **end-users** (decision makers) queries
  - It's oriented around **understandability** of the results, without the need of knowing a complex data schema
  - It's oriented to **performance**, redundancy is accepted if brings a performance advantage, less degree of normalization is required
- The paradigm used to design a data warehouse is called **multidimensional modeling** (or simply dimensional modeling, DM)

Multidimensional modeling (1/2)

- The multidimensional modeling views data as consisting of **facts** linked to **dimensions**
- A **fact** represents the focus of analysis, the main object we are interested into (e.g., in the analysis of sales in stores, the fact is the sale).
- **Measures** quantify facts: usually measures are **numeric** values (e.g. the amount of sales, number of iterms, etc)
- **Dimensions** are used to view measures from several perspectives, for example:
  - **Time dimension**: to analyze changes in sales over specific periods of time
  - **Location dimension**: to analyze sales according to the geographic distribution of stores

Multidimensional modeling (2/2)

- Dimensions include attributes that form **hierarchies**, which allow decision making users to explore measures at various **levels of detail**, for example:



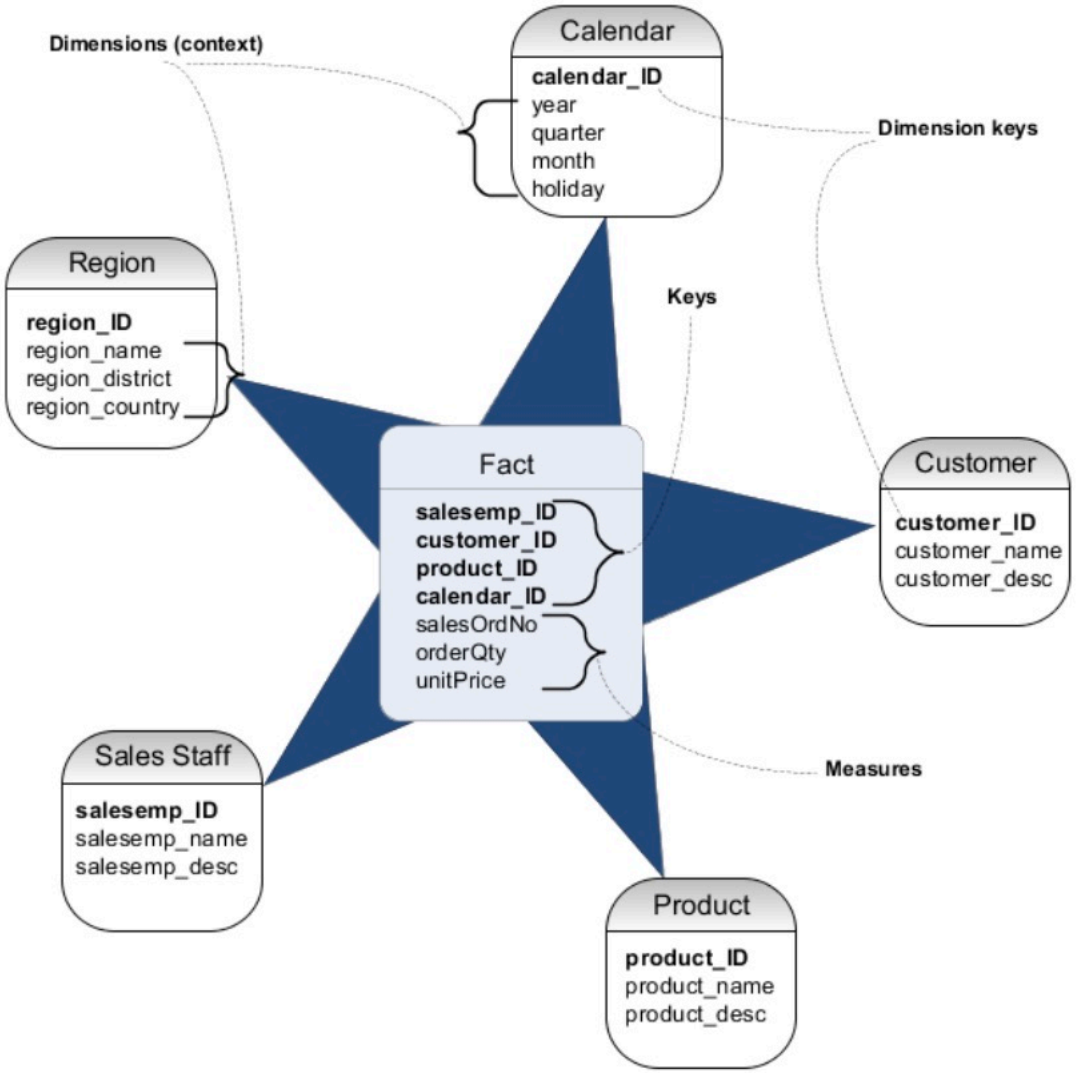
- **Aggregation** of measures occurs every time a hierarchy is traversed, e.g., moving from month to year will aggregate values of sales for each year

Star and snowflake schemas

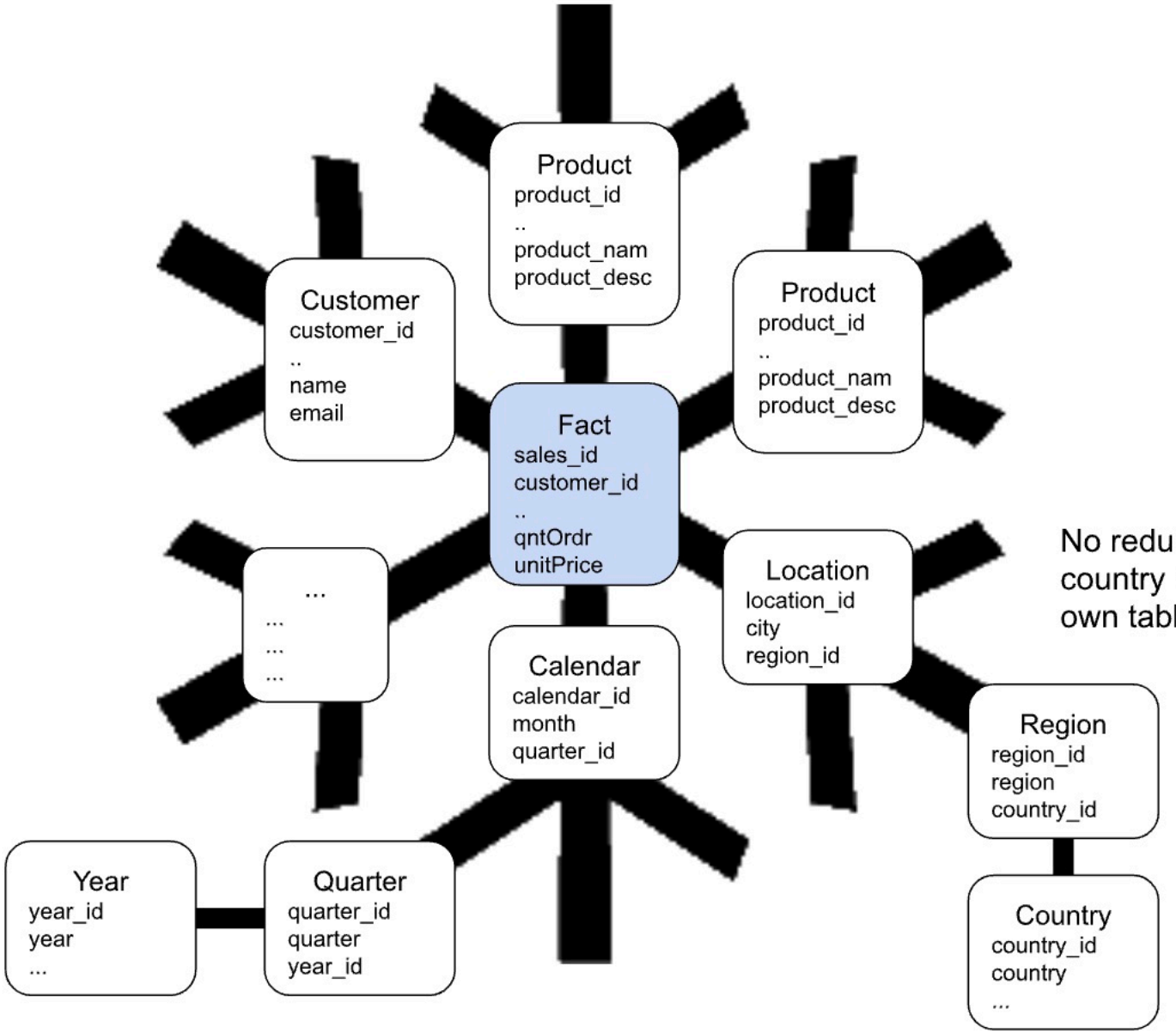
- The multidimensional modeling is a **conceptual** model for data warehouses.
- At the **logical** level, the multidimensional model is usually represented by relational tables organized in **star schemas** and **snowflake schemas**.
  - These schemas put the **fact table in the center**, linked to several dimension tables
  - **Star** schemas use a **unique** table for each dimension, even in the presence of hierarchies (**denormalized** table, because of redundancies)
  - **Snowflake** schemas use **multiple normalized** tables the hierarchies of a dimension

Star schema

Redundancy in star schema: country will be repeated for all the regions in the country



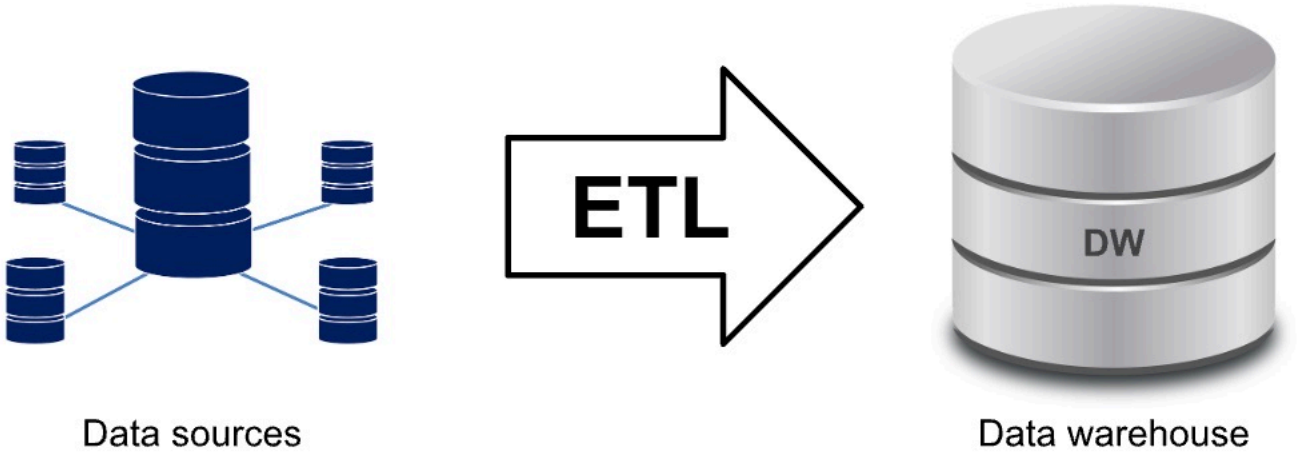
Snowflake schema



No redundancy: country has its own table

Populating a DW: ETL

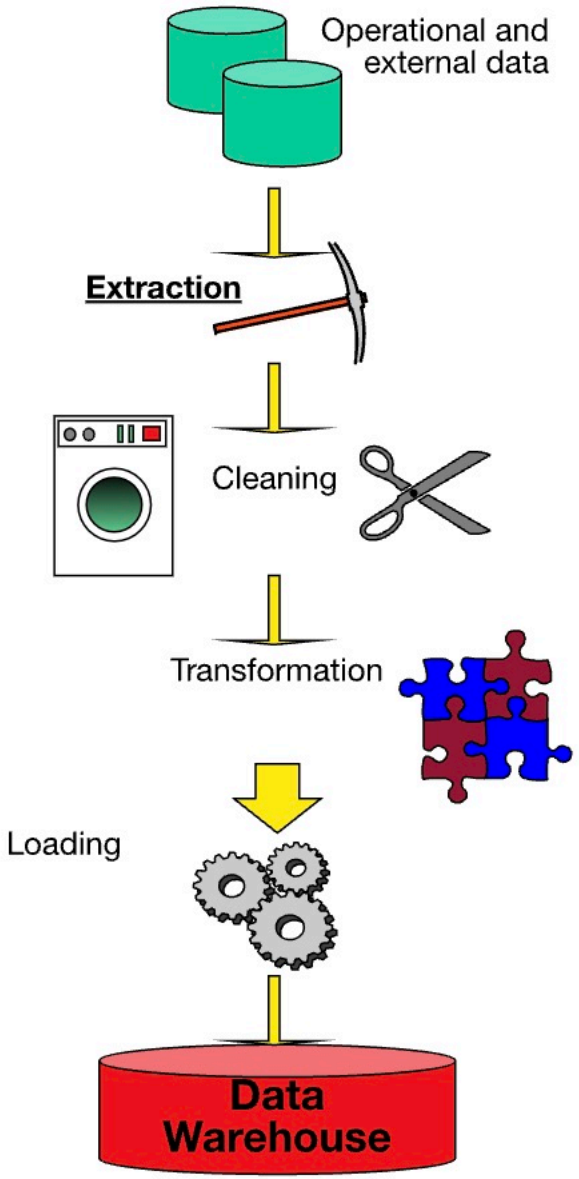
- ETL stands for **Extraction, Transformation and Loading**.
- It's a crucial process for the success of a data warehousing project and it goes through 3 steps:
  - **Extract** data from several source systems
  - Clean and **Transform** data to fit the data warehouse model
  - **Load** transformed data into the data warehouse



- Requires great efforts: ETL is estimated to be the 80% of the total DW cost!!

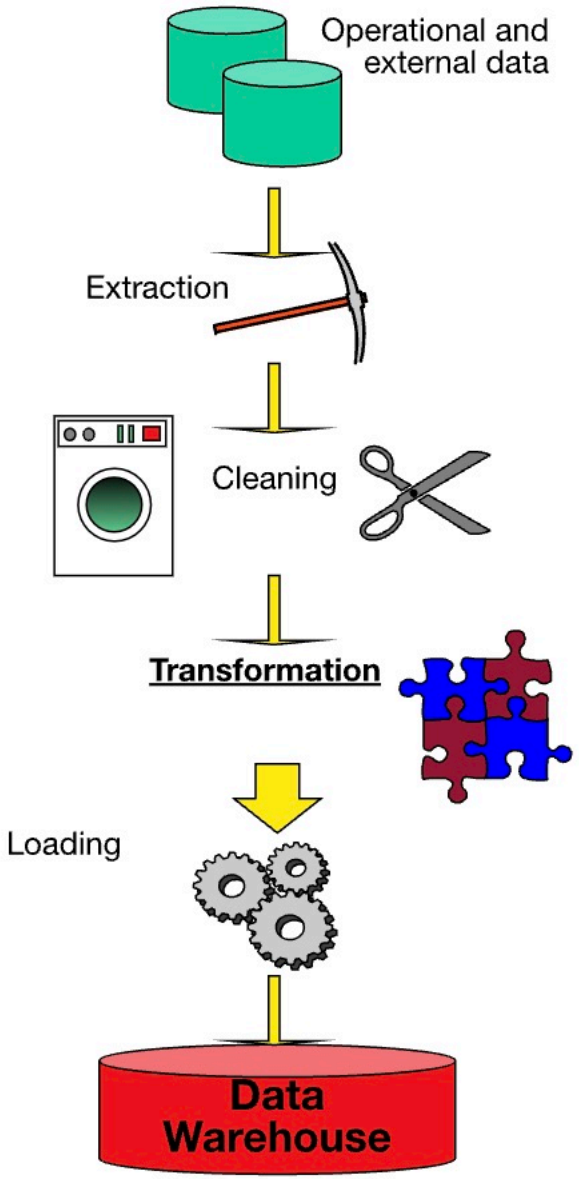
ETL: Extraction

- Relevant data is extracted from the source.
  - **Static** extraction is carried out when the DW must be populated for the first time, it basically consists of making a “snapshot” of the operational data.
  - **Incremental** extraction is used to update the DW, and it captures only the changes with respect to the last extraction
    - Based on the DBMS log
    - Based on timestamps
- The selection on what data should be extracted depends mainly on its quality.



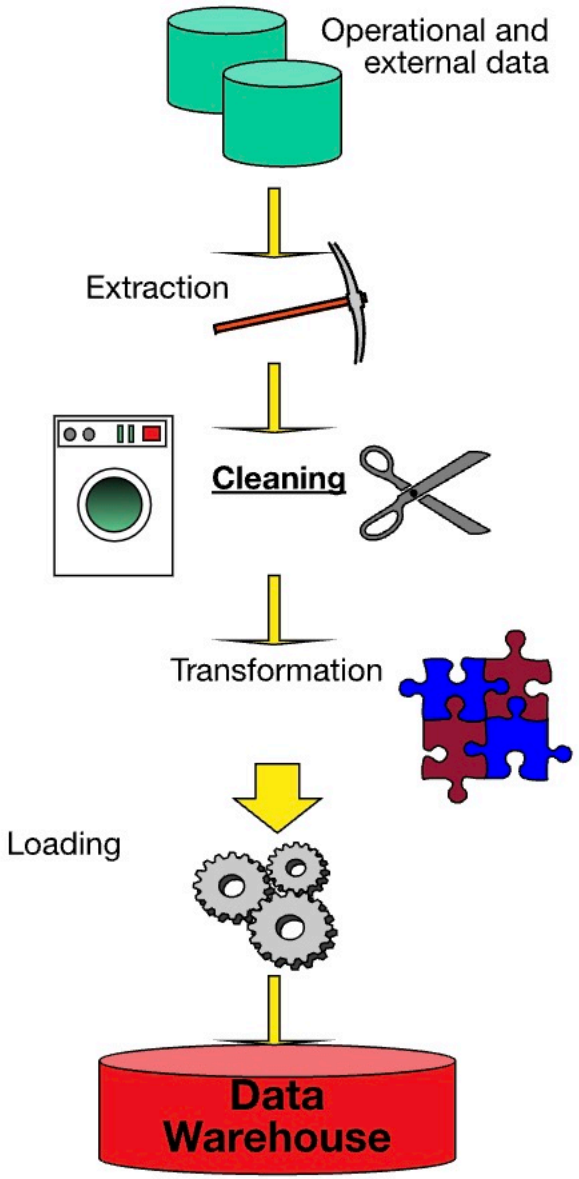
ETL: Transformation (1/2)

- The process converts the data from the operational format to the DW format. The correspondence with the source level is made more complex by the heterogeneity of the different sources, that requires a complex integration step.
  - Presence of free text data, hiding important information
  - Use of different formats for the same data

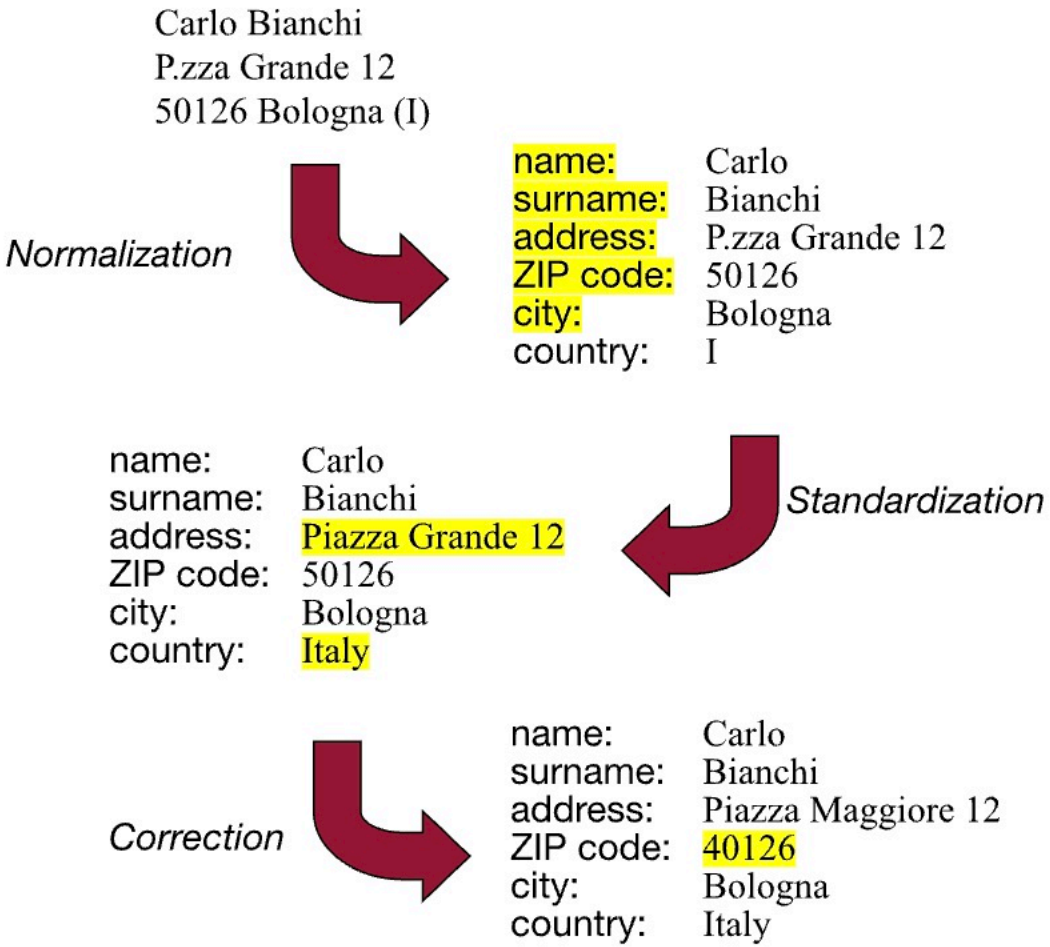


ETL: Cleaning

- The cleaning process has the goal of improving the data quality of the sources
  - Duplicate data
  - Inconsistency between associated values
  - Missing values
  - Misuse of a field
  - Impossible or erroneous values
  - Inconsistent values for the same entity because of different standards
  - Inconsistent values for the same entity given by typing errors



Cleaning&Transformation: Example



OLTP Systems

- **Traditional database** systems designed and tuned to support the day-to-day **operations**:
  - ensure fast, concurrent access to data
  - transaction processing and concurrency control
  - focus on online update data consistency
  - known as operational databases or online transaction processing (**OLTP**)
- **OLTP** DB data characteristics:
  - **Detailed** data
  - Do **not include historical** data
  - Highly **normalized**
  - Poor performance on complex queries including **joins** an aggregation

- **Data analysis** requires a **new paradigm**: **online analytical processing** (**OLAP**)
  - Typical OLTP query: pending orders for customer c1
  - Typical OLAP query: total sales amount by product and by customer

OLAP Systems

- The **online analytical processing** (**OLAP**) characteristics:
  - While OLTP paradigm focused on transactions, OLAP focused on **analytical queries**
  - Normalization not good for analytical queries, reconstructing data requires a high number of joins (denormalized tables in OLAP)
  - OLAP databases support a heavy query load
  - OLTP indexing techniques not efficient in OLAP: oriented to access few records
  - OLAP queries typically include aggregation

- The need for a different database model to support OLAP was clear: led to data warehouses

Comparison: OLTP vs OLAP

- On many aspects, there are several differences between OLTP and OLAP systems:

	OLTP	OLAP
User type	Operators, office employees	Managers, executives
Usage	Predictable, repetitive	Ad hoc, nonstructured
Data content	Current, detailed data	Historical, summarized data
Data organization	According to operational needs	According to analysis needs
Data structures	Optimized for small transactions	Optimized for complex queries
Access frequency	High	From medium to low
Access type	Read, insert, update, delete	Read, append only
Number of records per access	Few	Many
Response time	Short	Can be long
Concurrency level	High	Low
Lock utilization	Needed	Not needed
Update frequency	High	None
Data redundancy	Low (normalized tables)	High (denormalized tables)
Data modeling	UML, ER model	Multidimensional model

DW: Recap

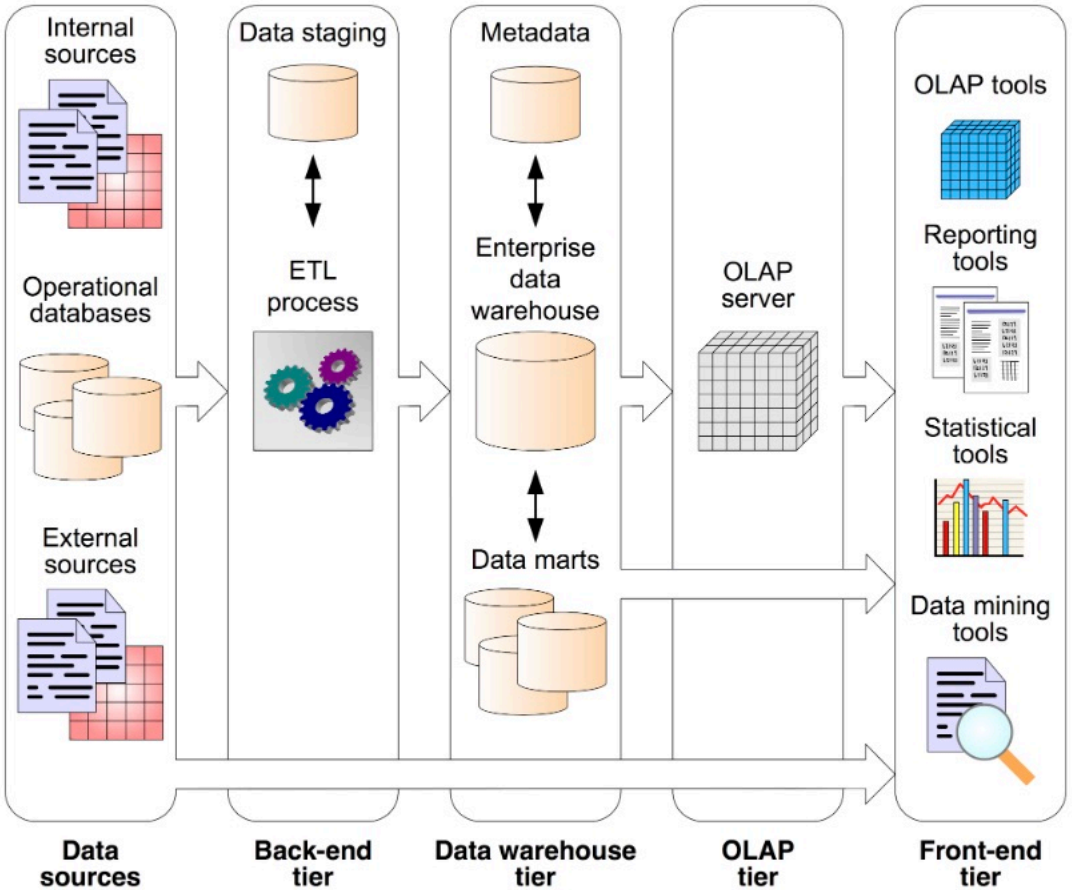
- Data warehouses:
  - large **repositories** that consolidate data from **different** sources (internal and external to the organization),
  - are updated **offline**,
  - follow the **multidimensional data model**,
  - implemented using **star or snowflake** schemas
  - to **efficiently** support **OLAP queries**

- We will now dive deeper into these concepts to better understand the **architecture**, **design,** and **querying** aspects of a data warehouse.

Architecture: Multiple tiers

Tiers 1/3

- Despite many different architectures have been proposed, a common trait is the presence of **several tiers**.



Tiers 2/3

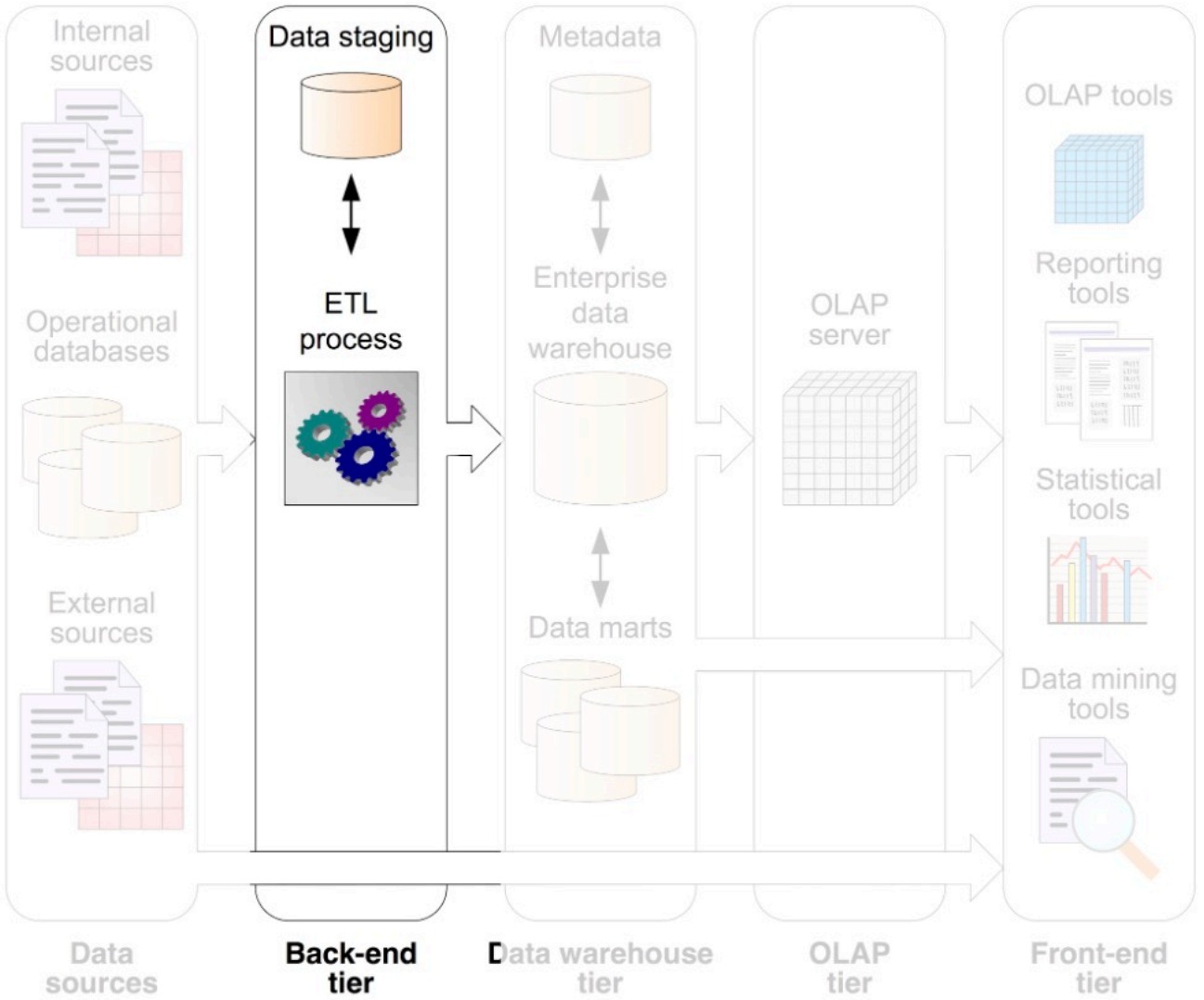
- **Data source** is not a tier of a data warehouse, but more of a collection of data of different kind (sometimes referred as *data lake*)

- **Back-end** tier composed of:
  - The **extraction, transformation, and loading** (ETL) tools: Feed data into the data warehouse from operational databases and internal and external data sources
  - The **data staging** area: An intermediate database where all the data integration and transformation processes are run prior to the loading of the data into the data warehouse

Tiers 3/3

- **Data warehouse** tier composed of:
  - An enterprise data warehouse and/or several data marts
  - A metadata repository storing information about the data warehouse and its contents
- **OLAP tier** composed of:
  - An OLAP server which provides a multidimensional view of the data, regardless the actual way in which data are stored
- **Front-end tier** is used for data analysis and visualization
  - Contains client tools such as OLAP tools, reporting tools, statistical tools, and data-mining tools

Back-end Tier 1/3



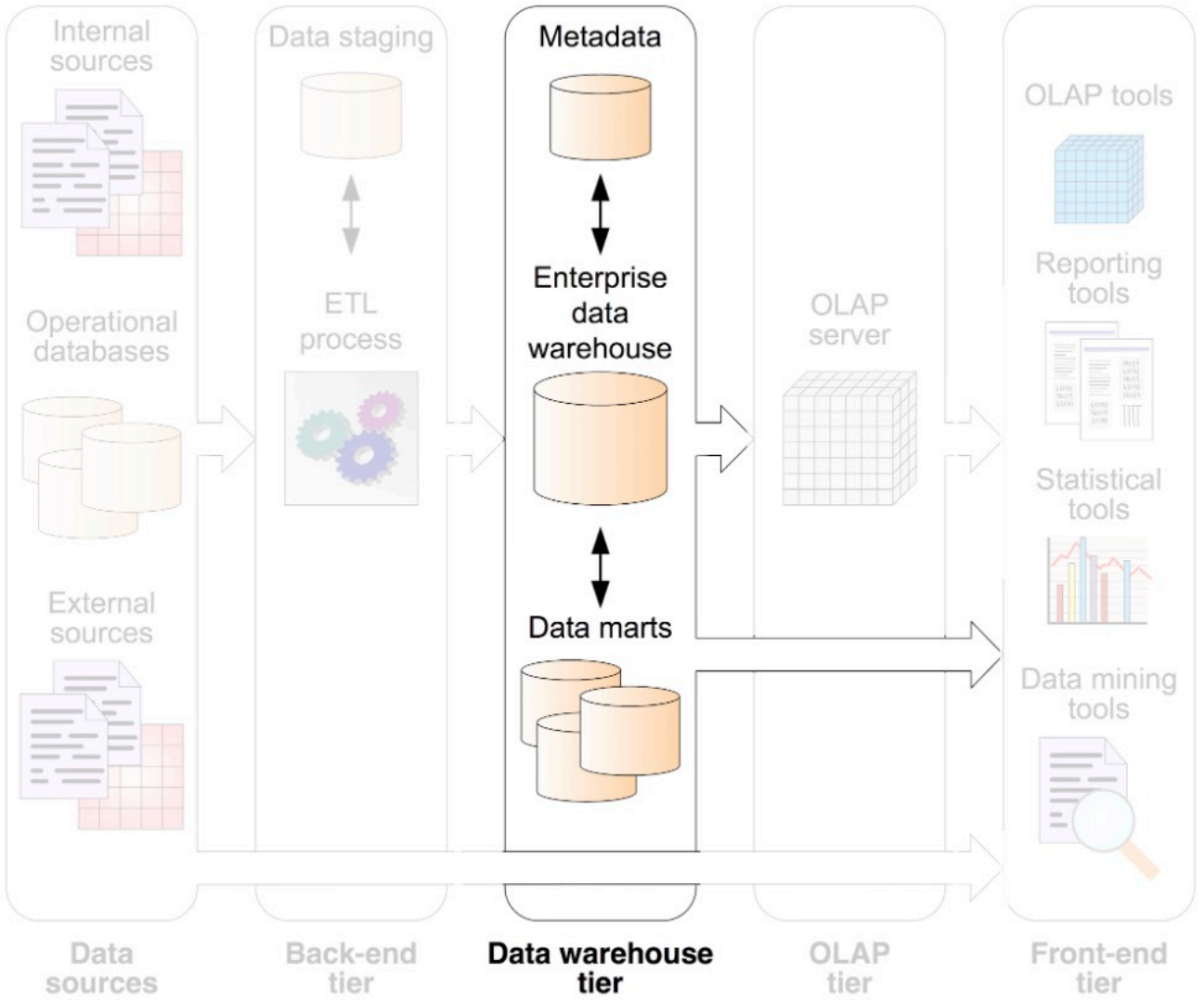
Back-end tier 2/3

- Performs Extraction, Transformation, and Loading. It is a three-step process:
  - **Extraction** gathers data from multiple, heterogeneous data sources internal or external to the organization
  - **Transformation** modifies the data from the format of the data sources to the warehouse format; this includes:
    - **Cleaning**: Removes errors and inconsistencies in the data and converts it into a standardized format.
    - **Integration**: Reconciles data from different data sources, both at the schema and at the data level
    - **Aggregation**: Summarizes the data obtained from data sources according granularity of the data warehouse
  - **Loading** feeds the data warehouse with the transformed data, including refreshing the data warehouse, that is, propagating updates from the data sources to the data warehouse at a specified frequency

Back-end tier 3/3

- Back-end tier has also a **data staging area** (usually called operational data store): A **database** where data extracted from the sources undergoes **successive modifications** before being loaded into the data warehouse

Data Warehouse Tier 1/3



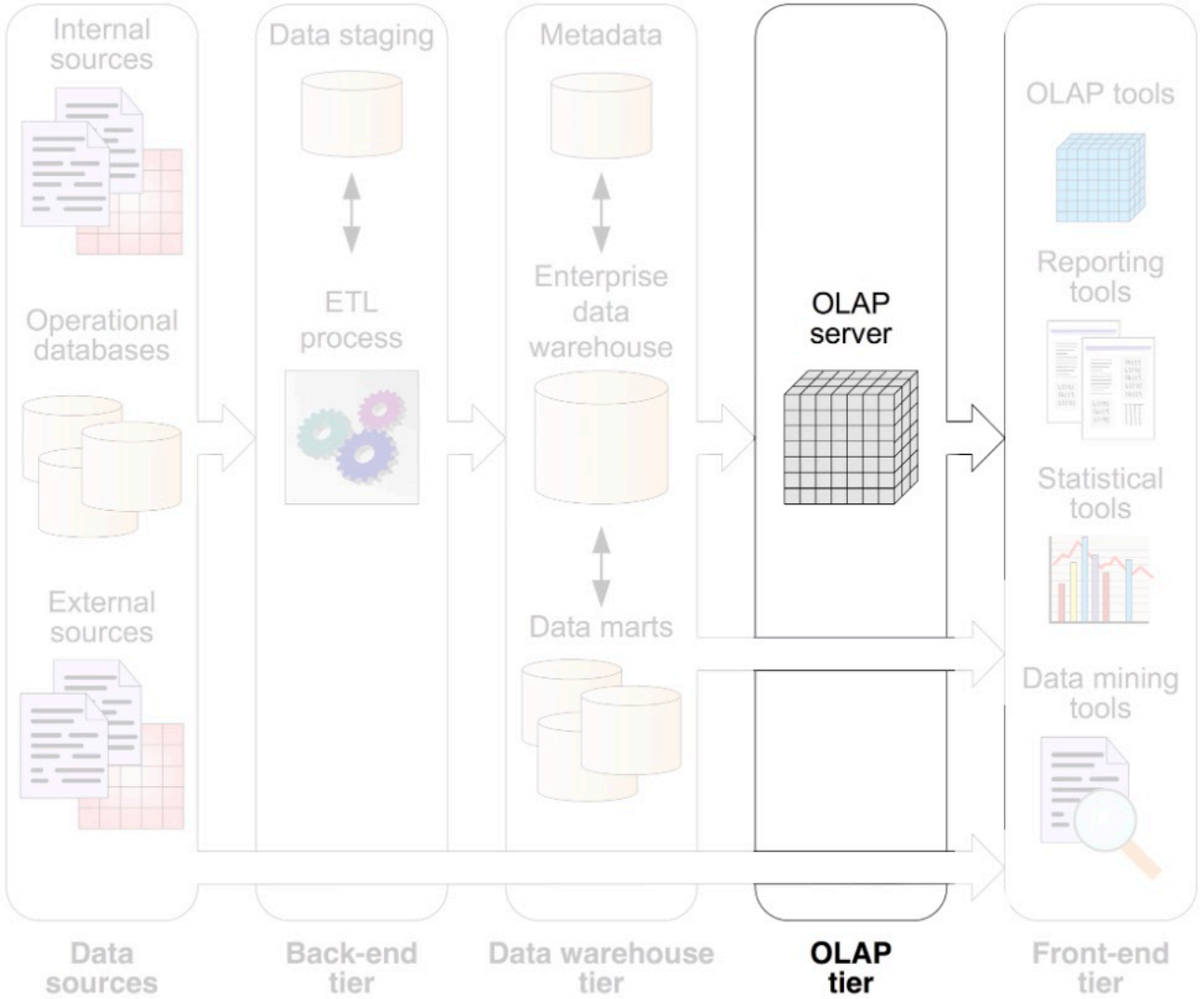
Data Warehouse tier 2/3

- **Components**:
  - An **enterprise data warehouse**, centralized and encompassing an entire organization
  - Several **data marts**: specialized departmental data warehouses
- **Metadata**
  - **Business metadata** describes the semantics of the data, and organizational rules, policies, and constraints related to the data
  - **Technical metadata** describes how data are structured and stored in a computer system, and the applications and processes that manipulate the data.

Data Warehouse tier 3/3

- The **metadata repository** of the data warehouse tier may contain information such as:
  - Metadata describing the **structure** of the data warehouse and the data marts, at the conceptual/logical level (facts, dimensions, hierarchies, ...) and at the physical level (indexes, partitions,...)
  - **Security information** (user authorization and access control), and monitoring information (usage statistics, error reports, audit trails)
  - Metadata describing **data sources**: schemas, ownership, update frequencies, legal limitations, access methods
  - Metadata describing the **ETL**: data lineage, data extraction, cleaning, transformation rules, etc.

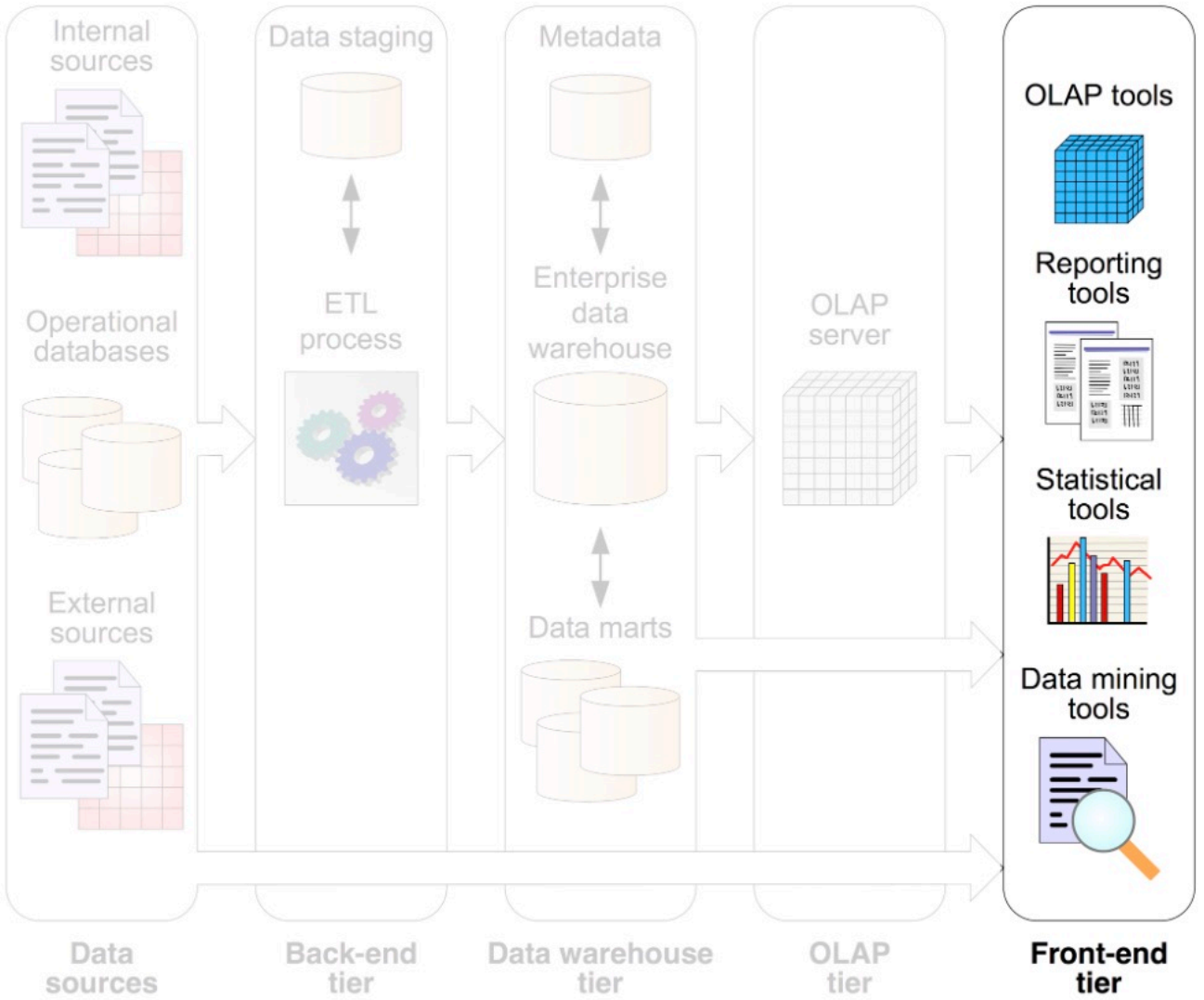
OLAP Tier 1/2


Internal sources
Operational databases
External sources
Data staging
ETL process
Metadata
Enterprise data warehouse
Data marts
OLAP server
OLAP tools
Reporting tools
Statistical tools
Data mining tools
Data sources
Back-end tier
Data warehouse tier
OLAP tier
Front-end tier

OLAP tier 2/2

- **Components**:
  - An enterprise data warehouse, centralized and encompassing an entire organization
  - Several data marts, specialized departmental data warehouses
- **Metadata**
  - **Business metadata** describes the semantics of the data, and organizational rules, policies, and constraints related to the data
  - **Technical metadata** describes how data are structured and stored in a computer system, and the applications and processes that manipulate the data.

Front-end tier



Front-end tier 2/2

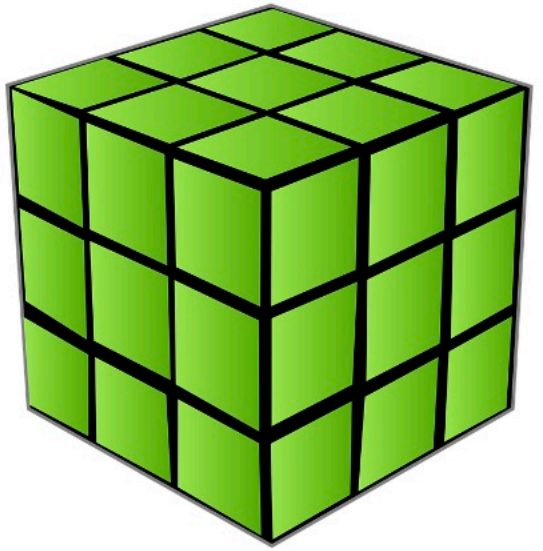
Composed of client tools that allow users to exploit the content of the data warehouse:

- **OLAP tools**: allow interactive exploration and manipulation of the warehouse data and formulation of complex ad hoc queries
- **Reporting tools** enable the production, delivery, and management of reports, which can be paper-based, interactive, or web-based
  - Reports use **predefined queries** asking for specific information in a specific format, performed on a regular basis
- **Statistical tools**: used to analyze and visualize the cube data using statistical methods
- **Data-mining tools** allow users to analyze data in order to discover valuable knowledge such as patterns and trends, and also allow to make predictions based on current data

Design:
Multidimensional model

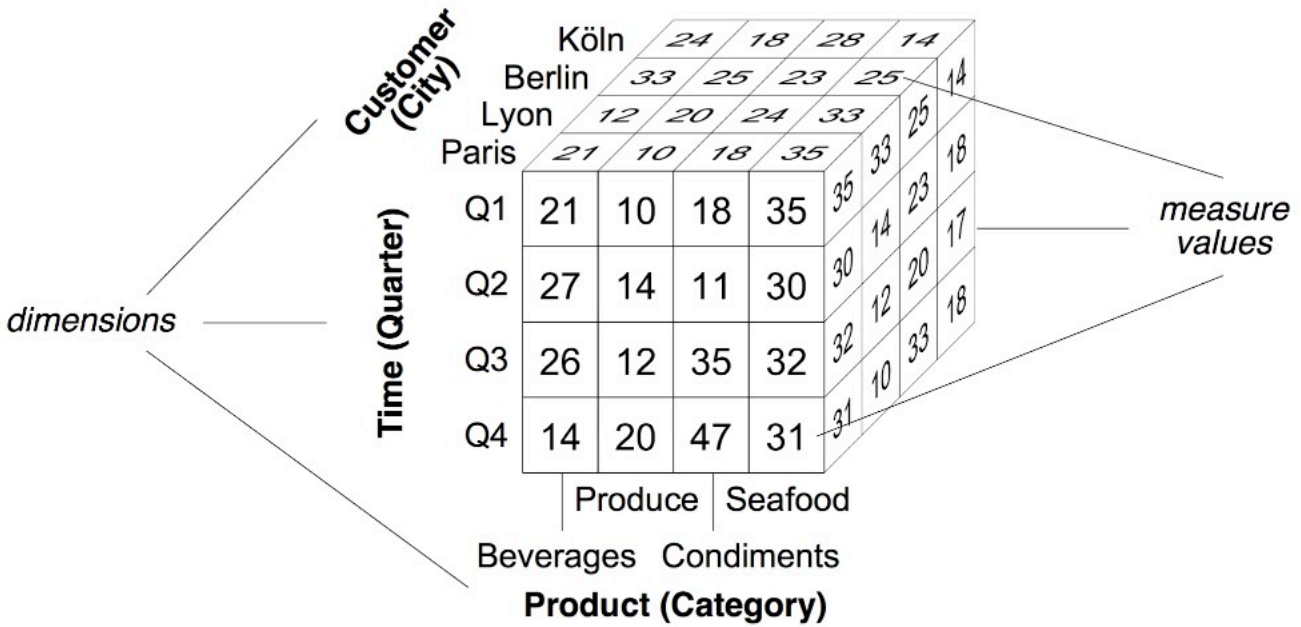
Design: Multidimensional model

- The multidimensional model (MultiDim) views data in an **n-dimensional** space: a data cube.
- A data cube is composed of **dimensions** and **facts**
- Remember that dimensions are **perspectives** used to analyze the data
- Example: A three-dimensional cube for **sales** data with dimensions **Product**, **Time**, and **Customer**, and a measure **Quantity**

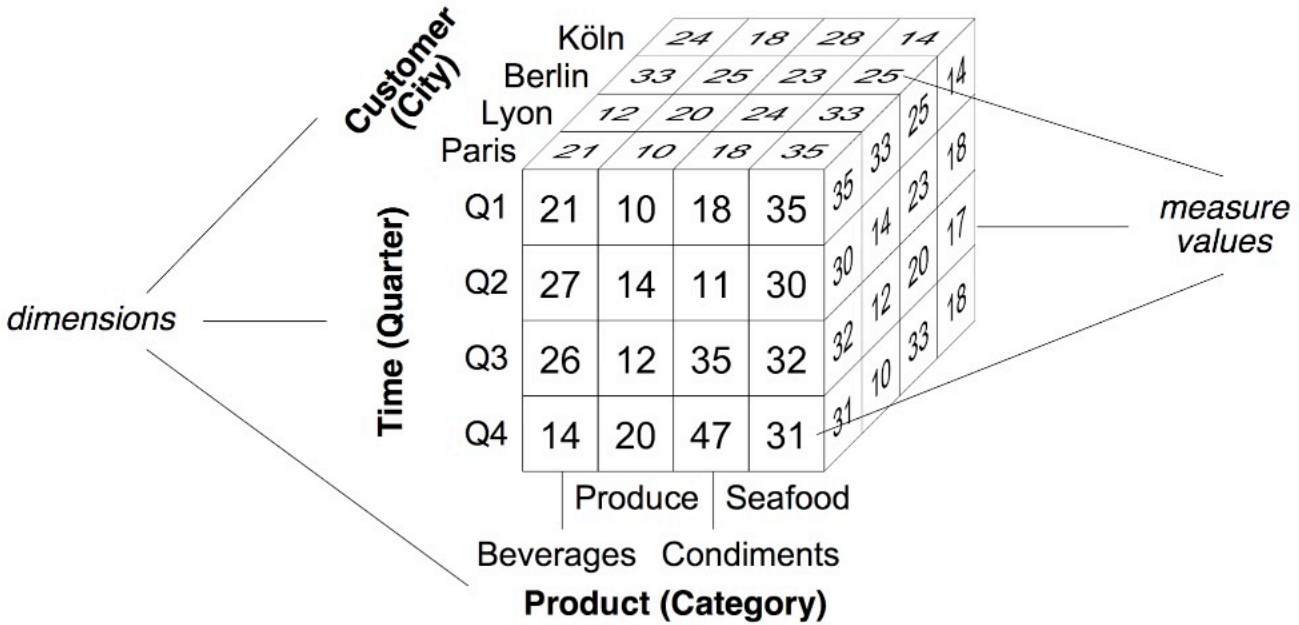


MultiDim: Example 1/2

- **Dimensions**: Customer, Time, Product
- **Facts**: Sales (each “little cube”)
- **Measures**: Quantity



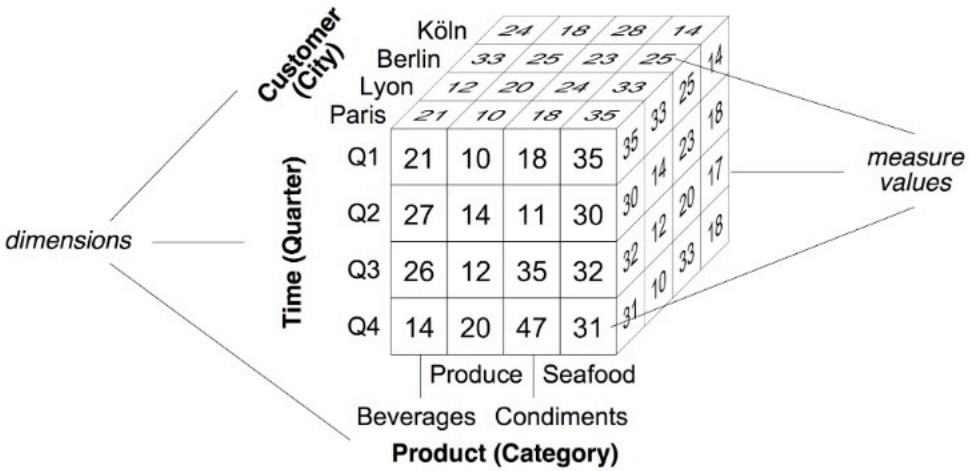
MultiDim: Example 2/2



- **Attributes** describe **dimensions**: Product dimension may have attributes ProductNumber and UnitPrice (not shown in the figure)
- The cells or facts of a data cube have **associated numeric values** called **measures**
- **Each cell** (little cube) **of the data cube represents Quantity of units sold by category, quarter, and customer's city**

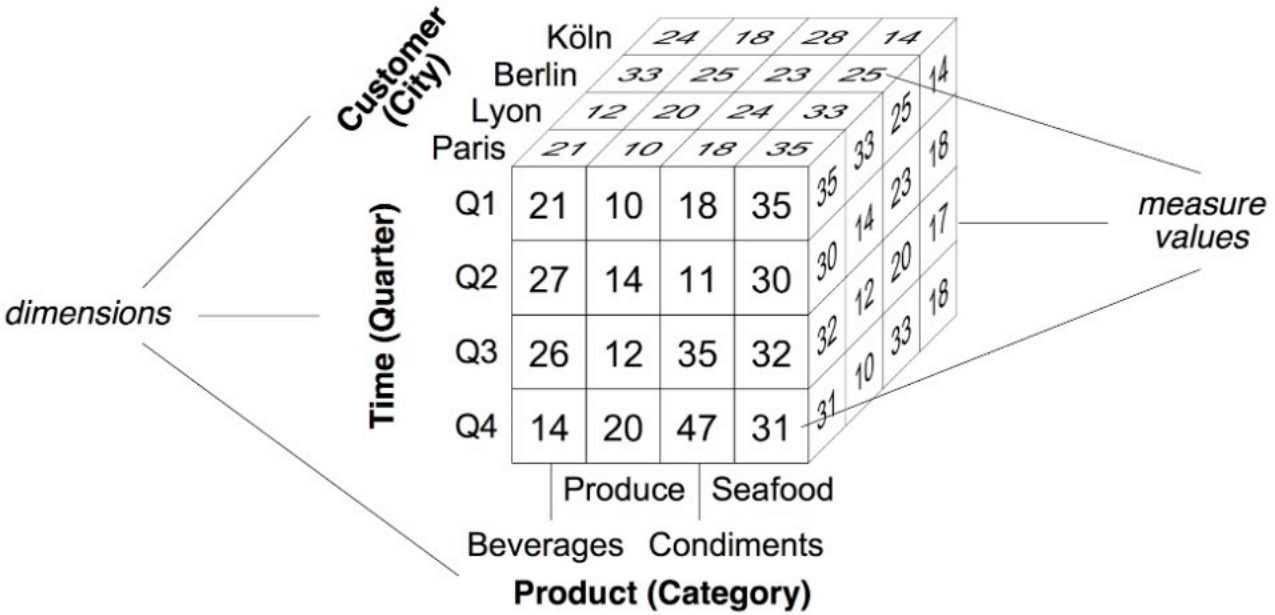
Data granularity

- **Data granularity**: level of detail at which measures are represented for each dimension of the cube
  - sales figures aggregated to granularities *Category*, *Quarter*, and *City*
- **Instances** of a dimension are called **members**
  - *Seafood* and *Beverages* are members of the *Product* at the granularity *Category*
- A data cube contains several measures, e.g. Amount, indicating the total sales amount (not shown) A data cube may be sparse (typical case) or dense
  - not all customers may have ordered products of all categories during all quarters



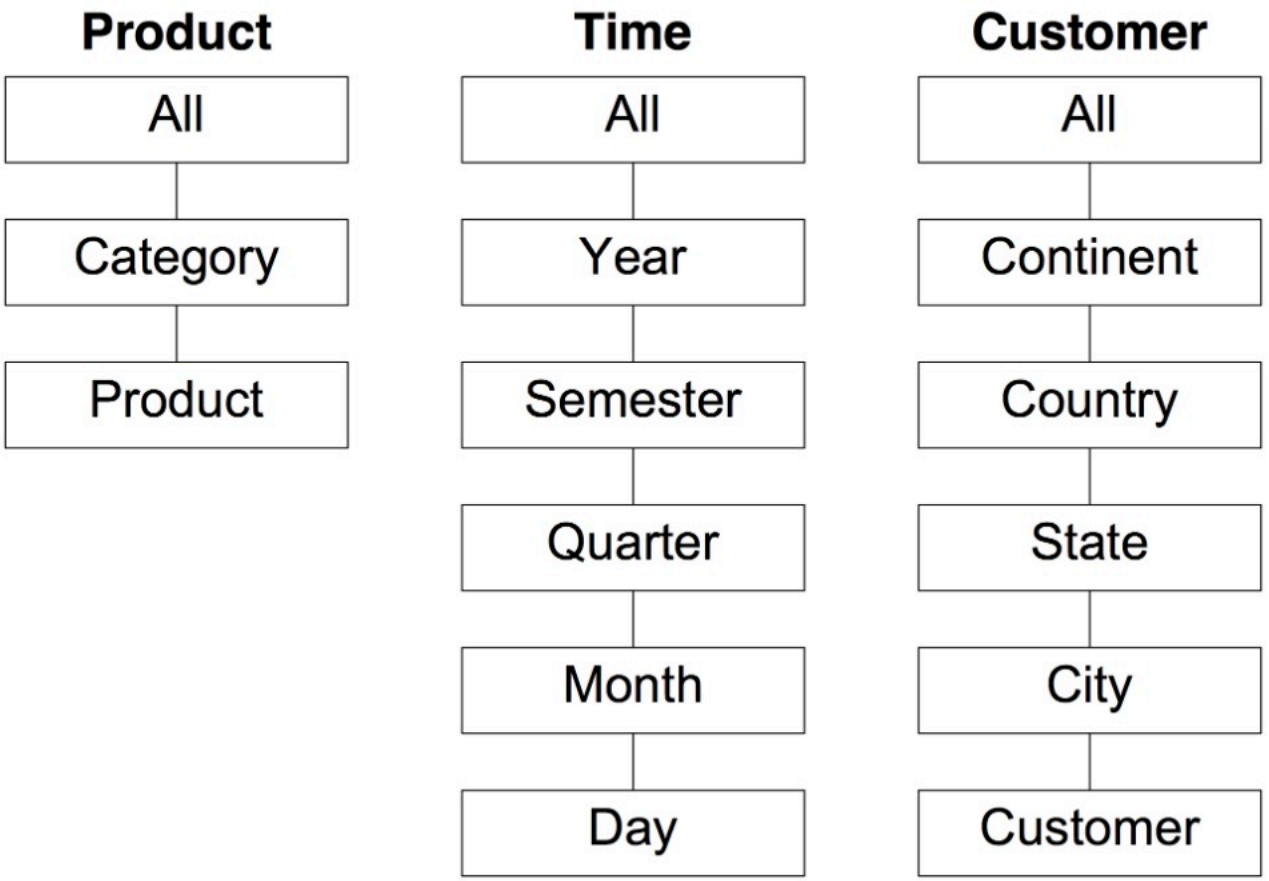
Data granularity: Hierarchies

- **Hierarchies**: allow viewing data at **several granularities**
  - Define mappings relating **lower-level**, detailed concepts to **higher-level** ones
  - The lower level is called the **child** and the higher level is called the **parent**
  - The hierarchical structure of a dimension is called the **dimension schema**
  - A **dimension instance** comprises **all members at all levels** in a dimension
  - In the example, granularity of each dimension indicated between parentheses: *Category* for the *Product* dimension, *Quarter* for *Time*, and *City* for *Customer*
  - We may want sales figures at a **finer granularity** (*Month*), or at a **coarser granularity** (*Country*)



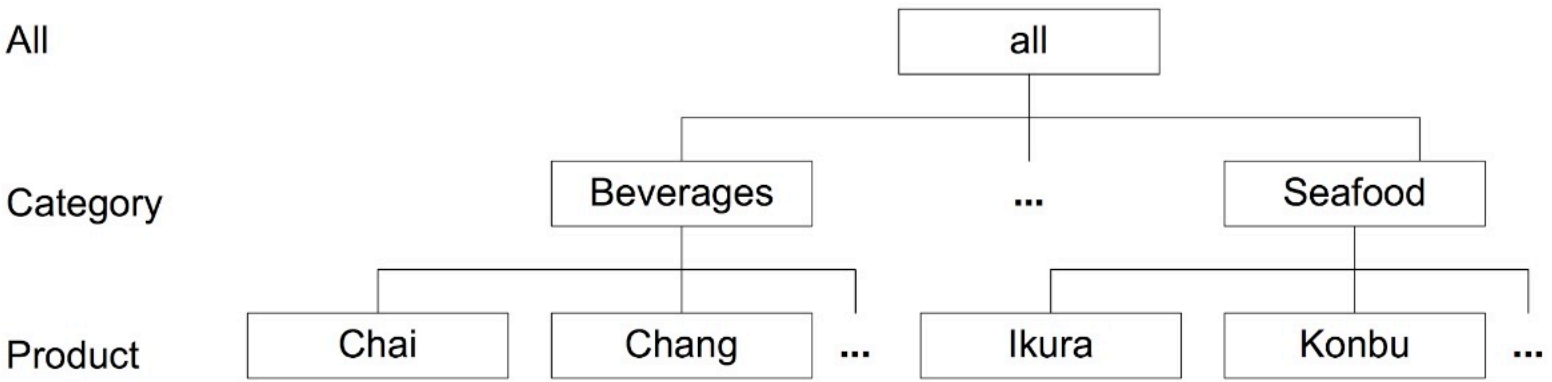
Hierarchies example

- **Hierarchies** of the *Product*, *Time*, and *Customer* dimensions:



Hierarchies members

- In the same example, we can look at the **members** (instances) of the Product hierarchy (Product->Category->All)



Measures and dimensions

- Aggregation of measures changes the abstraction level at which data in a cube are visualized.
- Measures can be:
  - **Additive**: can be meaningfully summarized along all the dimensions, using addition
    - The most common type of measures
  - **Semiadditive**: can be meaningfully summarized using addition along some dimensions
    - Example: inventory quantities, which cannot be added along the Time dimension
  - **Nonadditive:** cannot be meaningfully summarized using addition across any dimension
    - Example: item price, cost per unit, and exchange rate

Other measures classifications

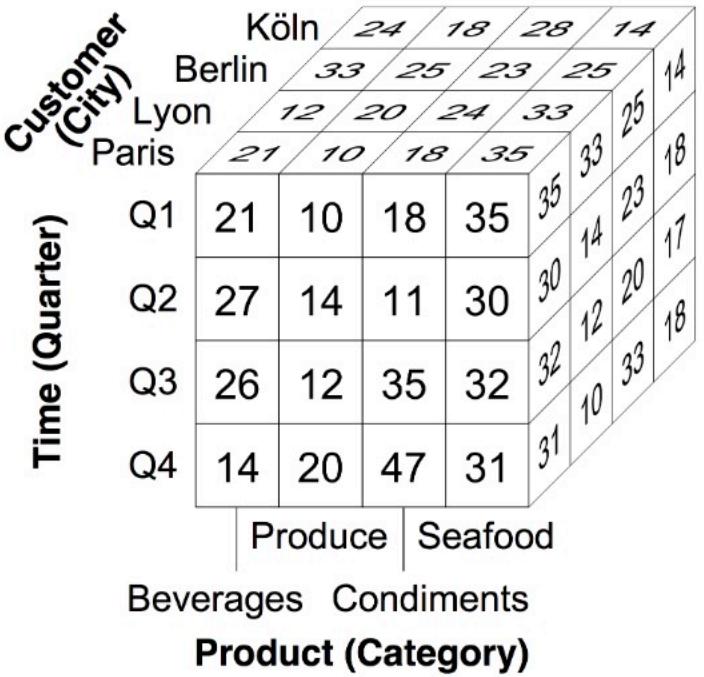
- Measures can also be:
- **Distributive** measures are defined by an aggregation function that can be computed in a distributed way
  - Functions **count**, **sum**, **minimum**, and **maximum** are distributive, **distinct count is not**
  - Example: S = {3,3,4,5,8,4,7,3,8} partitioned in subsets {3,3,4}, {5,8,4}, {7,3,8} gives a result of 8, while the answer over the original set is 5
- **Algebraic** measures are defined by an aggregation function that can be expressed as a scalar function of distributive ones
  - Example: **average**, computed by dividing the sum by the count
- **Holistic** measures cannot be computed from other subaggregates (e.g., median, rank)

OLAP Querying

OLAP operations

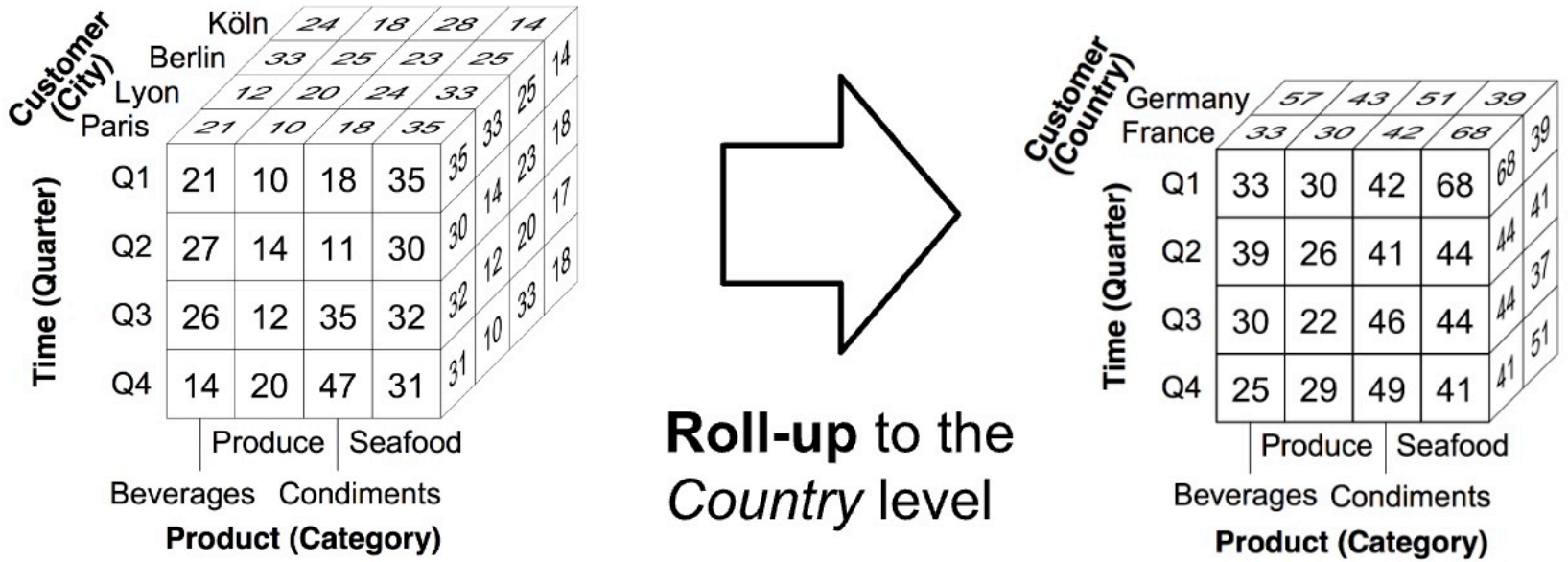
- Taking the same example, we will now see a set of OLAP **operations** we can apply to **explore the cube** of sales (in thousands) by product category and customer cities for 2003:

  - **Roll-up**
  - **Drill-down**
  - **Sort**
  - **Pivot**
  - **Slice**
  - **Dice**



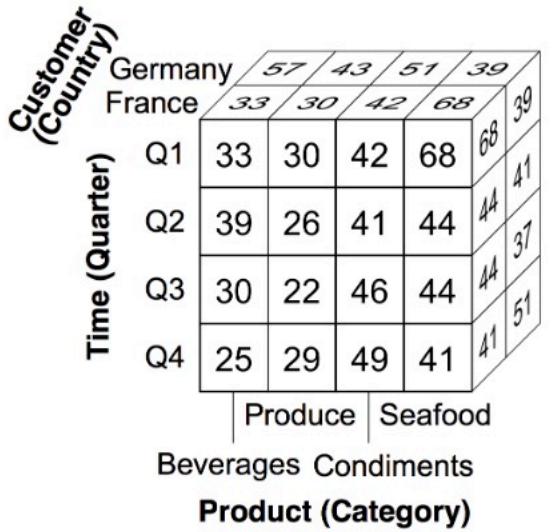
OLAP operation: Roll-up 1/2

- We compute the sales quantities by country: a roll-up operation to the *Country* level along the *Customer* dimension



OLAP operation: Roll-up 2/2

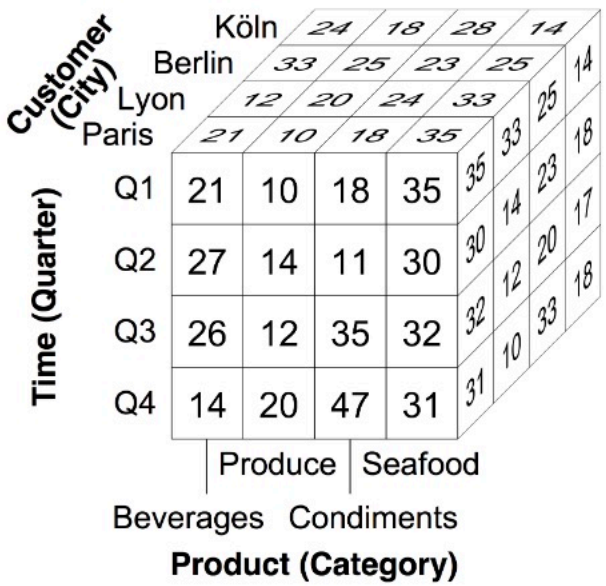
- After rolling-up we can look at measures at *Country* level.
- We notice that France seafood sales for the first quarter are significantly higher



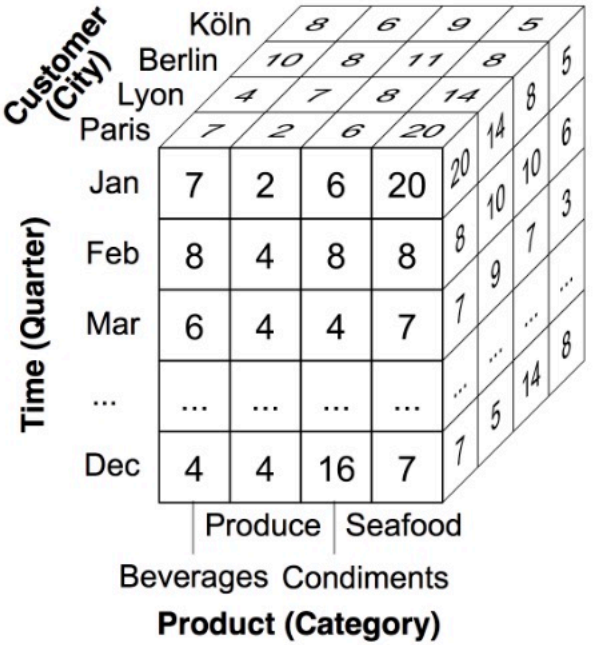
- To find out if this occurred during a particular month of quarter Q1, we take cube back to *City* aggregation level, and **drill-down** along *Time* to the *Month* level…

OLAP operation: Drill-down

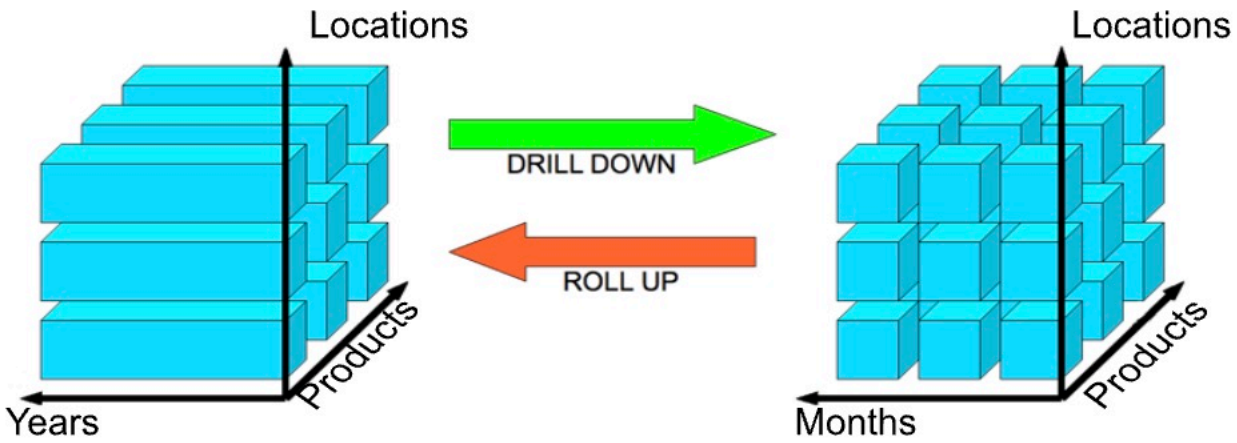
- Drilling-down we go from a coarser granularity (*Quarter*) to a *Month* granularity



Drill-down to the *Month* level



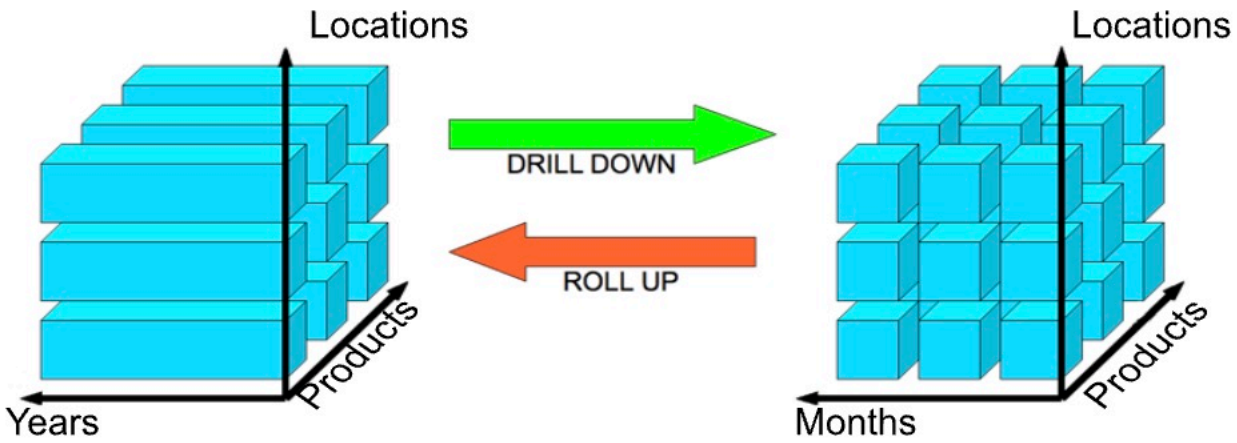
Roll-up/Drill-down in SQL: Example



- Drill-down:

```
SELECT SUM(F.vendite)
FROM Time T JOIN Fatto F JOIN Product P JOIN Location L
GROUP BY T.month, L.region, P.brand
WHERE T.year = 2003
```

Roll-up/Drill-down in SQL: Example



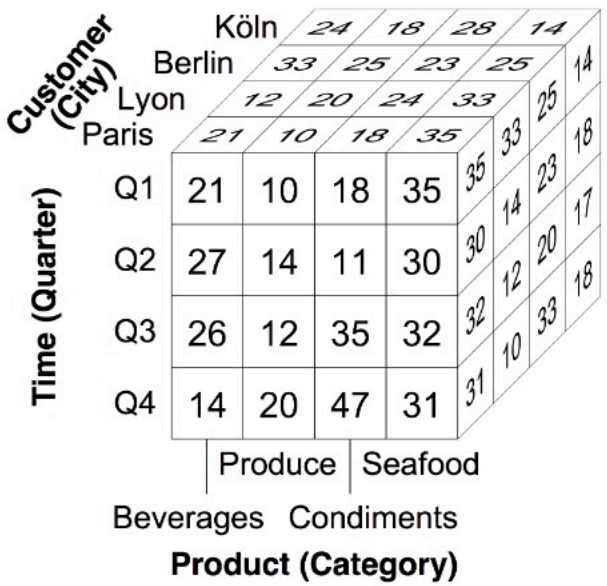
- Roll-up:

```
SELECT SUM(F.vendite)
FROM Time T JOIN Fatto F JOIN Product P JOIN Location L
GROUP BY T.year, L.region, P.brand
WHERE T.year = 2018
```

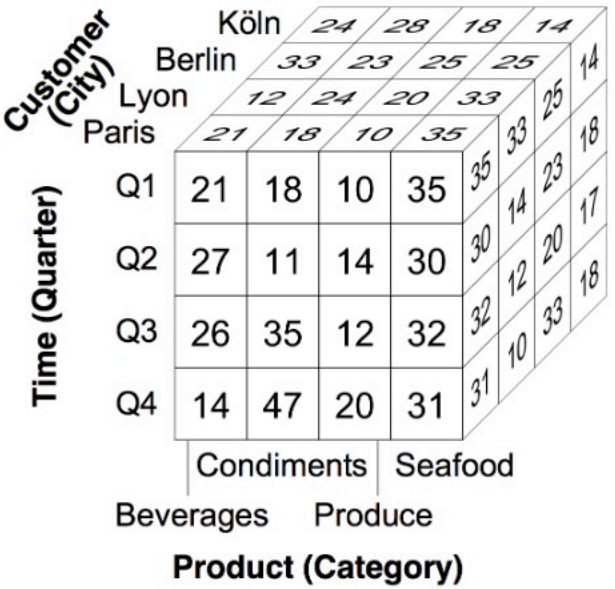
  - Products are now in random order, we now want to visualize the original cube with sorted product...

OLAP operation: Sort

- Sorting the *Product* dimension, now the products, at *Category* level, are sorted alphabetically.

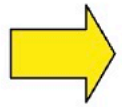


Sort of the *Product* dimension

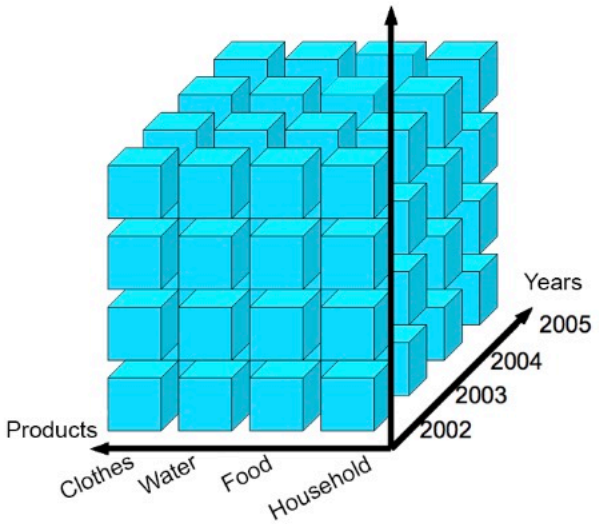


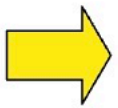
Sort in SQL - Example

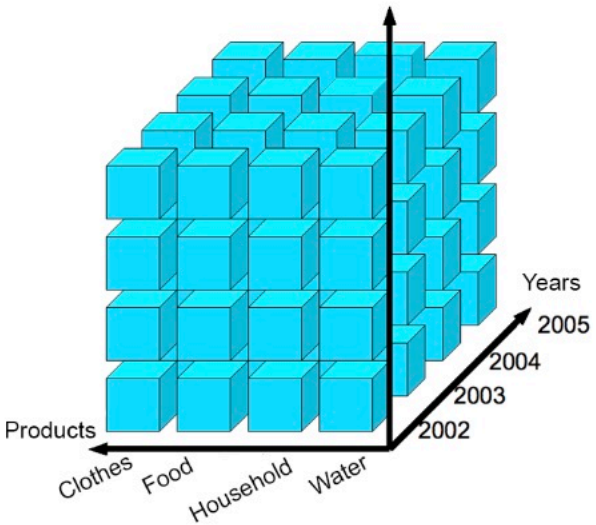
```
SELECT SUM(F.sales)
FROM Time T JOIN Fact F
JOIN Products P
JOIN Locations L
```



```
SELECT SUM(F.sales)
FROM Time T JOIN Fact F
JOIN Products P
JOIN Locations L
ORDER BY P.category
```



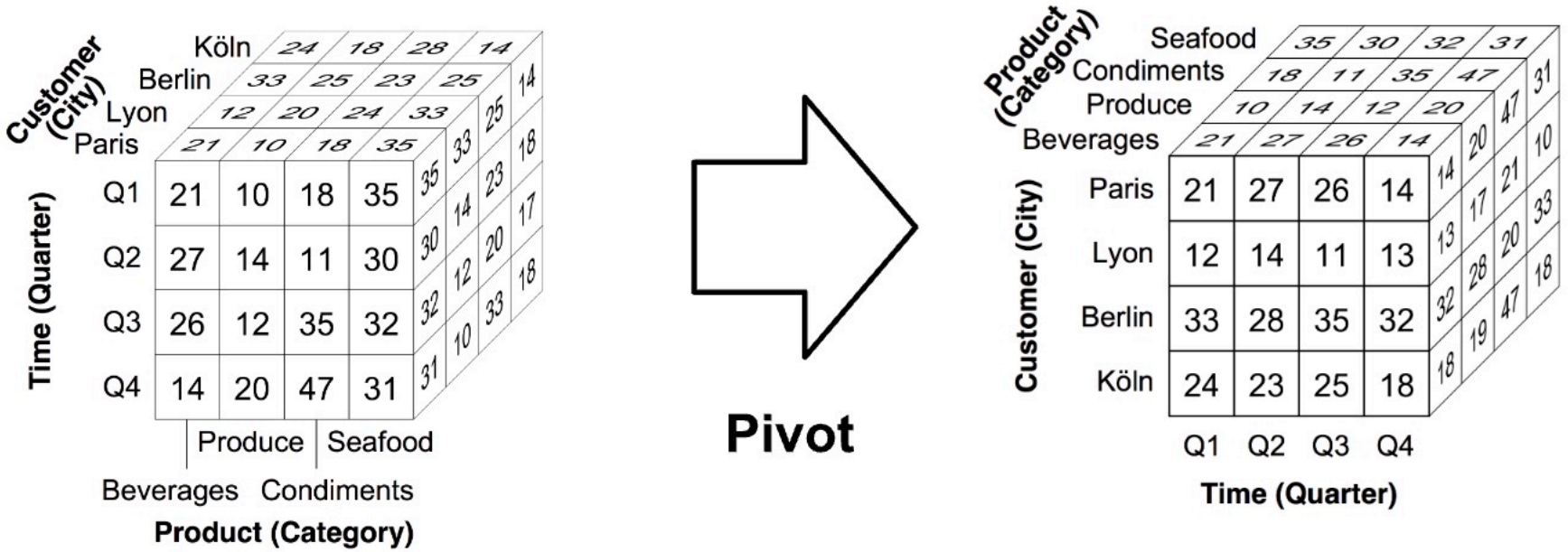




- To introduce the next operation, suppose that we want to rotate the cube to see the Time dimension on the X axis...

OLAP operation: Pivot

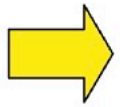
- By using the **pivot** operation, the *Time* dimension is now on the **X axis**, the *Customer* dimension is now in the **Y axis**, while the *Product* dimension is on the **Z axis**.



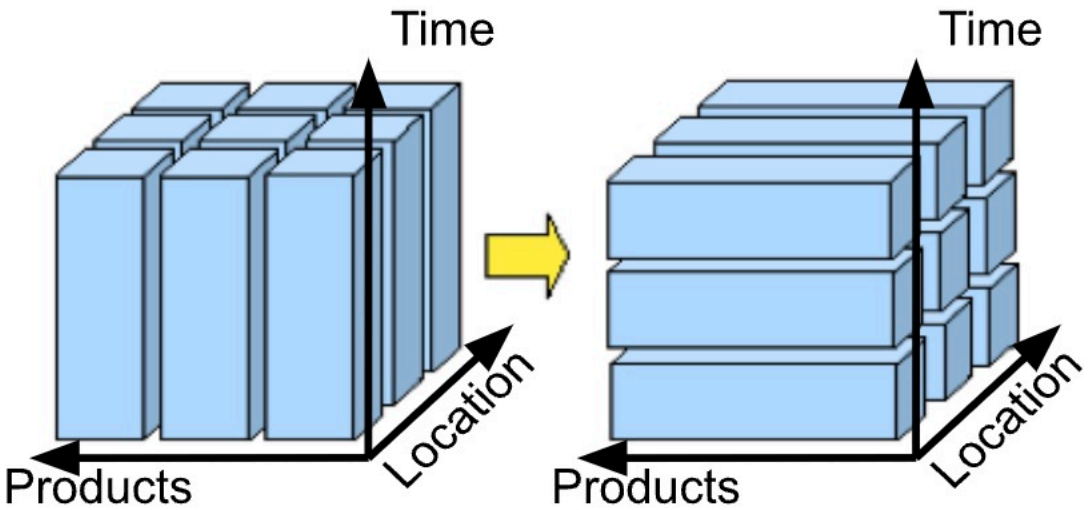
- Let's see an example in SQL.

Pivot in SQL - Example

```
SELECT SUM(F.sales)
FROM Time T JOIN Fact F
JOIN Products P
JOIN Locations L
GROUP BY P.category
```



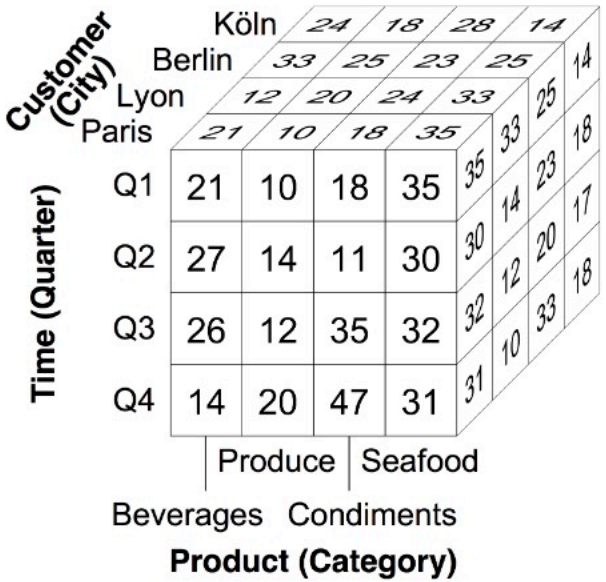
```
SELECT SUM(F.sales)
FROM Time T JOIN Fact F
JOIN Products P
JOIN Locations L
GROUP BY T.year
```



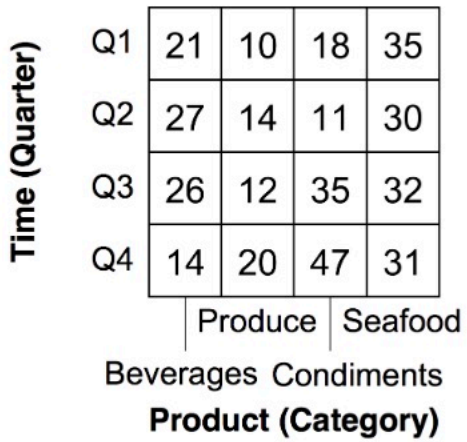
- What about filtering our data? Let's suppose we want to look at a single *City*...

OLAP operation: Slice

- The **slice** operation visualizes the data for Paris only, resulting in a 2-dimensional "subcube"



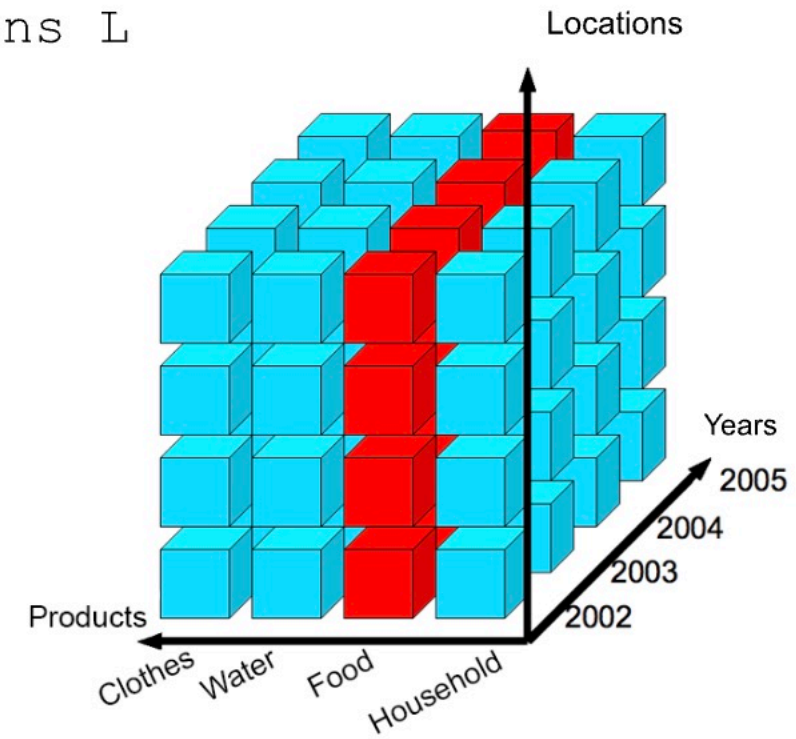
Slice on *City*="Paris"



- Let's see an example in SQL.

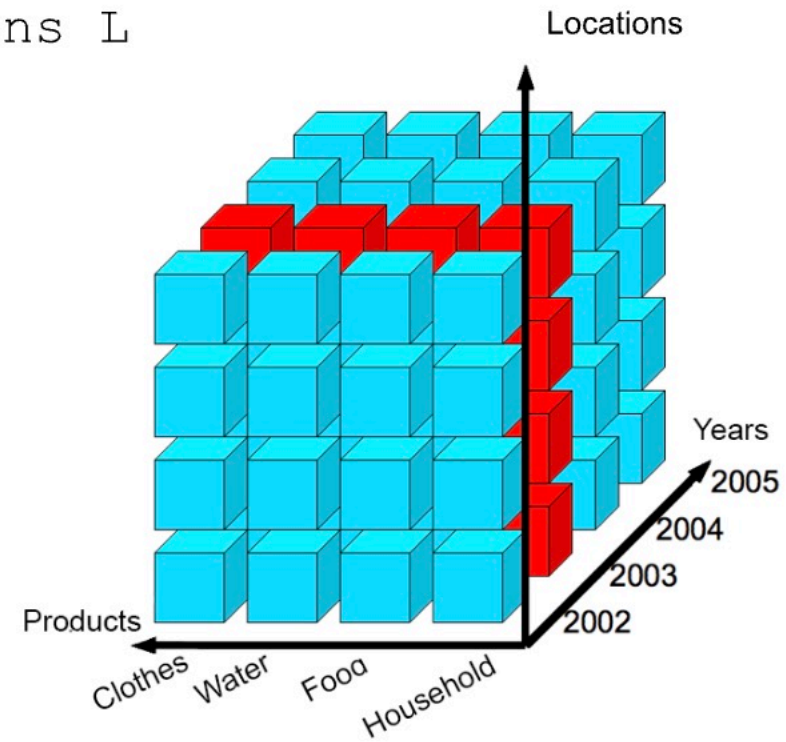
Slice in SQL - Example 1

```
SELECT F.sales
FROM Time T JOIN Fact F
JOIN Products P JOIN Locations L
WHERE P.category="Food"
```



Slice in SQL - Example 2

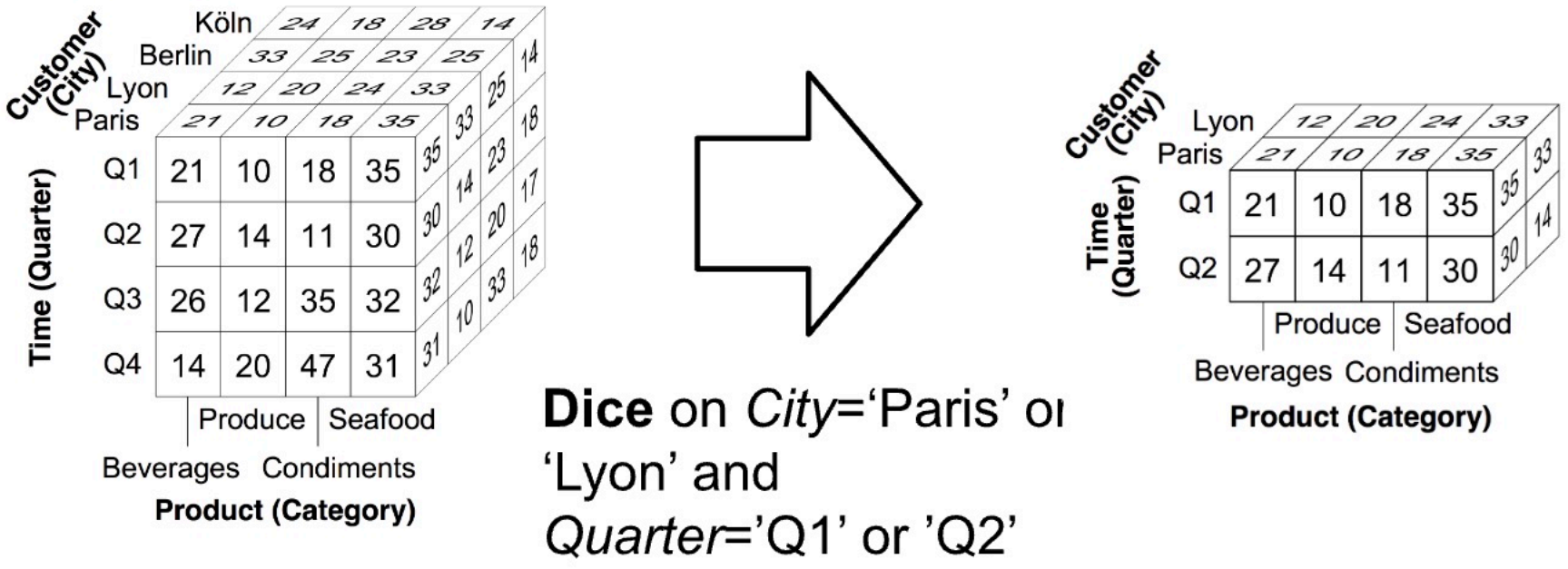
```
SELECT F.sales
FROM Time T JOIN Fact F
JOIN Products P JOIN Locations L
WHERE T.years=2003
```



- Slice allows to show only a single value for a single dimension. What if we want to select multiple values from different dimensions?

OLAP operation: Dice

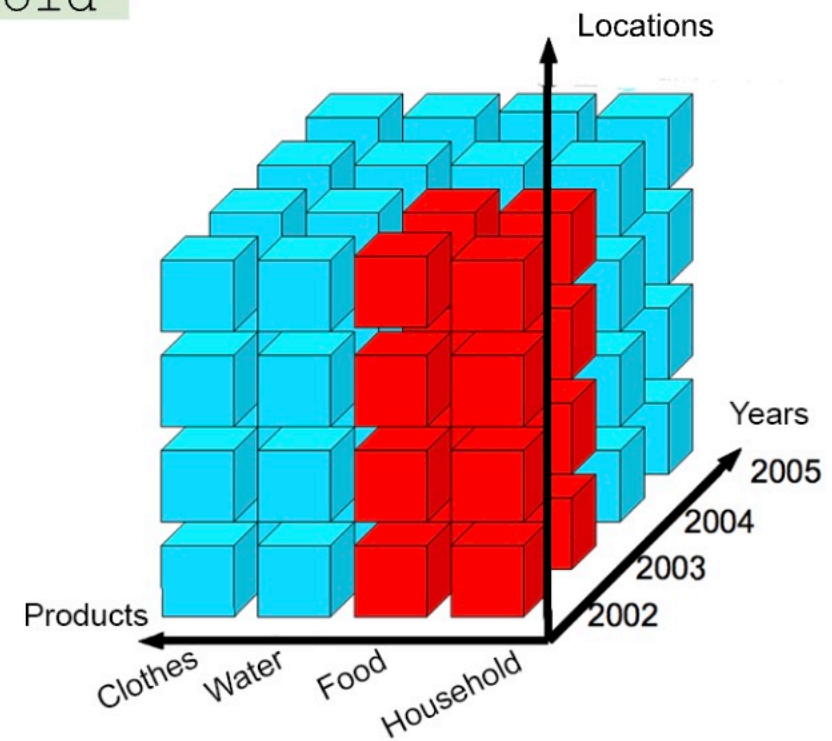
- With **dice** we can select over multiple conditions and multiple dimensions, producing a 3-dimensional subcube.



- We will now see all the operations with their **syntax** and more details.

Dice in SQL - Example

```
SELECT F.sales
FROM Time T JOIN Fact F
JOIN Products P JOIN Locations L
WHERE T.years<2003
AND P.category LIKE 'Food'
OR P.category LIKE 'Household'
```



References

Data Warehouse Systems
Design and Implementation

Authors: Vaisman, Alejandro,
Zimányi, Esteban

Extensive coverage of all data warehouse issues, ranging from basic technologies to the most recent findings and systems