

Data Management for Data Science

*Corso di laurea magistrale in Data Science
Sapienza Università di Roma
2015/2016*

Data Warehousing

Domenico Lembo, Riccardo Rosati

*Dipartimento di Ingegneria Informatica Automatica e Gestionale A. Ruberti
Sapienza Università di Roma*

Credits

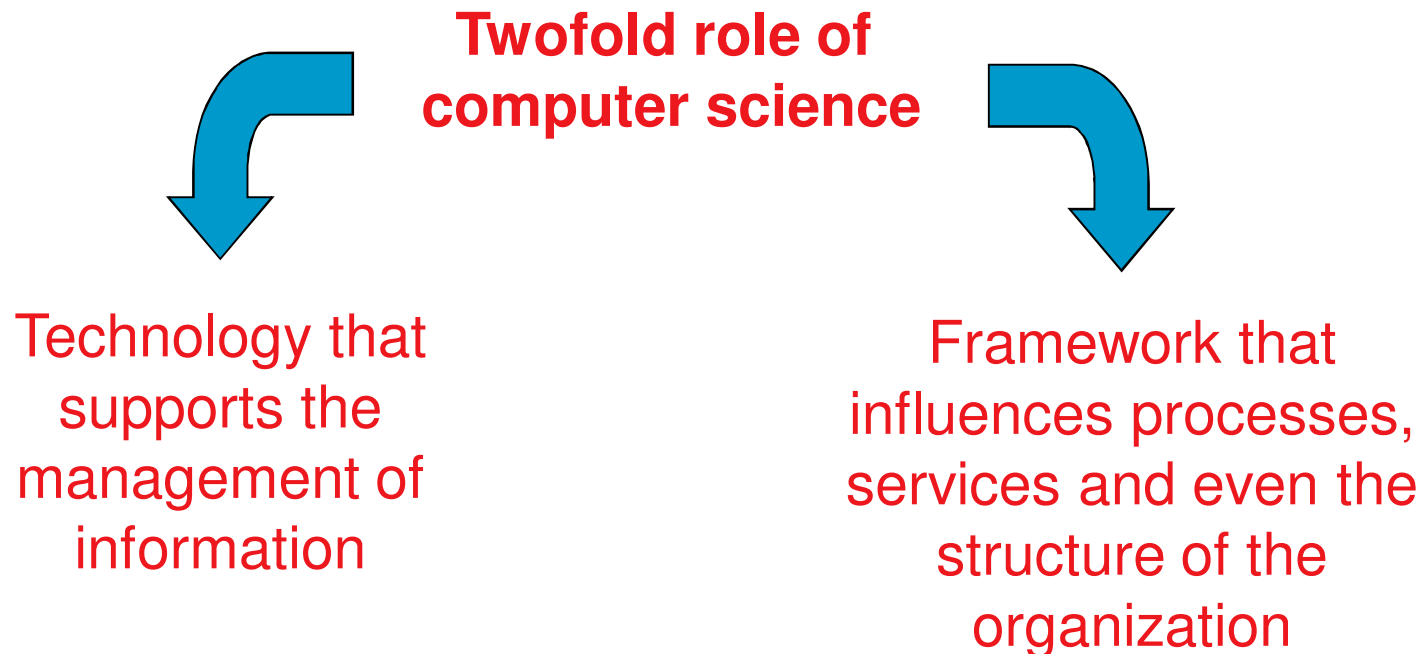
- These slides are adapted from the original slides of prof. Stefano Rizzi, University of Bologna
- Bibliographic reference:
M. Golfarelli, S. Rizzi. Data Warehouse Design: Modern Principles and Methodologies. McGraw-Hill, 2009.

Enterprise Databases

- Until the mid-1980s, enterprise databases stored only *operational data*, i.e., data created by business operations involved in daily management processes
- Computer science was considered as a *support* means aimed at saving time and resources in the management of information
- Thus it was not conceived as valuable *per se*, i.e., for its ability of positively influencing business processes and generating added value for them

Evolution of Information Systems

- The role of Information systems has radically changed from the early seventies, when they started to be commonly used
- Indeed, from **tools only useful to improve the efficiency of business processes** they have become **central elements in the organizations**, able to completely change the structure of such processes



Directional Portfolio

- Is a set of applications and services used by managers to:
 - ✓ Analyze the status of the enterprise
 - ✓ Take strategic decisions
 - ✓ Take fast and best strategic decisions
- Information technologies that support the achievement of the above aims are commonly known as *Business Intelligence* technologies, i.e.:

Discipline that allows the decision makers in the company to understand, through software solutions, the key factors of the business, and consequently to make the best decisions at that time

Business intelligence

- In order to allow managers to realize powerful and flexible analysis, it is necessary to define an appropriate hardware and software infrastructure, consisting of:
 - ✓ Dedicated Hardware
 - ✓ Network Infrastructures
 - ✓ DBMS
 - ✓ Back-end software
 - ✓ Front-end software
- The key role of a business intelligence platform is to transform business data into information exploitable at different levels of detail

From data to Information

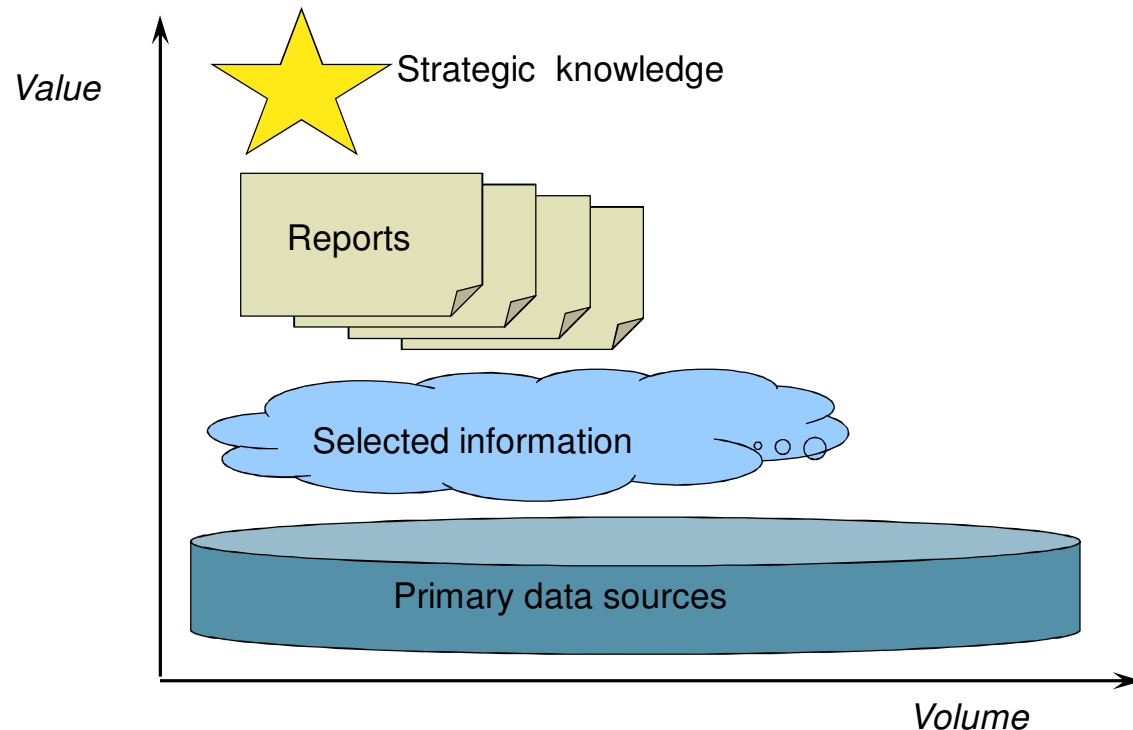
- Information is an important enterprise asset, needed to effectively plan and control the business activities
- It is the raw material that is transformed by information systems, such as semi-finished products are transformed by production systems

data ≠ information

- Often the availability of too much data makes it difficult, if not impossible, to extrapolate the really important information

From data to Information

- Every enterprise must have **quick, comprehensive** access to the information required by decision-making processes. This strategic information is extracted mainly from the huge amount of operational data stored in enterprise databases by means of a progressive selection and aggregation process



Decision Support Systems

- Decision Support Systems (DSSs) started to become popular in the eighties:

A *DSS* is a set of expandable, interactive IT techniques and tools designed for processing and analyzing data and for supporting managers in decision making.

Role of DSSs

In the past	In the future
Describe the past	Anticipate future
Reduce costs	Increase Profits
Describe problems	Suggest changes

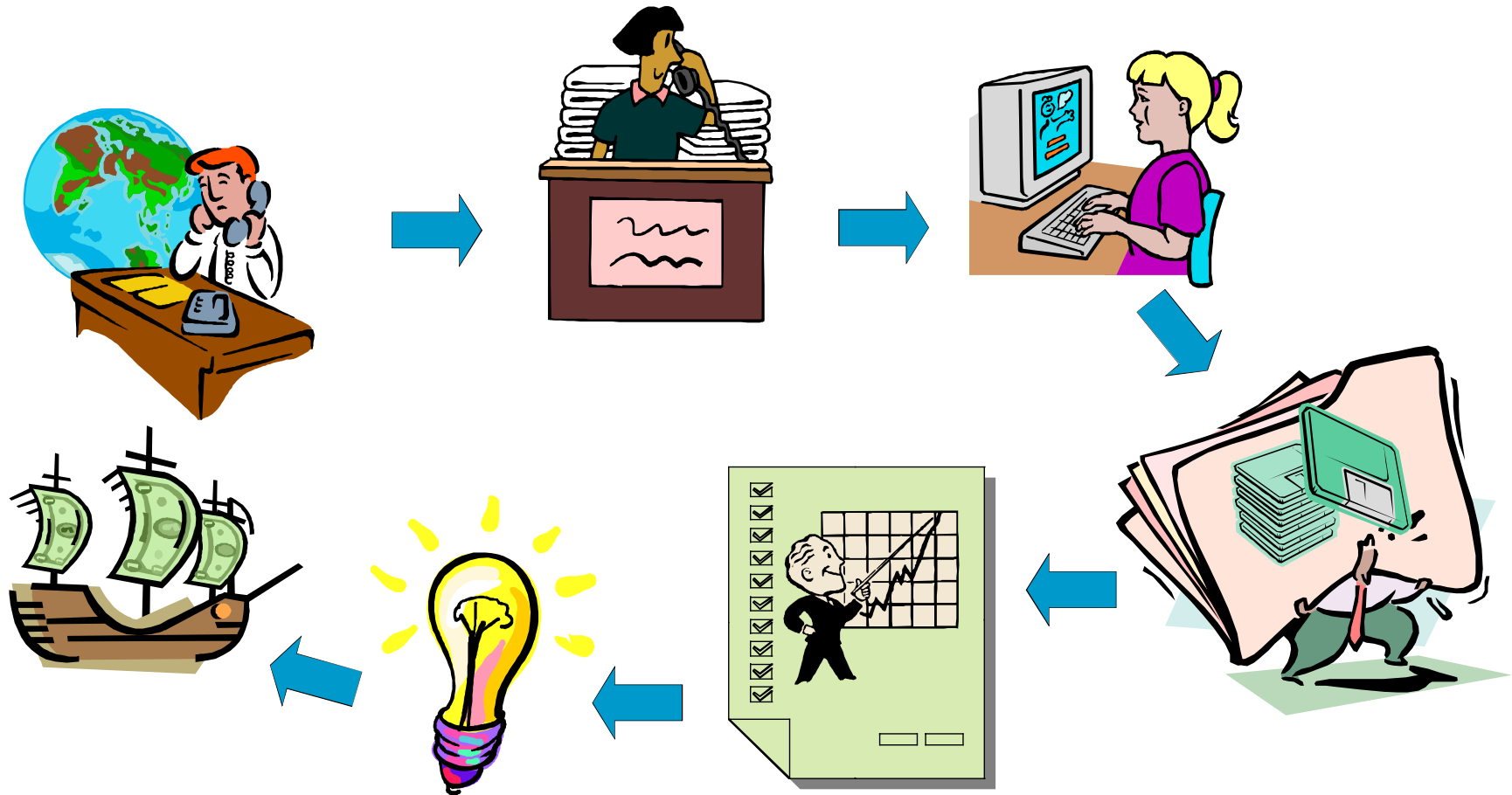
- *Data warehouse systems have been managing the data back-ends of DSSs since the 1990s*

Classification of DSSs

System	Description
Passive DSS	Supports decision-making processes, but it does not offer explicit suggestions on decisions or solutions.
Active DSS	Offers suggestions and solutions.
Collaborative DSS	Operates interactively and allows decision-makers to modify, integrate, or refine suggestions given by the system. Suggestions are sent back to the system for validation.
Model-driven DSS	Enhances management of statistical, financial, optimization, and simulation models.
Communication-driven DSS	Supports a group of people working on a common task.
Data-driven DSS	Enhances the access and management of time series of corporate and external data.
Document-driven DSS	Manages and processes nonstructured data in many formats.
Knowledge-driven DSS	Provides problem-solving features in the form of facts, rules, and procedures.

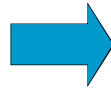
A typical scenario....

- .. is that of a large company, with numerous branches, whose managers want to evaluate the contribution of each branch to the overall business performance of the company



A typical scenario....

- An information repository that integrates and reorganizes data collected from sources of various kind and makes them available for analysis and evaluations aimed at planning and decision-making

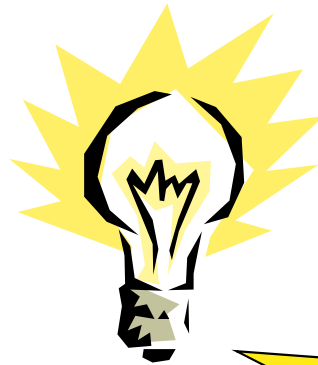


**DATA
WAREHOUSE**

Data Warehousing: a collection of methods, techniques, and tools used to support *knowledge workers* (senior managers, directors, managers, and analysts) to conduct data analyses

OLTP e OLAP

- Mixing together "analytical" and "transactional" queries leads to inevitable delays that make dissatisfied users of both categories.



One of the main aims of Data Warehousing is to **maintain separate** On-Line Analytical Processing (OLAP) from On-Line Transactional Processing (OLTP)

Some Areas where DW technologies are normally adopted

- **Commerce** (analysis of sales and complaints, shipping and inventory control, customer care)
- **Manufactury** (control of production costs, support of suppliers and orders)
- **Financial services** (risk analysis, fraud detection)
- **Trasportation** (fleet management)
- **Telecommunication** (analysis of call data, customer profile)
- **Healthcare** (analysis of admissions and discharges, accounting for cost centers)
-

Companies complaints

- *We have heaps of data, but **we cannot access it!***
- *How can people playing the same role achieve substantially different results?*
- *We want to **select, group, and manipulate data** in every possible way!*
- *Show me just what matters!*
- *Everyone knows that **some data is wrong!***

R. Kimball, The Data Warehouse Toolkit



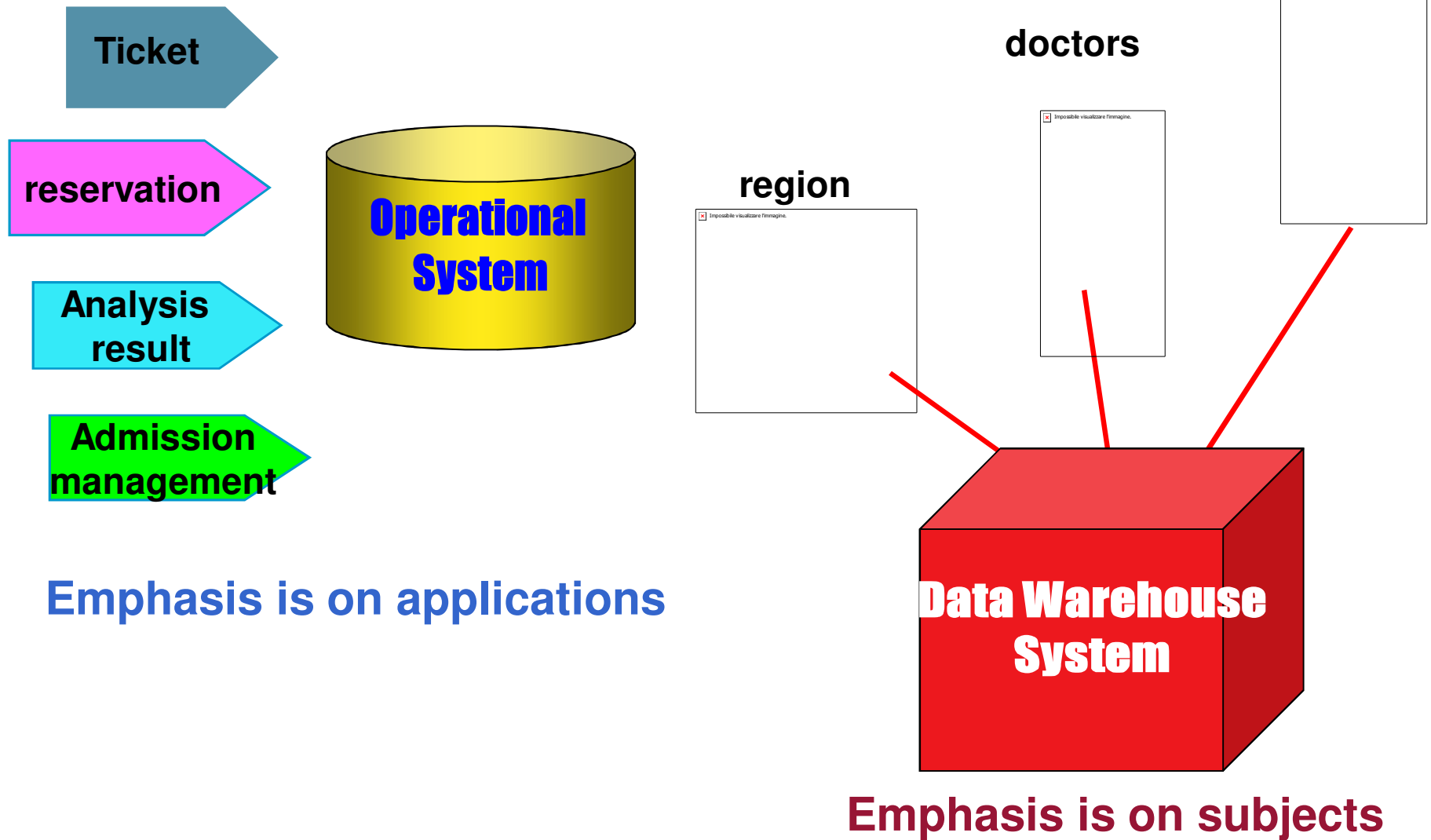
Data Warehousing Characteristics

- *accessibility* to users not very familiar with IT and data structures;
- *integration* of data on the basis of a standard enterprise model;
- *query flexibility* to maximize the advantages obtained from the existing information;
- *multidimensional representation* giving users an intuitive and manageable view of information;
- *correctness and completeness* of integrated data.

Data Warehouse

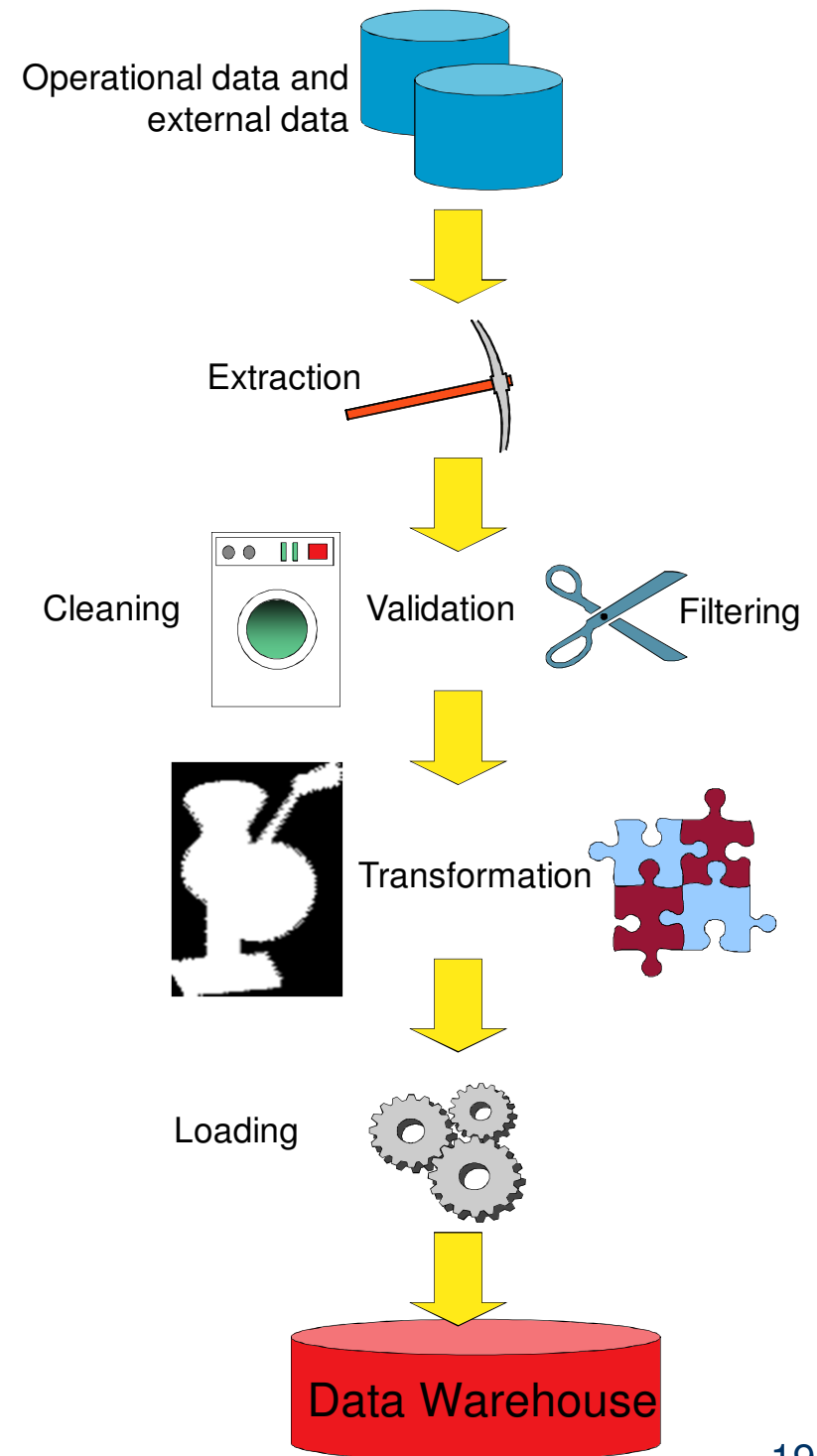
- Data warehouses are placed right in the middle of this process and act as repositories for data. They make sure that the requirements set can be fulfilled.
- *A data warehouse* is a collection of data that supports decision-making processes. It provides the following features (Inmon, 2005):
 - ✓ It is subject-oriented.
 - ✓ It is integrated and consistent
 - ✓ It shows its evolution over time
 - ✓ it is not volatile.

...subject oriented



...integrated and consistent

- Data warehouses take advantage of multiple data sources, such as data extracted from production and then stored to enterprise databases, or even data from a third party's information systems.
- A data warehouse **should provide a reconciled unified view of all the data**. Generally speaking, we can state that creating a data warehouse system does not require that new information is added; rather, existing information needs rearranging.



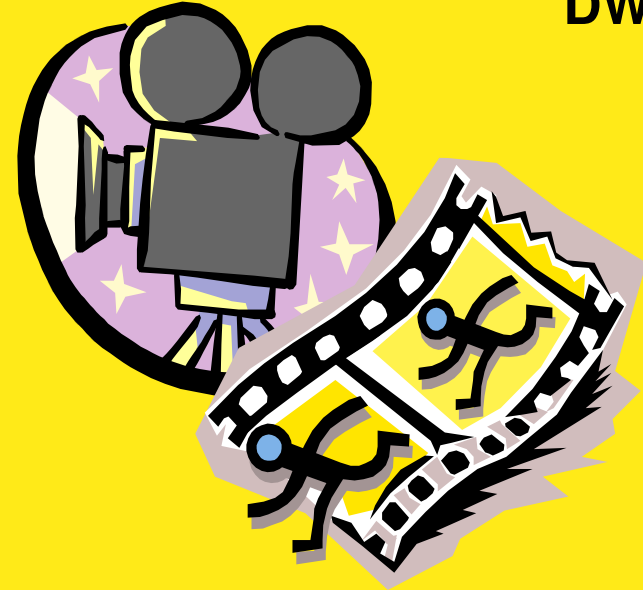
...shows its evolution over the time

Operational DBs



Operational data usually covers a short period of time, because most transactions involve the latest data. **No historical data:** data are updated and old value cancelled. **Time is not in the keys.**

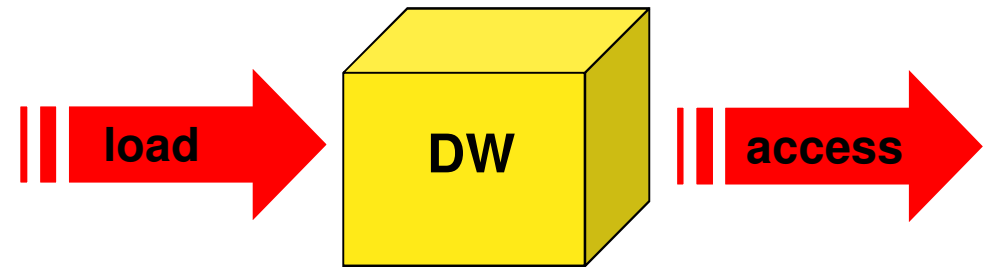
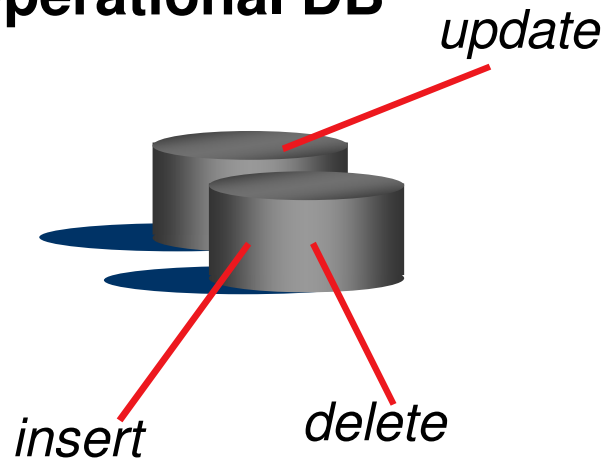
DW



Enable analyses that cover a few years. Regularly updated and continuously growing. **Time is part of the keys.**

...non-volatile

Operational DB



**Huge data volume:
from tens of GB to some TB
or even more**

- ✓ **data is never deleted** from data warehouses and updates are normally carried out when data warehouses are offline.
- ✓ This means that data warehouses can be essentially viewed as **read-only databases**.
- ✓ **in a DW** there is no need for advanced transaction management techniques required by operational applications.
- ✓ **Key problems** are query-throughput and resilience.

Queries

■ OLTP:

- ✓ Operational queries execute transactions that generally read/write a small number of tuples from/to many tables connected by simple relations. For example, this applies if you search for the data of a customer in order to insert a new customer order.
- ✓ The core workload is often “frozen” into applications (ad hoc data queries are occasional).

■ OLAP:

- ✓ Queries execute dynamic, multidimensional analyses that need to scan a huge amount of records to process a set of numeric data summing up the performance of an enterprise.
- ✓ Data warehouse interactivity is an essential property for analysis sessions, so the actual workload constantly changes as time goes by.

Summarizing

Feature	Operational Databases	Data Warehouses
Users	Thousands	Hundreds
Workload	Preset transactions	Specific analysis queries
Access	To hundreds of records, write and read mode	To millions of records, mainly read-only mode
Goal	Depends on applications	Decision-making support
Data	Detailed, both numeric and alphanumeric	Summed up, mainly numeric
Data integration	Application-based	Subject-based
Quality	In terms of integrity	In terms of consistency
Time coverage	Current data only	Current and historical data
Updates	Continuous	Periodical
Model	Normalized	Denormalized, multidimensional
Optimization	For OLTP access to a database part	For OLAP access to most of the database

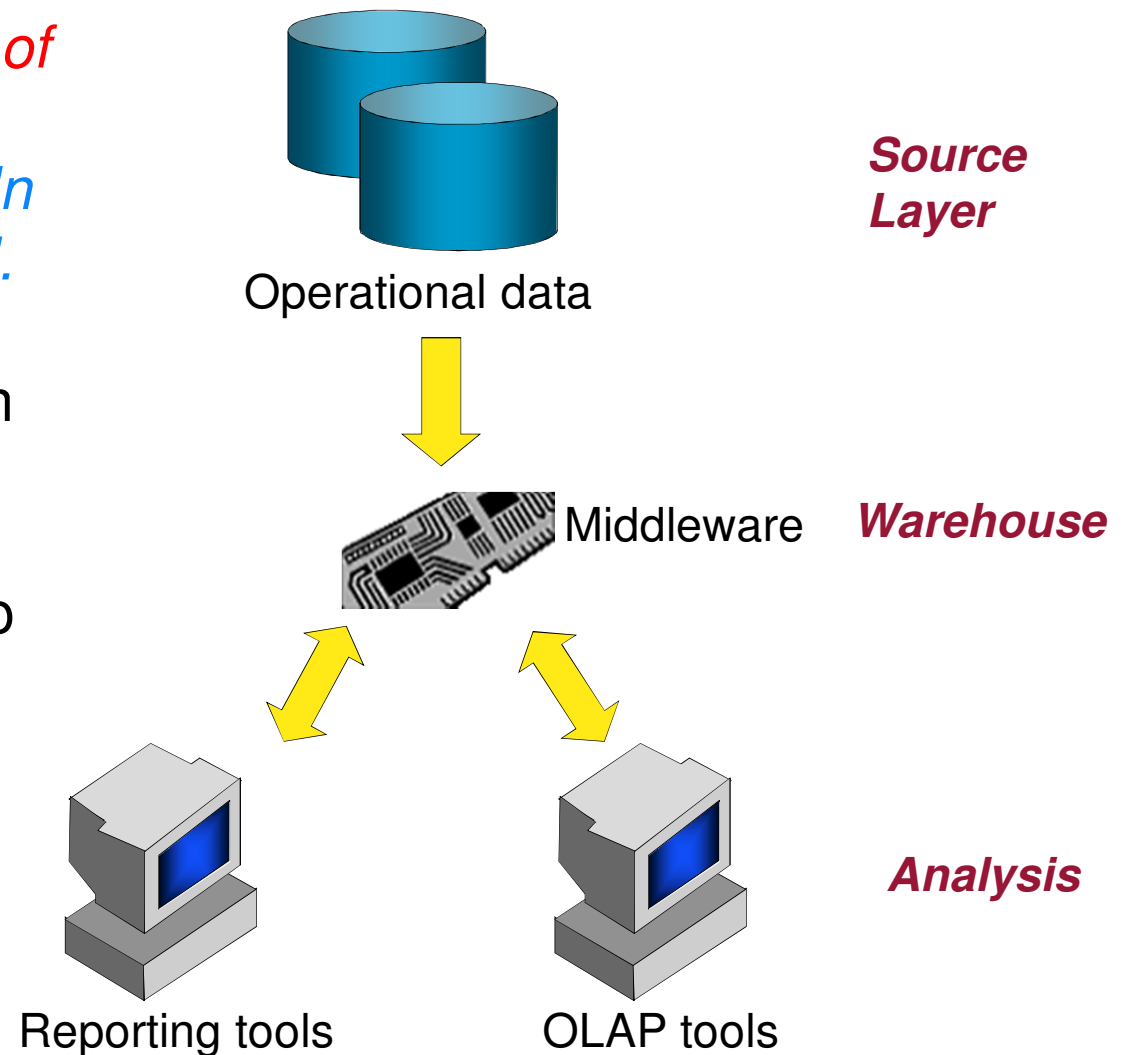
Architectures: requirements

- **Separation:** Analytical and transactional processing should be kept apart as much as possible.
- **Scalability:** Hardware and software architectures should be easy to upgrade as the data volume (which has to be managed and processed) and the number of users' requirements (which have to be met) progressively increase.
- **Extensibility:** The architecture should be able to host new applications and technologies without redesigning the whole system.
- **Security:** Monitoring accesses is essential because of the strategic data stored in data warehouses.
- **Administerability:** Data warehouse management should not be overly difficult.

Single-Layer Architecture

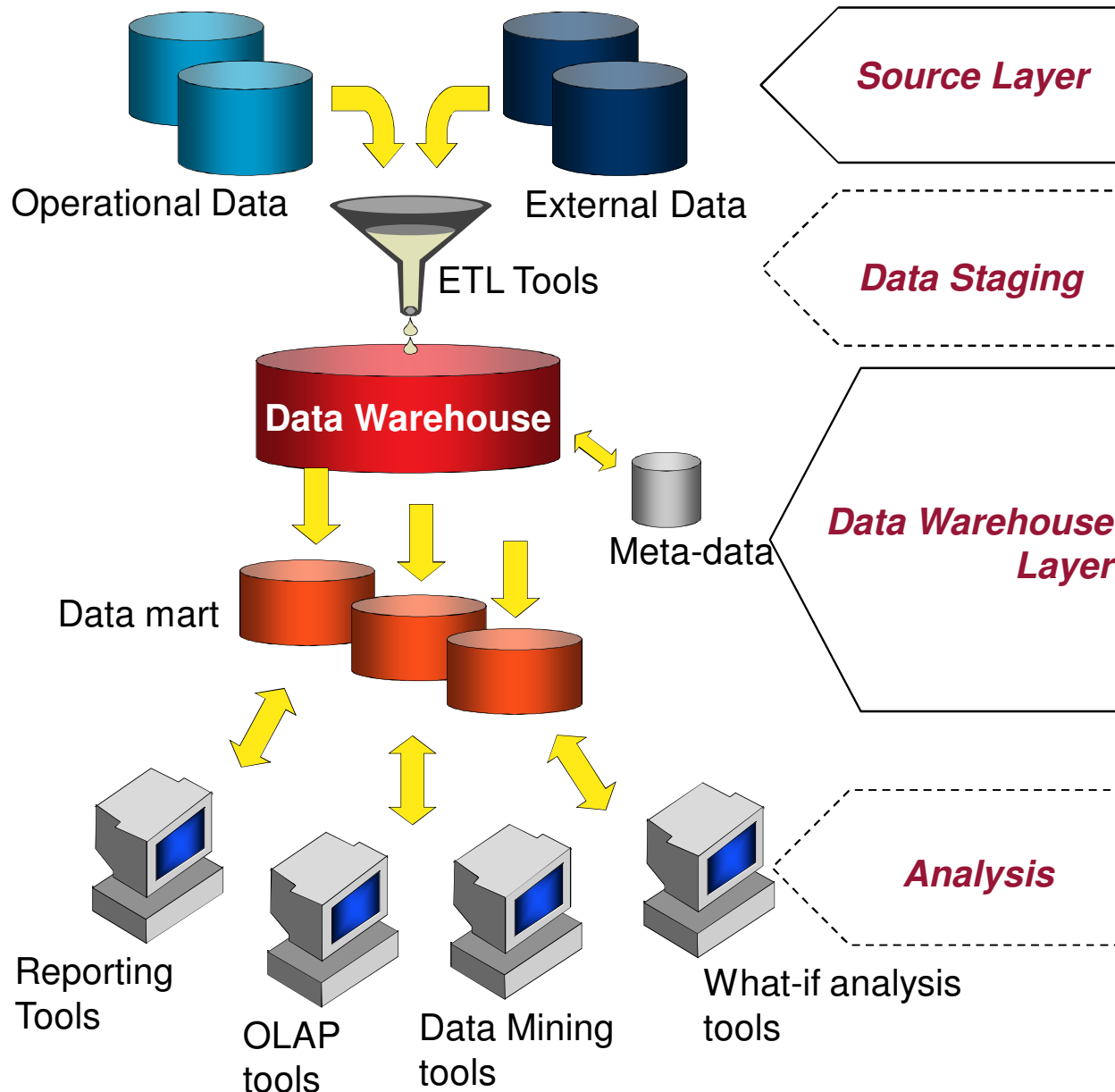
Its goal is to minimize the amount of data stored. The only physically available layer is the source layer. In this case, data warehouse is virtual.

- ✓ Fails to meet separation between OLTP and OLAP
- ✓ Analysis queries are submitted to operational data after the middleware interprets them, thus affecting regular transactional workloads
- ✓ It cannot log more data than sources do



Traditionally, single-layer architecture has not been frequently used in practice: a virtual approach to data warehouses can be successful only if analysis needs are particularly restricted and the data volume to analyze

Two-Layer Architecture*



DATA MART:
Subset or aggregation of the data stored to a primary data warehouse. It includes a set of information pieces relevant to a specific business area, corporate department, or category of users.

* The name highlights a separation between physically available sources and data warehouses, but in fact it consists of four subsequent data flow stages

Two-Layer Architecture

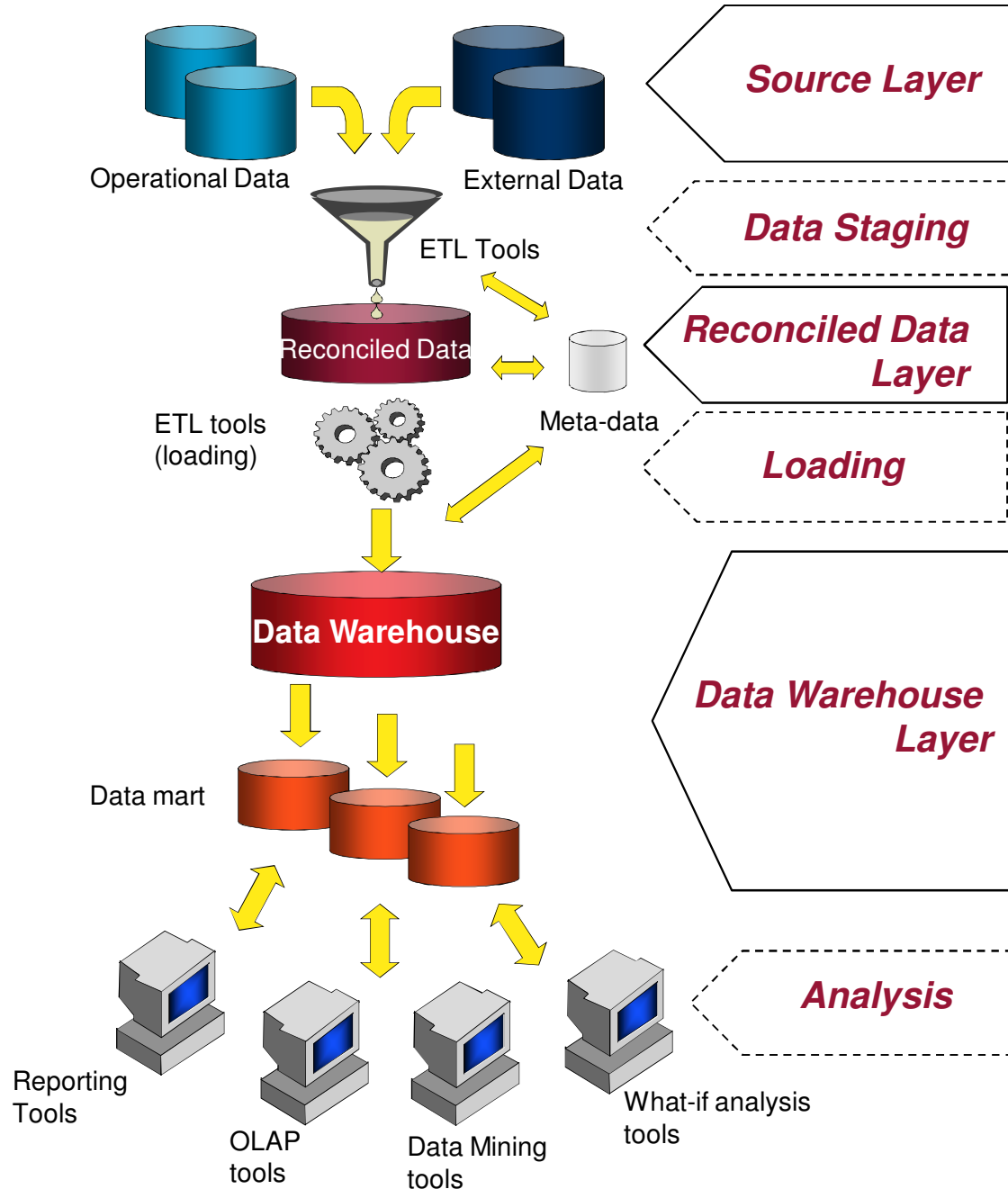
- The data marts populated from a primary data warehouse are often called *dependent*. They are very useful for data warehouse systems in midsize to large enterprises because
 - ✓ they are used as building blocks while incrementally developing data warehouses;
 - ✓ they mark out the information required by a specific group of users to solve queries;
 - ✓ they can deliver better performance because they are smaller than primary data warehouses.
- Sometimes, mainly for organization and policy purposes, a different solution is adopted in which sources are used to directly populate data marts. These data marts are called *independent*
 - ✓ If there is no primary data warehouse, this streamlines the design process, but it leads to the risk of inconsistencies between data marts.

Two-Layer Architecture

■ Advantages:

- ✓ In data warehouse systems, good quality information is always available, even when access to sources is denied temporarily for technical or organizational reasons.
- ✓ Data warehouse analysis queries do not affect the management of transactions, the reliability of which is vital for enterprises to work properly at an operational level.
- ✓ Data warehouses are logically structured according to the multidimensional model, while operational sources are generally based on relational or semi-structured models.
- ✓ A mismatch in terms of time and granularity occurs between OLTP systems, which manage current data at a maximum level of detail, and OLAP systems, which manage historical and summarized data.
- ✓ Data warehouses can use specific design solutions aimed at performance optimization of analysis and report applications.

Three-Layer Architecture



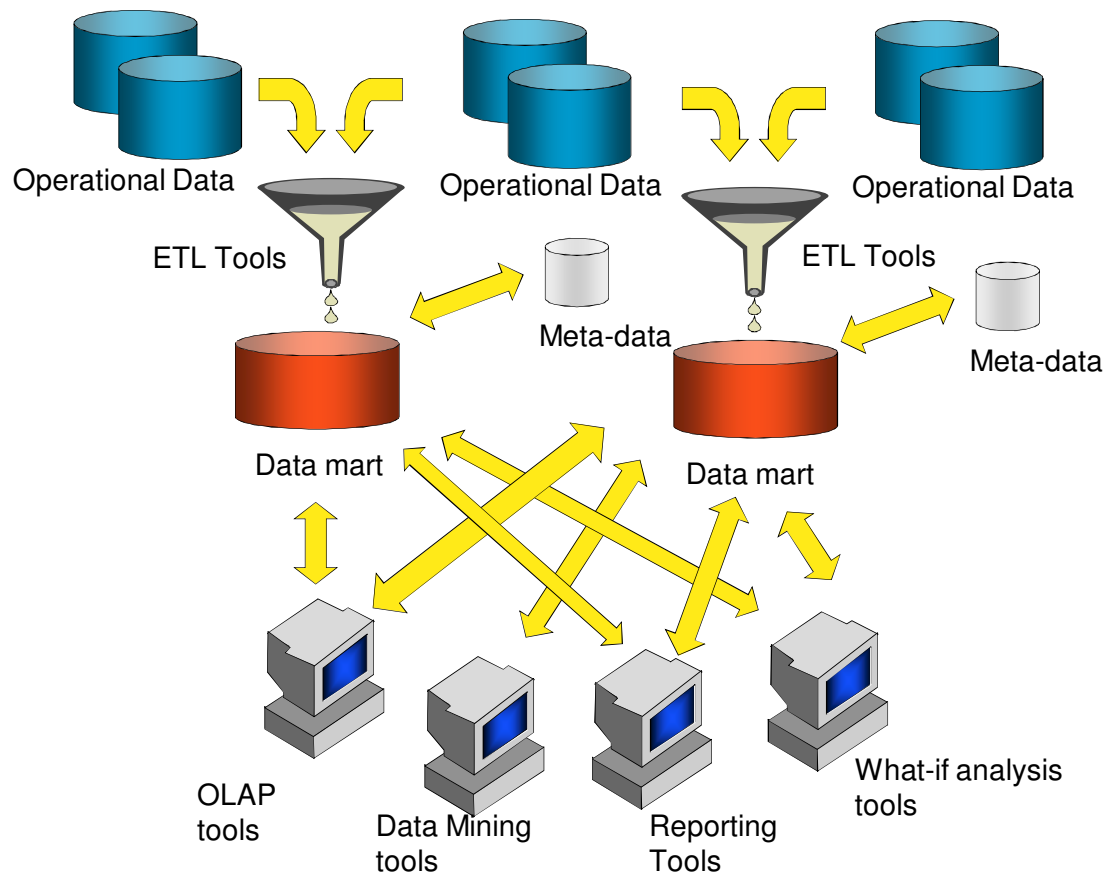
RECONCILED DATA:
This layer materializes operational data obtained after integrating and cleansing source data. As a result, those data are integrated, consistent, correct and detailed.

Three-Layer Architecture

- The main advantage of the reconciled data layer is that it creates a common reference data model for a whole enterprise. At the same time, it sharply separates the problems of source data extraction and integration from those of data warehouse population.
- However, reconciled data leads to more redundancy of operational source data.

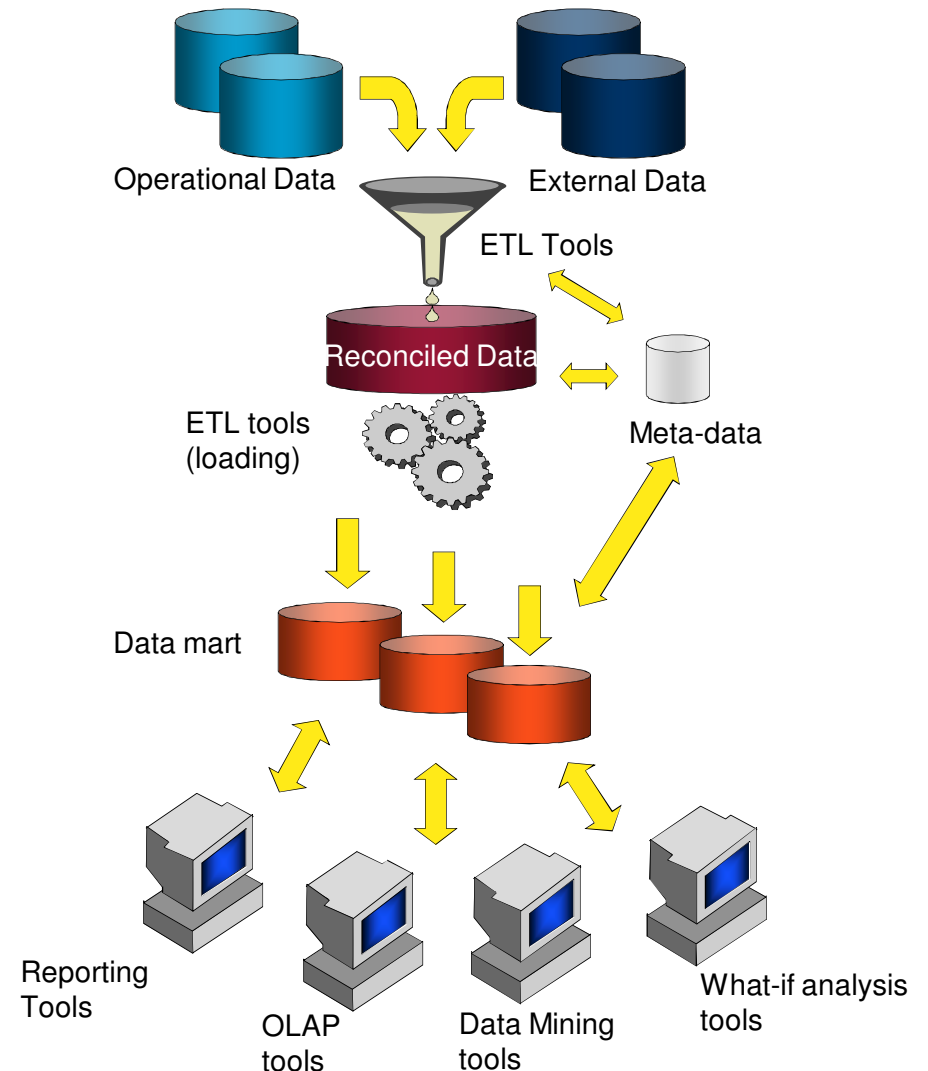
An additional classification (1)

- *independent data marts architecture*, different data marts are separately designed and built in a nonintegrated fashion (it is an extreme case of independent data marts in the two-layer architecture, in the sense that there can be no integration among different datamarts)
- *bus architecture* (recom. by Ralph Kimball), is similar to the preceding architecture, but adopts a basic set of *conformed dimensions* as a common design guideline. This ensures logical integration of data marts and an enterprise-wide view of information.



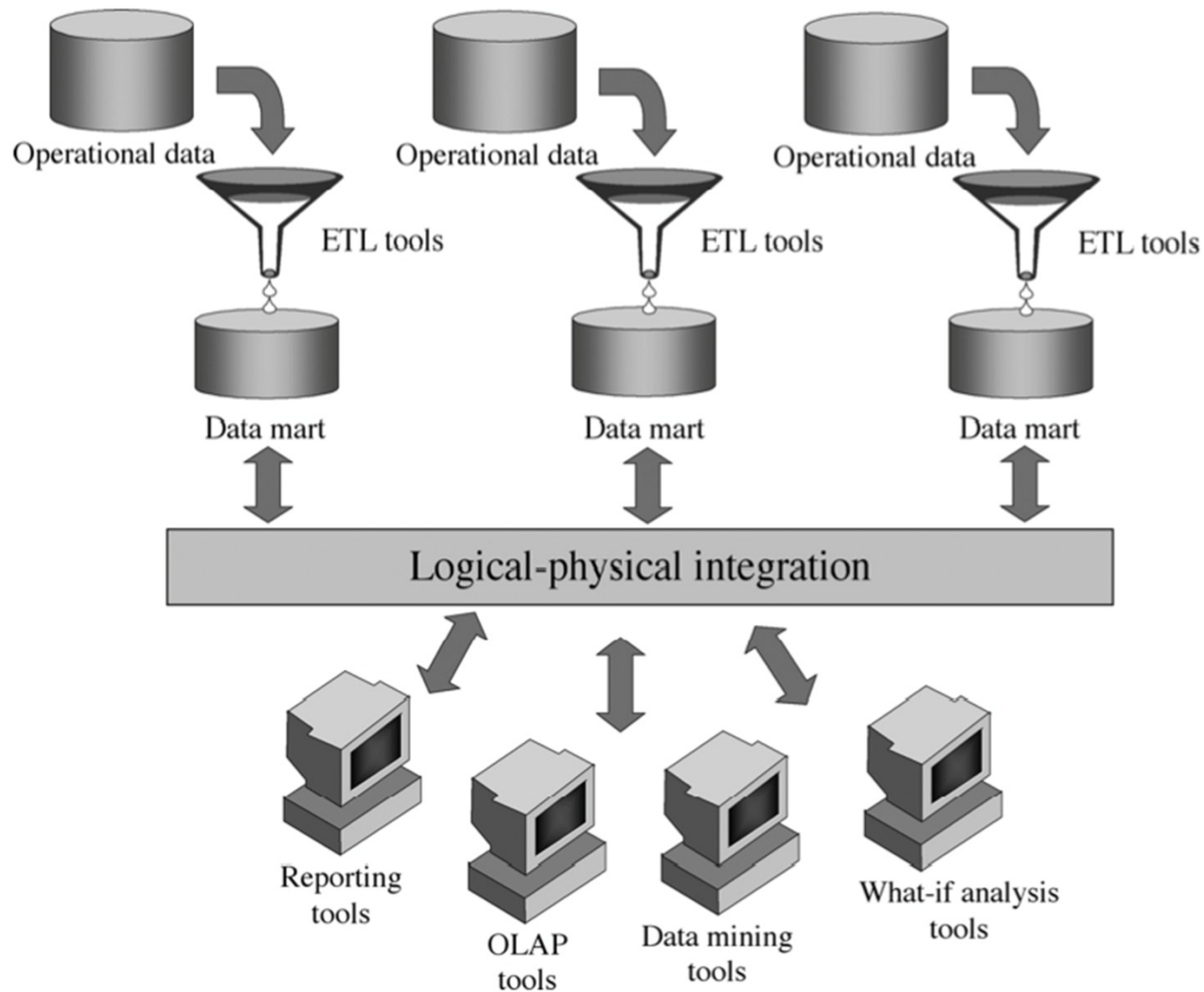
An additional classification (2)

- *hub-and-spoke architecture* poses much attention to scalability and extensibility, and to achieving an enterprise-wide view. Atomic, normalized data is stored in a reconciled layer that feeds a set of data marts containing summarized data in multidimensional form
- The *centralized architecture*, recommended by Bill Inmon, can be seen as a particular implementation of the hub-and-spoke architecture, where the reconciled layer and the data marts are collapsed into a single physical repository



An additional classification (3)

- The *federated architecture* is adopted in dynamic contexts where preexisting data warehouses/data marts are to be noninvasively integrated. Each data warehouse/data mart is either virtually or physically integrated with the others, leaning on a variety of advanced techniques such as distributed querying, ontologies, and meta-data interoperability



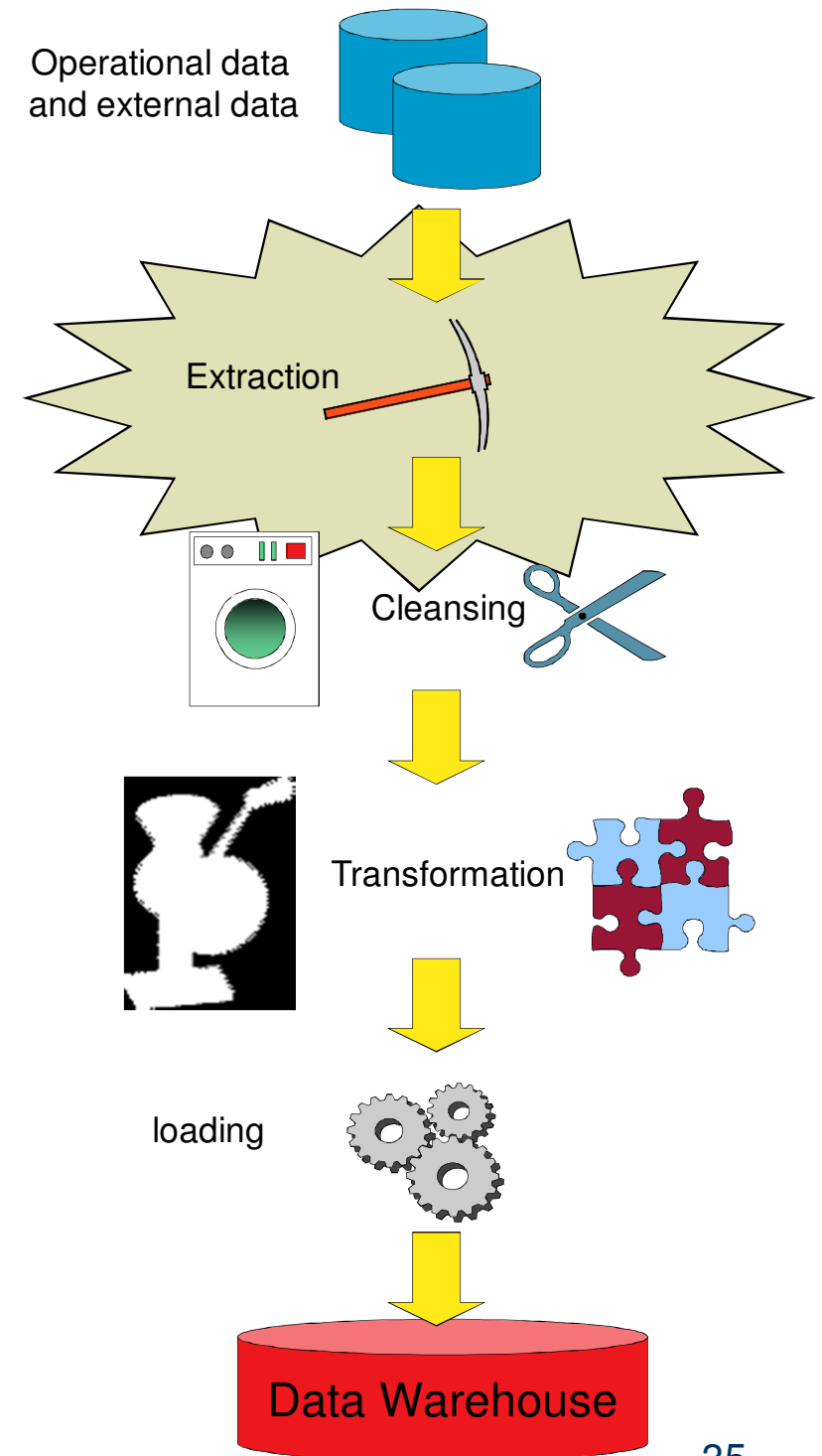
ETL

- ETL processes extract, integrate, and clean data from operational sources to eventually feed the data warehouse.
- At the abstract level, ETL processes produce a single, high-quality, detailed data source, that in turn feed the DW (*reconciliation*)
- Depending on the architecture, this data source is physical (reconciled data layer) or virtual. In the former case ETL processes are physically directly connected to the reconciled layer, in the latter to the (primary) DW or to the datamarts.
- Reconciliation takes place in two occasions: when a data warehouse is populated for the first time, and every time the data warehouse is updated.
- ETL consists in four phases:
 - ✓ extraction
 - ✓ cleansing (or cleaning)
 - ✓ transformation
 - ✓ loading

Note: Cleansing and transformation are not always considered as separate phases in the literature. Here we say that cleansing is essentially devoted to correct *values*, whereas transformation is mainly devoted to correct *formats*.

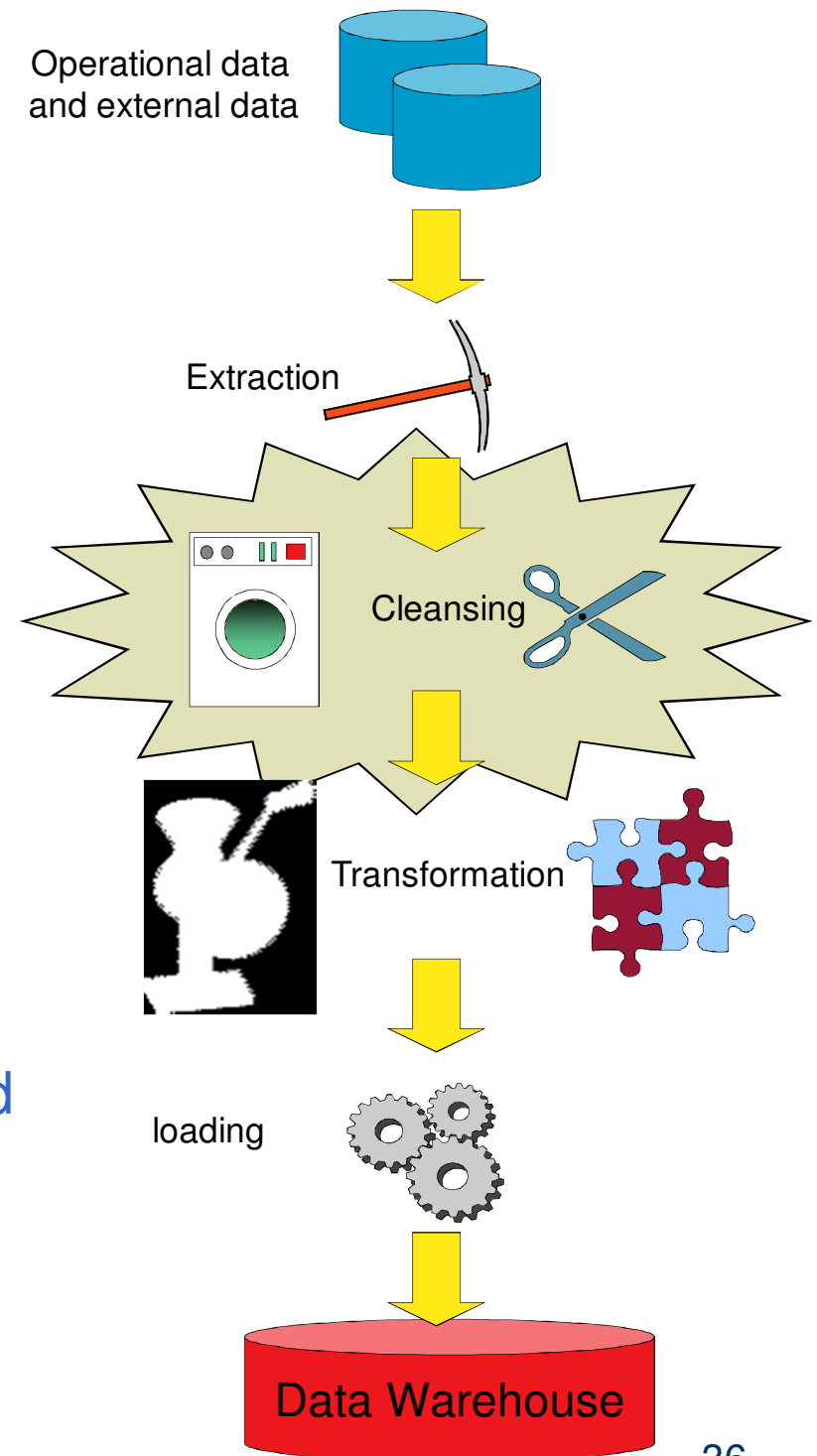
Extraction

- Relevant data is obtained from sources:
 - ✓ *Static extraction* is performed when a data warehouse needs populating for the first time. Conceptually speaking, this looks like a snapshot of operational data.
 - ✓ *Incremental extraction* is used to update data warehouses regularly, and seizes the changes applied to source data since the latest extraction.
 - often based on the log maintained by the operational DBMS
 - based on time-stamp
 - source-driven
- The data to be extracted is mainly selected on the basis of its quality.



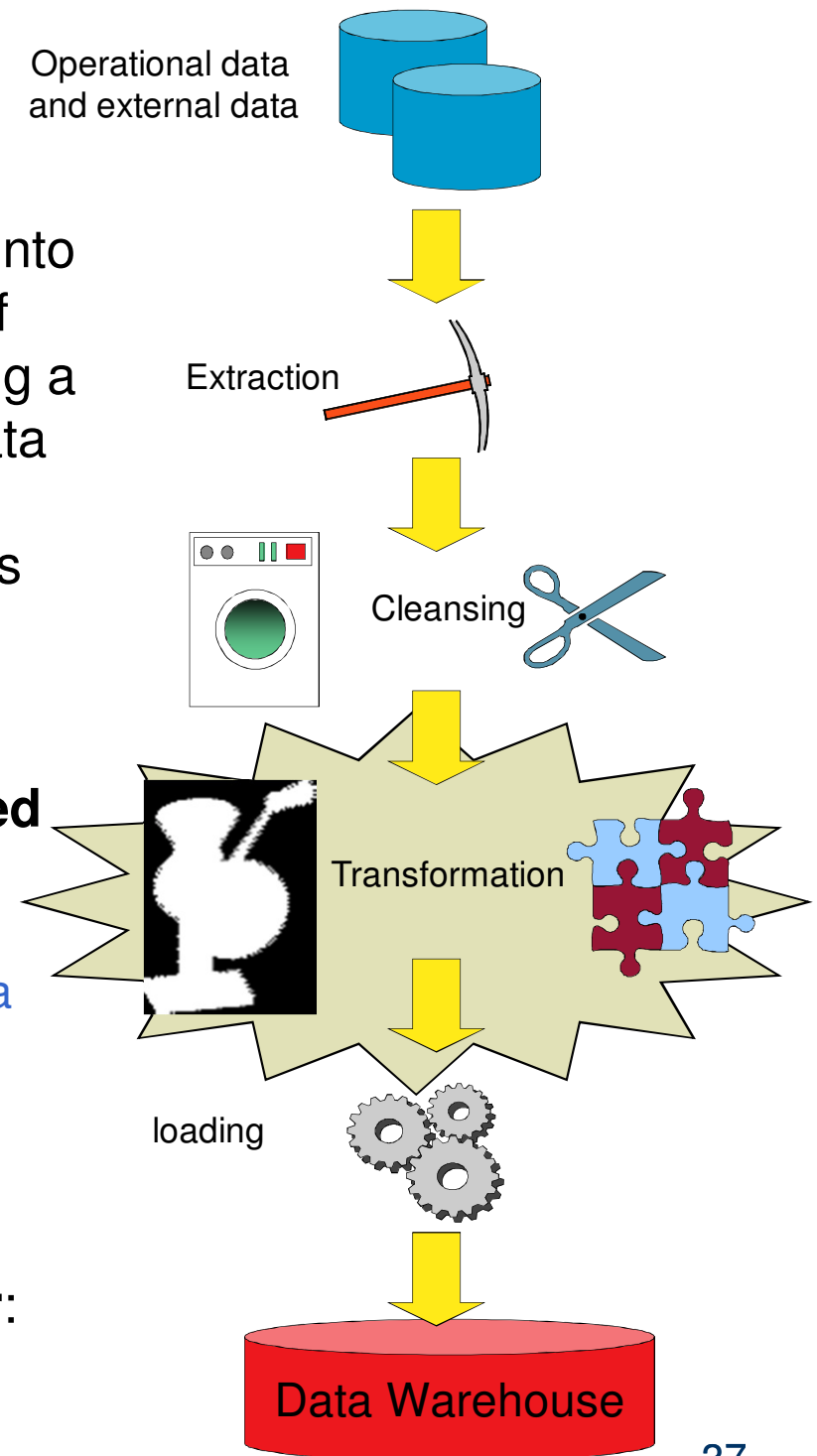
Cleansing

- It is supposed to improve data quality—normally quite poor in sources. The most frequent inconsistencies are due to
 - ✓ Duplicate data
 - ✓ Inconsistent values that are logically associated (*e.g., cities and zip codes*)
 - ✓ Missing Data
 - ✓ Unexpected use of fields
 - ✓ Impossible or wrong values
 - ✓ Inconsistent values for a single entity because different practices were used (*e.g., due to use of abbreviations*)
 - ✓ Inconsistent values for a single entity because of typing mistakes

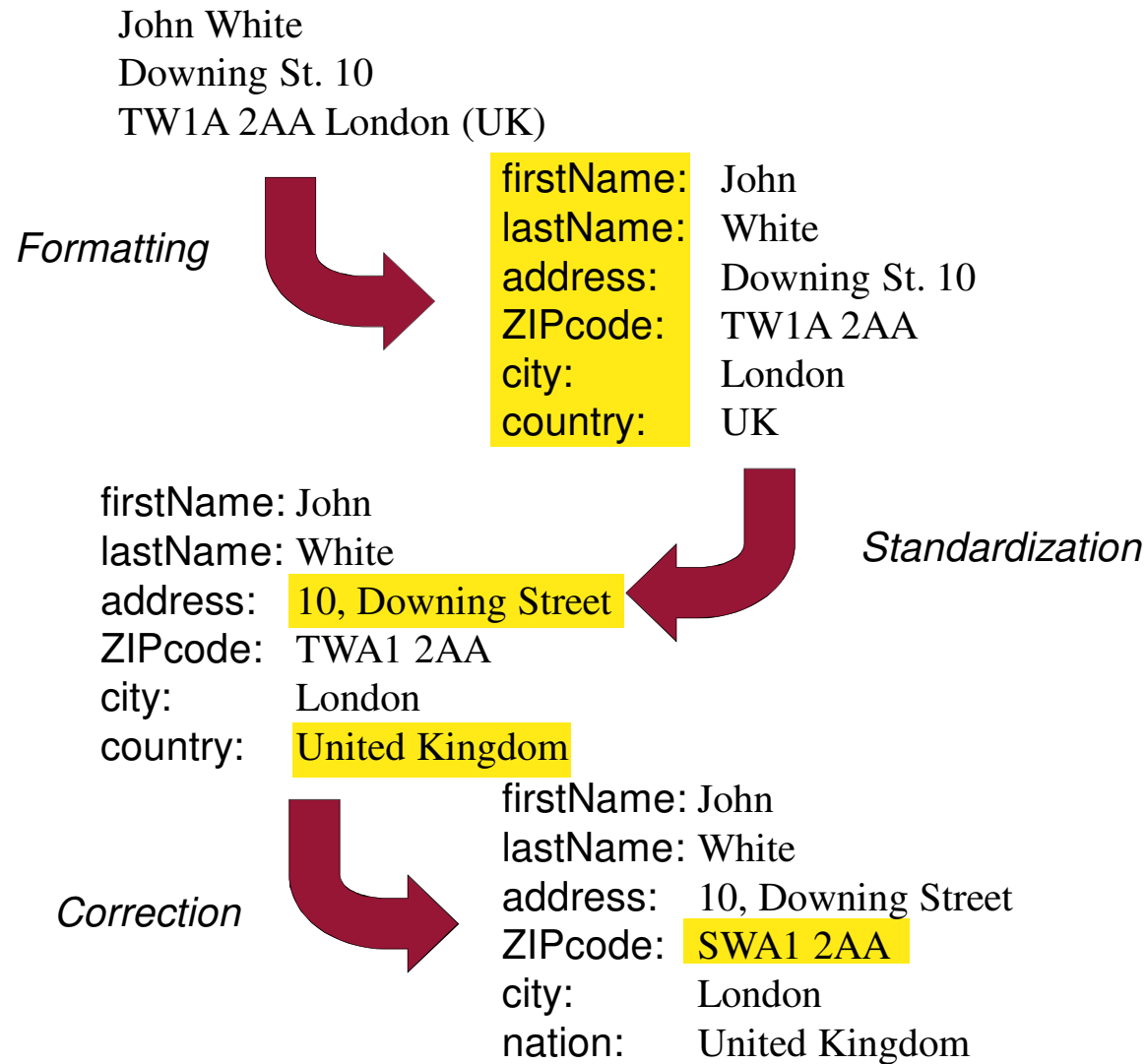


Transformation

- It converts data from its operational source format into a specific data warehouse format. Independently of the presence of a reconciled data layer, establishing a mapping between the source data layer and the data warehouse layer is generally made difficult by the presence of many different, heterogeneous sources
 - ✓ There may be loose texts that can hide valuable information
 - ✓ Different formats can be used for individual data
- main transformations for **populating the reconciled data layer**:
 - ✓ conversion and standardization that operate on both storage formats and units of measure to uniform data
 - ✓ matching that associates equivalent fields in different sources
 - ✓ selection that reduces the number of source fields and record
- main transformations for **populating the DW layer**:
 - ✓ normalization is substituted by denormalization
 - ✓ Aggregation is used to sum up data



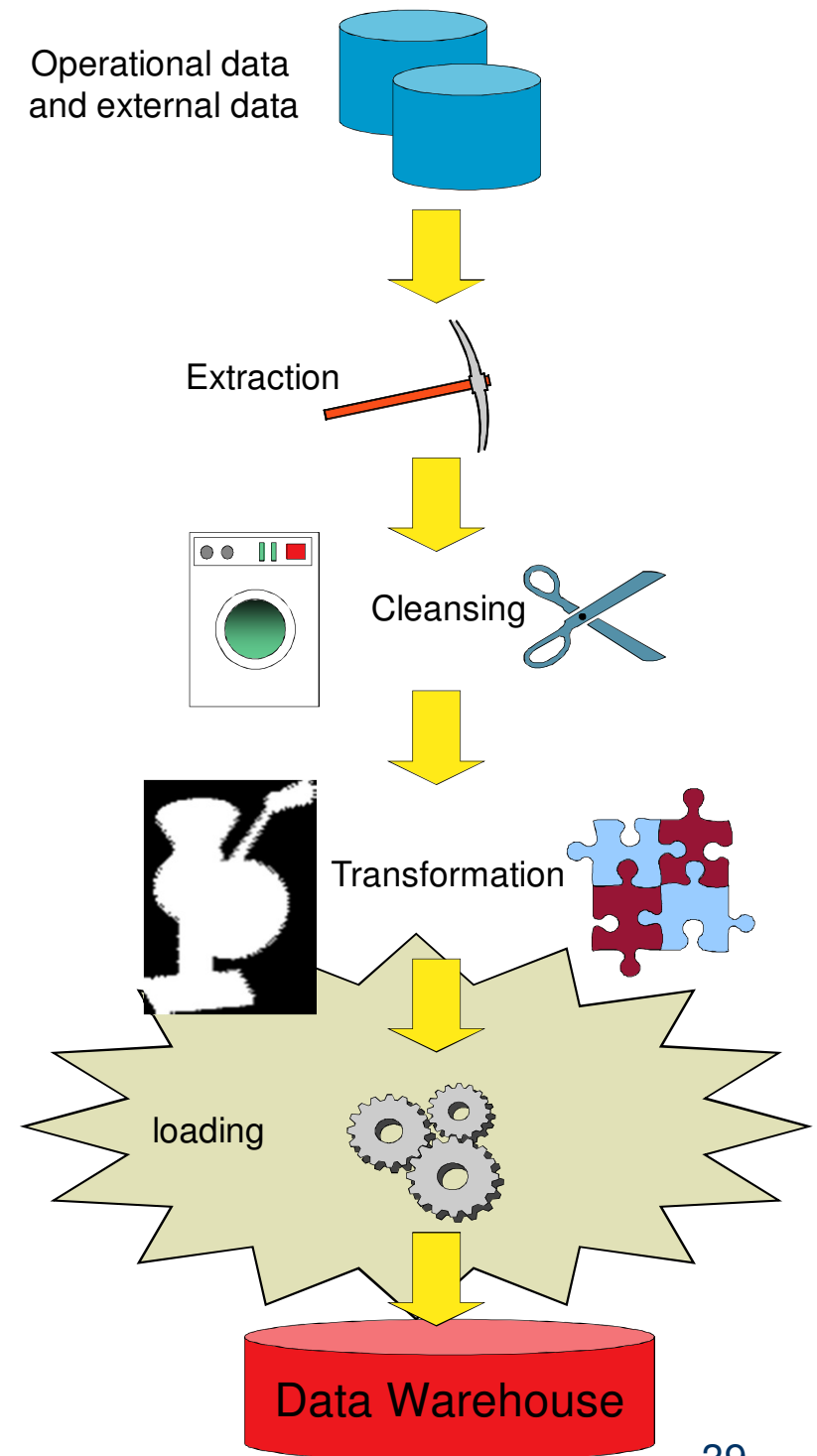
Cleansing and Transformation



Loading

■ Loading data into the DW

- ✓ **Refresh:** Data warehouse data is completely rewritten. This means that older data is replaced. Refresh is normally used in combination with static extraction to initially populate a data warehouse.
- ✓ **Update:** Only those changes applied to source data are added to the data warehouse. Update is typically carried out without deleting or modifying preexisting data. This technique is used in combination with incremental extraction to update data warehouses regularly.



Multidimensional Model

“What is the total amount of receipts recorded last year per state and per product category?”

“What is the relationship between the trend of PC manufacturers’ shares and quarter gains over the last five years?”

“Which orders maximize receipts?”

“Which one of two new treatments will result in a decrease in the average period of admission?”

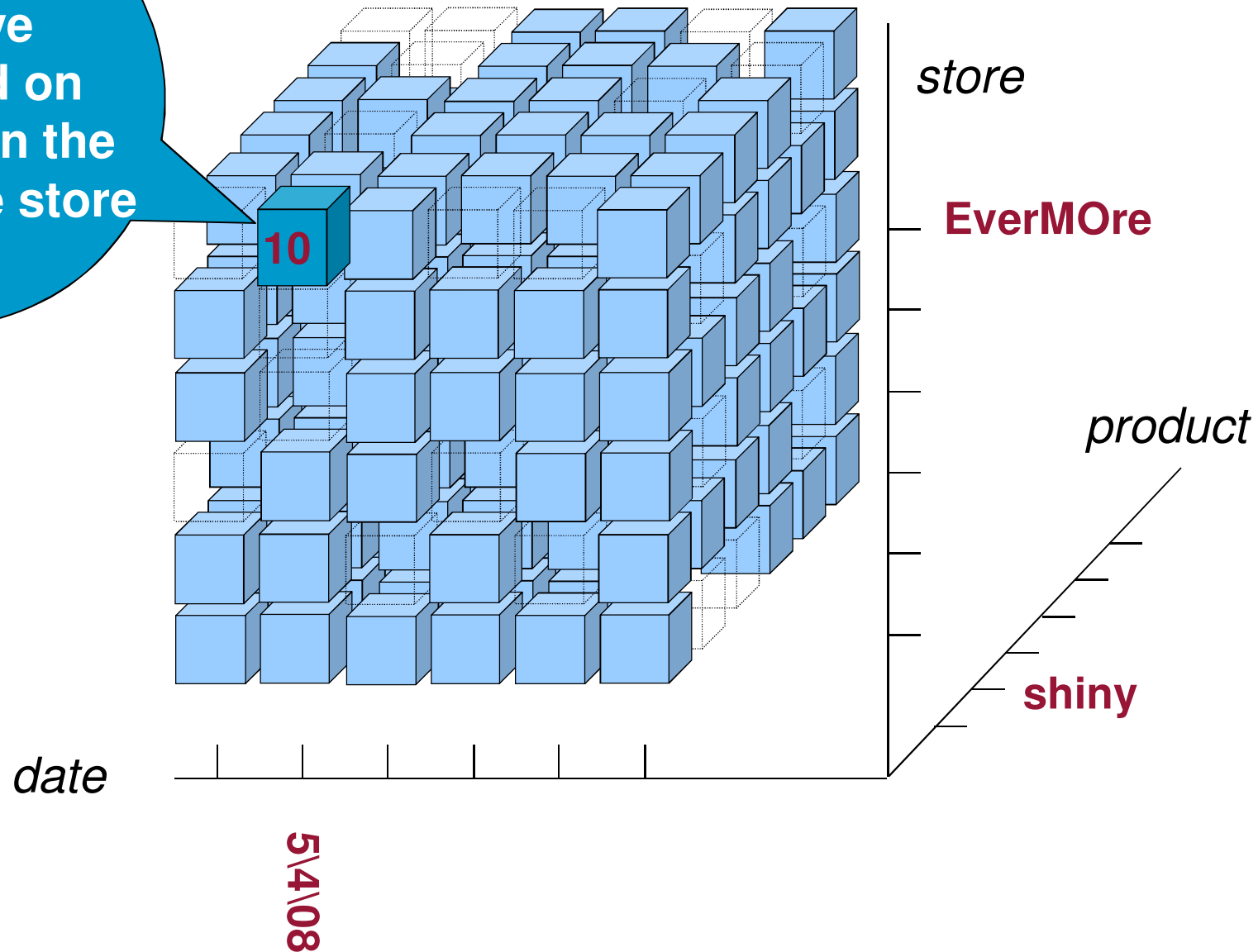
“What is the relationship between profit gained by the shipments consisting of less than 10 items and the profit gained by the shipments of more than 10 items?”

Multidimensional Model

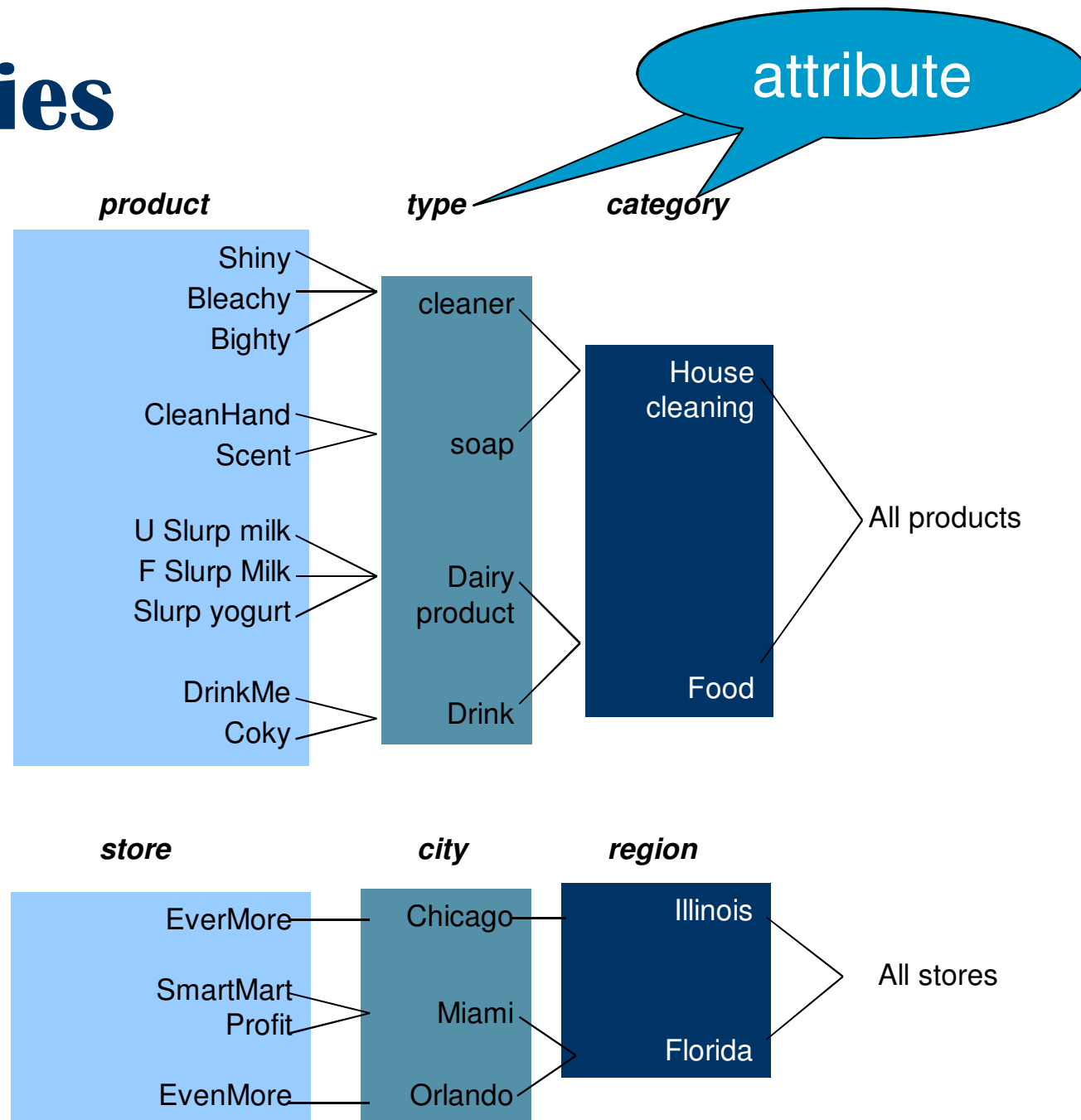
- It is the basic principled foundation for the representation and querying of data in a DW.
- Enterprise-specific *Facts* are represented as *cubes* where:
 - ✓ Each cell contains numerical *measures* that quantify the facts from various point of views.
 - ✓ Each axis represents a *dimension* of interest for the analysis.
 - ✓ Each dimension can be associated with a *hierarchy of dimensional attributes* used to aggregate data stored in the cubes.

Three-dimensional cube modeling sales

10 packs of shiny have been sold on 5/4/2008 in the EverMore store

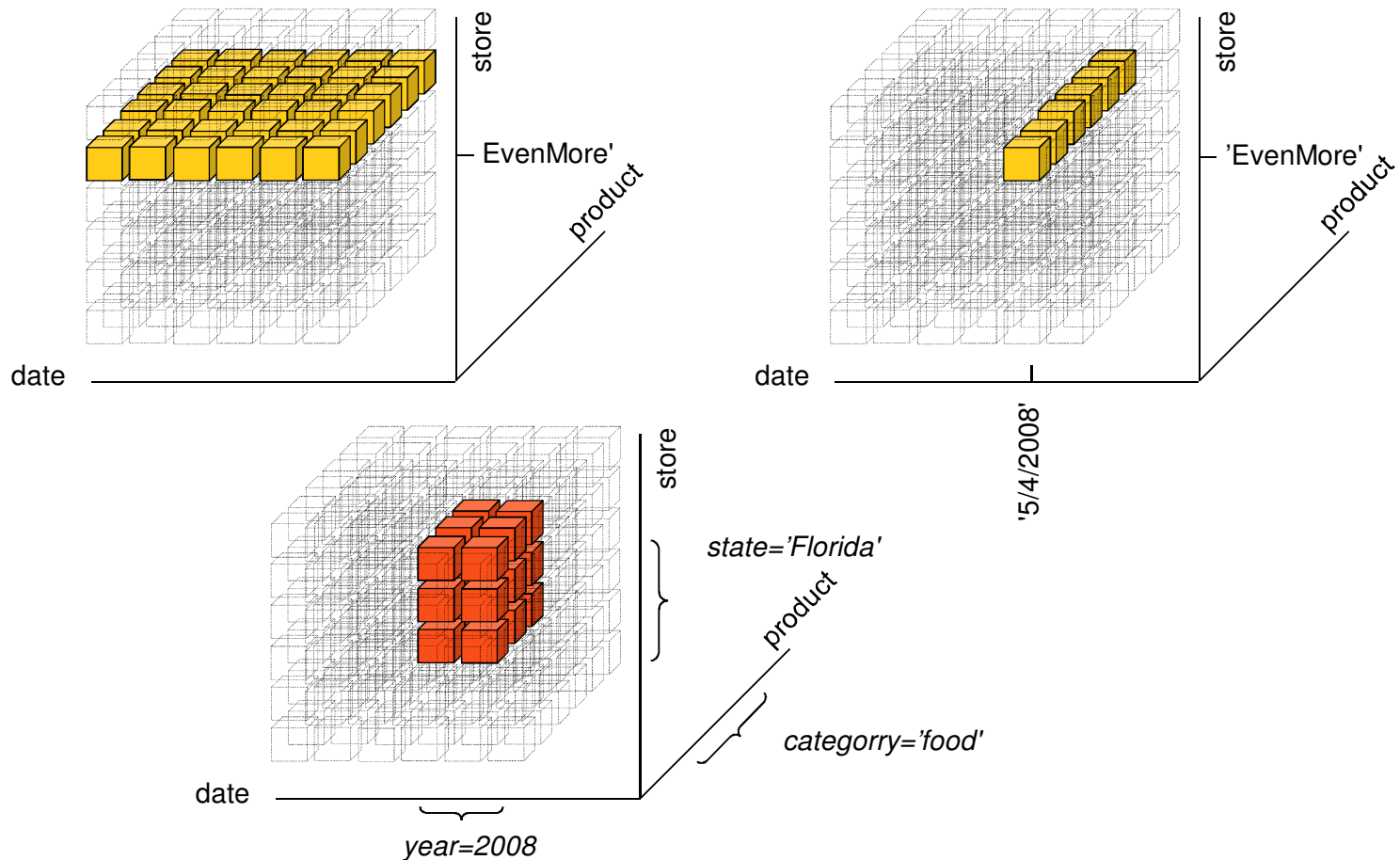


Hierarchies



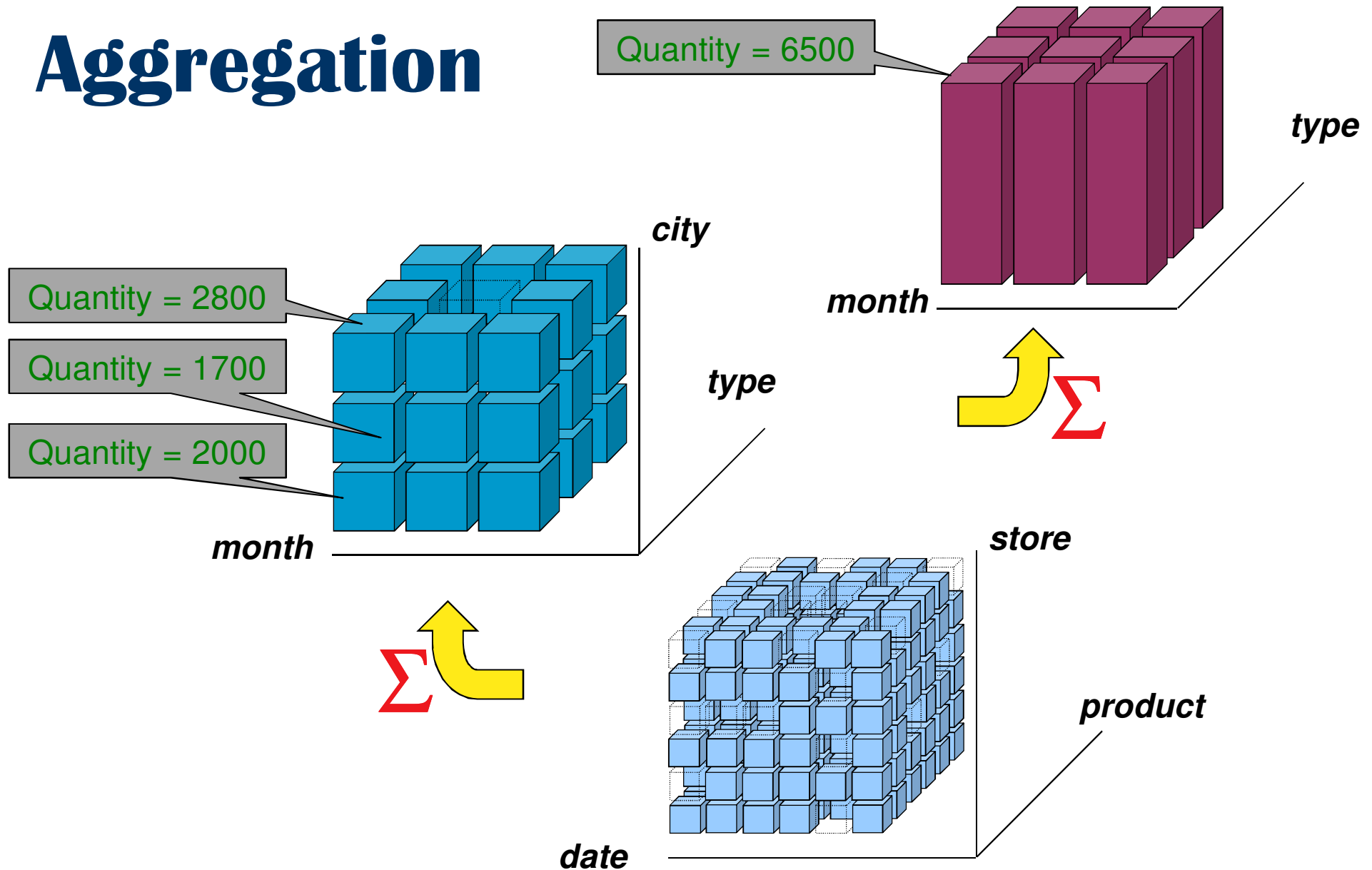
Functional dependencies: *product -> type -> category*
store -> city -> region

Restrictions



- When you **slice** data, you decrease cube dimensionality by setting one or more dimensions to a specific value.
- **Dicing** is a generalization of slicing. It poses some constraints on dimensional attributes (not only equalities – it acts on the hierarchy) to select only some cubes (notice that it does not aggregate).
- **Projection:** only a subset of measures are returned

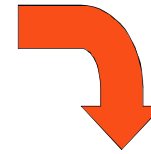
Aggregation



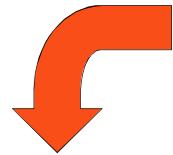
Every macro fact measure is the sum of its component fact measures

Aggregation

	EvenMore	EvenMore2	SmartMart
1/1/2000	–	–	–
2/1/2000	10	15	5
3/1/2000	20	–	5
.....
1/1/2001	–	–	–
2/1/2001	15	10	20
3/1/2001	20	20	25
.....
1/1/2002	–	–	–
2/1/2002	20	8	25
3/1/2002	20	12	20
.....



	EvenMore	EvenMore2	SmartMart
January 2000	200	180	150
February 2000	180	150	120
March 2000	220	180	160
.....
January 2001	350	220	200
February 2001	300	200	250
March 2001	310	180	300
.....
January 2002	380	200	220
February 2002	310	200	250
March 2002	300	160	280
.....



	EvenMore	EvenMore2	SmartMart
2000	2400	2000	1600
2001	3200	2300	3000
2002	3400	2200	3200



	EvenMore	EvenMore2	SmartMart
Total:	9000	6500	7800

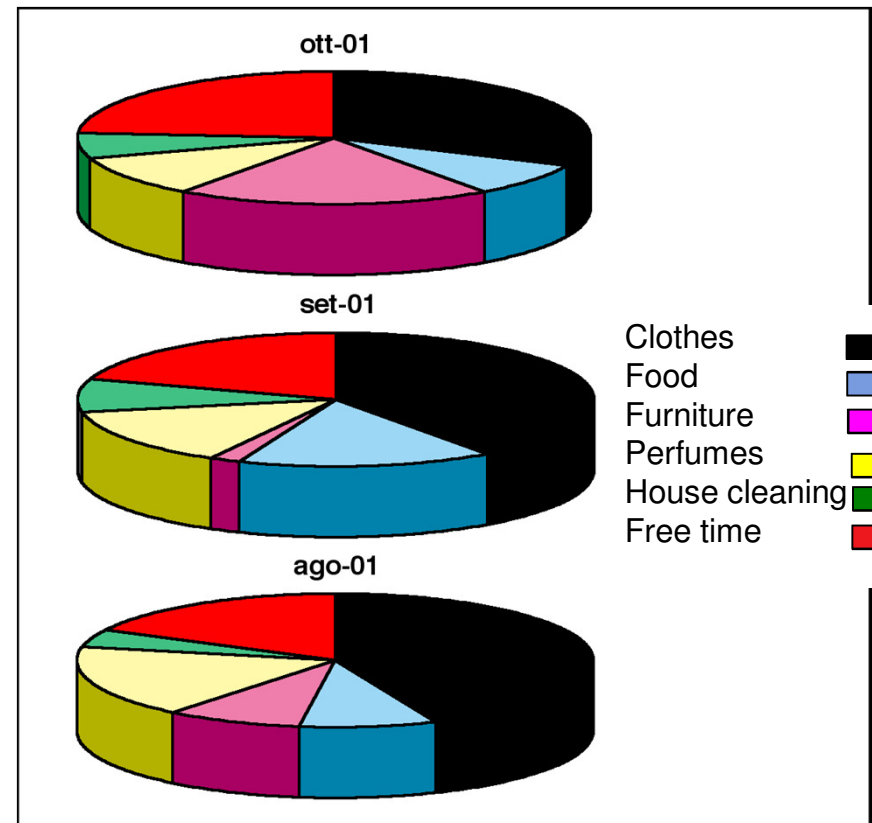
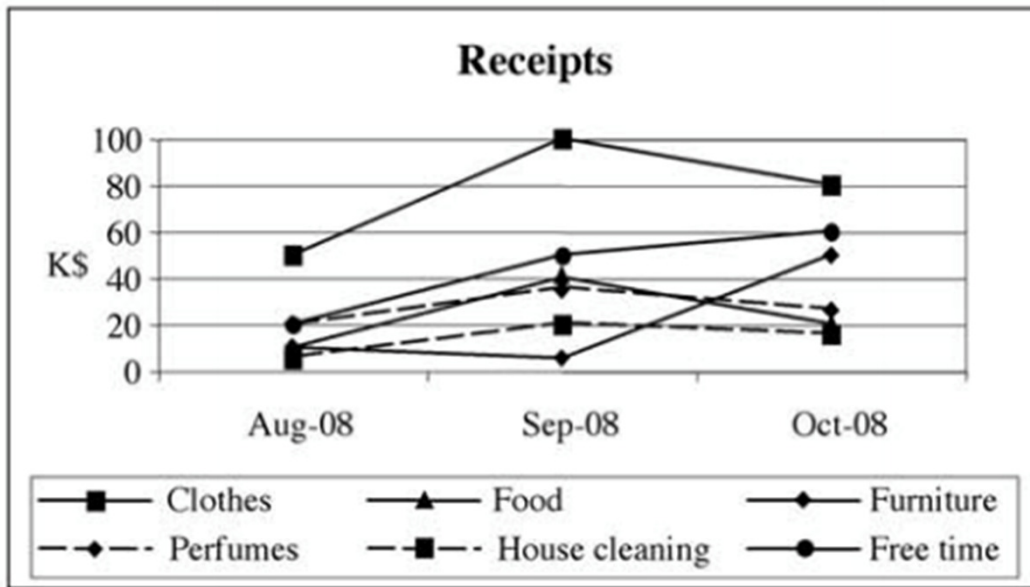
Accessing Data Warehousing

- After cleansing, integrating, and transforming data, you should determine how to get the best out of it in terms of information
- The three main approaches for end users to query data warehouses are essentially:
 - ✓ *reports*: It does not require IT knowledge
 - ✓ *OLAP*: requires the user to think in a multidimensional way and to know the interface of the graphical tool used
 - ✓ *dashboards*: GUIs that display a limited amount of relevant data in a brief and easy-to-read format
 - ✓ *data mining*: applies algorithms to identify hidden associations between items and make them visible to the user

Reports

Devoted to users that need to access information structured in an essentially static way, at predetermined time intervals

receipts (K€)	October 2001	September 2001	August 2001
clothes	80	100	50
food	20	40	10
forniture	50	5	10
Perfumes	25	35	20
House cleaning	15	20	5
Free time	60	50	20



OLAP

- It is the most popular way to exploit information in a data warehouse.
- it gives end users, whose analysis needs are not easy to define beforehand, the opportunity to analyze and explore data interactively on the basis of the multidimensional model.
- While users of reporting tools essentially play a passive role, OLAP users are able to start a complex analysis session actively, where each step is the result of the outcome of the preceding steps
 - ✓ in-depth knowledge of data by users is required
 - ✓ complex queries have to be allowed
 - ✓ design for users not familiar with IT has to be guaranteed



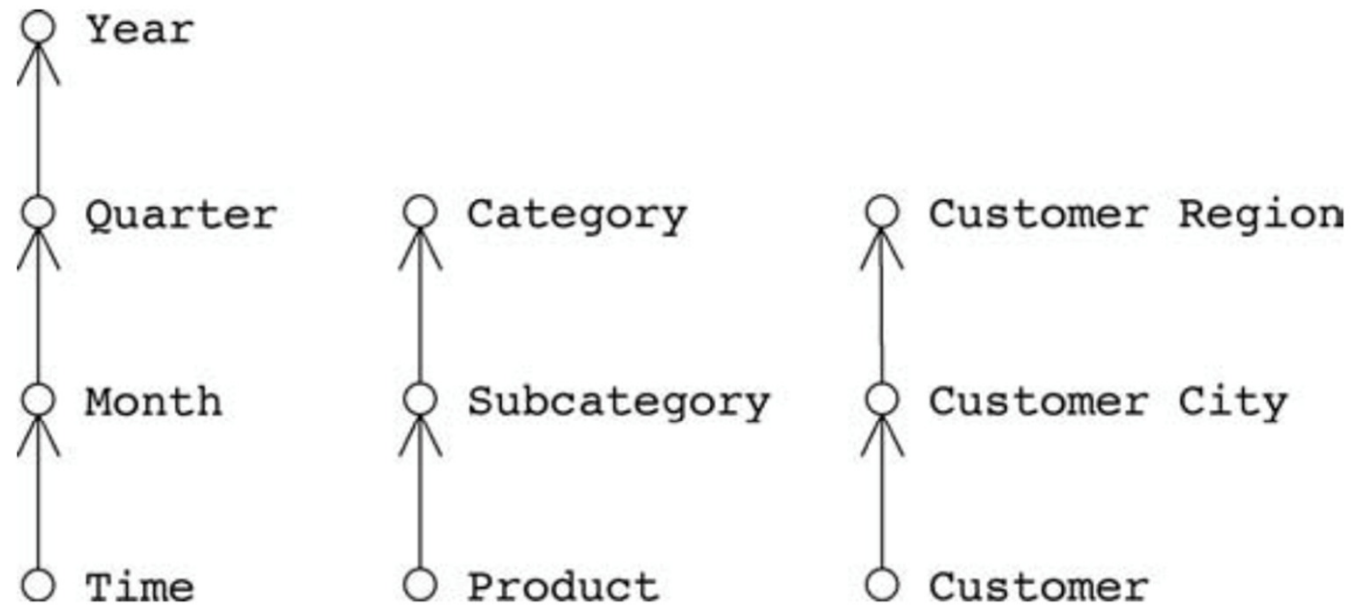
Flexible interface, easy to use and effective

OLAP: session

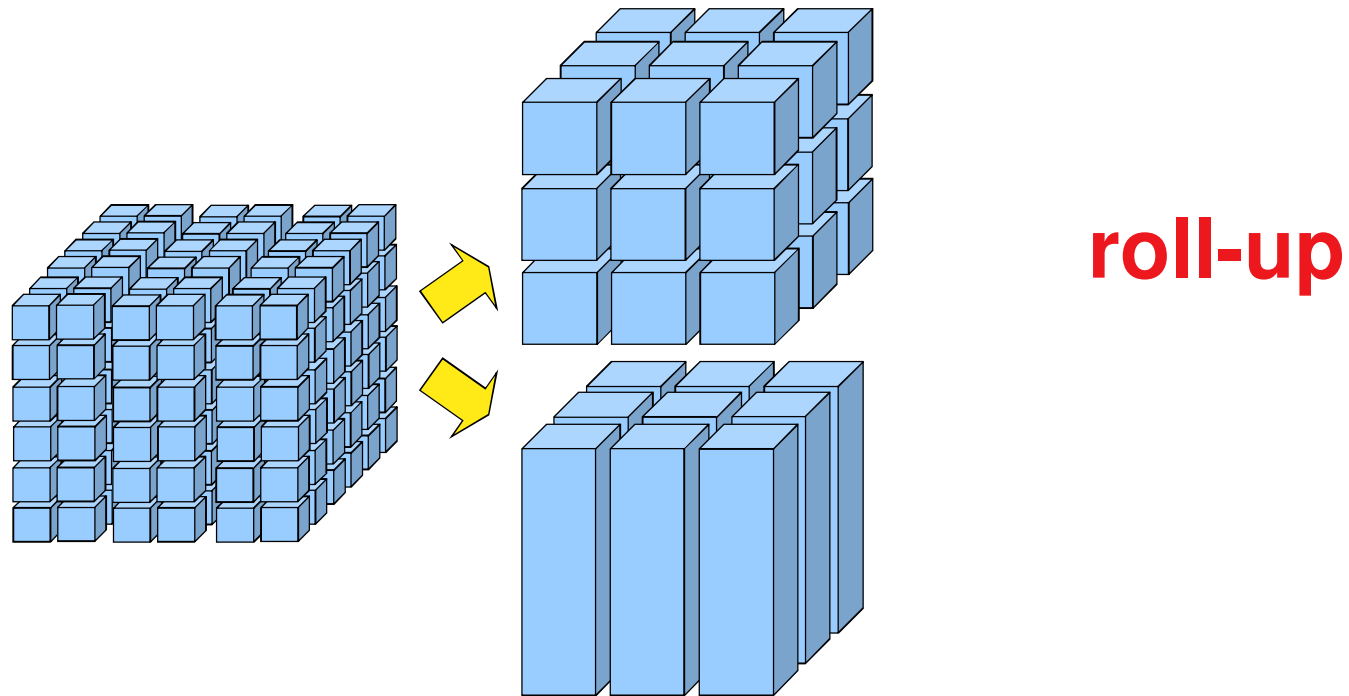
- An OLAP session consists of a *navigation path* that corresponds to an analysis process for facts according to different view points and different levels of detail. This path is turned into a sequence of queries, which are often not issued directly, but differentially expressed with reference to the previous query.
- At every step of the session an **OLAP operator** is used, which transforms the last performed query into a new one.
- The results of queries are multidimensional. OLAP tools typically use tables to display data, with multiple headers, colors, and other features to highlight data dimensions.

The Virtual-mall example

FIGURE 1-17
Attribute
hierarchies in
V-Mall; arrows
show functional
dependencies



OLAP: operators



The *roll-up* operator causes an increase in data aggregation and removes a detail level from a attribute hierarchy.

OLAP: operators

Month	Metrics	Revenue						
	Customer Region	Northeast	Mid-Atlantic	Southeast	Central	South	Northwest	Southwest
Jan 2005		\$160,155	\$518,405	\$81,381	\$322,294	\$98,001	\$103,368	\$298,730
Feb 2005		\$170,777	\$491,628	\$80,399	\$314,466	\$91,222	\$114,341	\$373,645
Mar 2005		\$200,434	\$611,424	\$129,102	\$382,946	\$123,038	\$147,472	\$351,602
Apr 2005		\$194,811	\$502,241	\$93,654	\$357,188	\$90,663	\$124,824	\$304,416
May 2005		\$169,998	\$462,364	\$117,780	\$300,389	\$79,999	\$117,506	\$311,123
Jun 2005		\$202,477	\$559,466	\$109,979	\$309,683	\$115,318	\$117,008	\$373,526
Jul 2005		\$194,490	\$577,515	\$105,099	\$332,300	\$92,730	\$103,494	\$369,380
Aug 2005		\$203,085	\$599,761	\$118,805	\$410,885	\$119,178	\$131,148	\$384,555
Sep 2005		\$241,992	\$625,517	\$122,261	\$415,763	\$75,655	\$124,974	\$364,651
Oct 2005		\$217,477	\$641,340	\$137,925	\$382,321	\$89,679	\$124,276	\$337,489
Nov 2005		\$238,004	\$708,036	\$156,525	\$457,105	\$116,478	\$156,466	\$386,399
Dec 2005		\$273,721	\$774,372	\$154,139	\$479,729	\$119,113	\$143,753	\$414,983
Jan 2006		\$215,786	\$662,632	\$125,238	\$392,922	\$91,791	\$122,235	\$343,027
Feb 2006		\$253,128	\$711,937	\$123,725	\$415,742	\$97,309	\$137,589	\$391,277
Mar 2006		\$253,564	\$704,652	\$135,180	\$430,143	\$112,459	\$144,659	\$406,956
Apr 2006		\$255,352	\$710,402	\$126,717	\$426,423	\$113,233	\$140,976	\$395,924
May 2006		\$231,766	\$676,205	\$130,981	\$440,813	\$107,277	\$136,043	\$377,349
Jun 2006		\$290,534	\$769,788	\$123,743	\$507,166	\$125,631	\$131,549	\$439,321
Jul 2006		\$247,683	\$811,060	\$145,955	\$448,939	\$113,683	\$128,113	\$415,251
Aug 2006		\$252,313	\$719,509	\$125,944	\$427,188	\$108,987	\$153,966	\$421,310
Sep 2006		\$288,772	\$801,819	\$148,023	\$539,406	\$112,784	\$149,236	\$419,878
Oct 2006		\$307,610	\$710,458	\$163,254	\$450,006	\$105,218	\$144,906	\$440,856
Nov 2006		\$284,671	\$800,941	\$157,117	\$505,952	\$118,552	\$163,560	\$470,591
Dec 2006		\$310,775	\$891,543	\$170,207	\$575,086	\$120,228	\$155,409	\$534,680



Quarter	Metrics	Revenue						
	Customer Region	Northeast	Mid-Atlantic	Southeast	Central	South	Northwest	Southwest
2005 Q1		\$531,366	\$1,621,457	\$290,882	\$1,019,706	\$312,261	\$365,181	\$1,023,977
2005 Q2		\$567,286	\$1,524,070	\$321,413	\$967,260	\$285,981	\$359,339	\$989,065
2005 Q3		\$639,567	\$1,802,793	\$346,165	\$1,158,948	\$287,563	\$359,616	\$1,118,587
2005 Q4		\$729,202	\$2,123,749	\$448,589	\$1,319,154	\$325,269	\$424,495	\$1,138,871
2006 Q1		\$722,478	\$2,079,221	\$384,143	\$1,238,807	\$301,559	\$404,483	\$1,141,260
2006 Q2		\$777,651	\$2,156,394	\$381,441	\$1,374,402	\$346,141	\$408,567	\$1,212,593
2006 Q3		\$788,768	\$2,332,388	\$419,923	\$1,415,533	\$335,455	\$431,315	\$1,256,439
2006 Q4		\$903,056	\$2,402,942	\$490,578	\$1,531,044	\$343,998	\$463,874	\$1,446,127

roll-up
(time hierarchy)

OLAP: operators

Category	Year	Metrics	Revenue						
		Customer Region	Northeast	Mid-Atlantic	Southeast	Central	South	Northwest	Southwest
Books	2005		\$416,183	\$316,104	\$36,517	\$207,850	\$137,502	\$19,062	\$187,368
	2006		\$534,932	\$401,908	\$42,027	\$239,806	\$138,683	\$22,655	\$183,275
Electronics	2005		\$1,860,172	\$6,517,723	\$1,226,825	\$3,719,752	\$915,633	\$1,434,575	\$3,625,191
	2006		\$2,403,311	\$8,253,620	\$1,451,397	\$4,631,259	\$999,611	\$1,615,848	\$4,298,985
Movies	2005		\$112,560	\$138,611	\$118,179	\$153,556	\$119,566	\$27,060	\$362,858
	2006		\$148,785	\$188,567	\$147,445	\$203,547	\$145,434	\$35,878	\$463,470
Music	2005		\$78,507	\$99,631	\$25,528	\$383,911	\$38,373	\$27,933	\$95,083
	2006		\$104,925	\$126,851	\$35,215	\$485,174	\$43,424	\$33,860	\$110,689

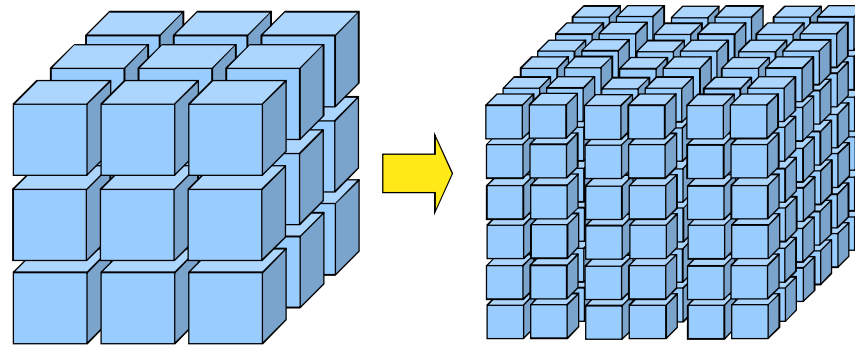


Category	Year	Metrics	Revenue
Books	2005		\$1,320,585
	2006		\$1,563,287
Electronics	2005		\$19,299,870
	2006		\$23,654,030
Movies	2005		\$1,032,391
	2006		\$1,333,126
Music	2005		\$748,966
	2006		\$940,136

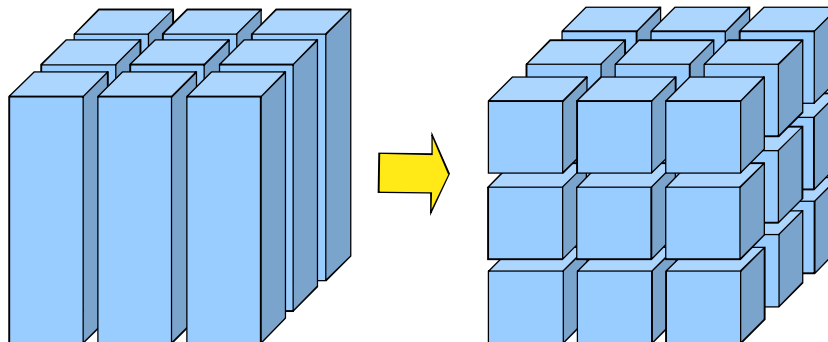
roll-up

(removing customer dimension)

OLAP: operators



drill-down



It is the inverse of roll-up. We use it to reduce the level of aggregation in the data

OLAP: operators

Quarter	Metrics	Revenue						
	Customer Region	Northeast	Mid Atlantic	Southeast	Central	South	Northwest	Southwest
2005 Q1		\$531,366	\$1,621,457	\$290,882	\$1,019,706	\$312,261	\$365,181	\$1,023,977
2005 Q2		\$567,286	\$1,524,070	\$321,413	\$967,260	\$285,981	\$359,339	\$989,065
2005 Q3		\$639,567	\$1,002,793	\$346,165	\$1,150,940	\$207,563	\$359,616	\$1,110,507
2005 Q4		\$729,202	\$2,123,749	\$440,509	\$1,319,154	\$325,269	\$424,495	\$1,130,071
2006 Q1		\$722,478	\$2,079,221	\$384,143	\$1,238,807	\$301,559	\$404,483	\$1,141,260
2006 Q2		\$777,651	\$2,156,394	\$381,441	\$1,374,402	\$346,141	\$408,567	\$1,212,593
2006 Q3		\$788,768	\$2,332,388	\$419,923	\$1,415,533	\$335,455	\$431,315	\$1,256,439
2006 Q4		\$903,056	\$2,402,942	\$490,578	\$1,531,044	\$343,998	\$463,874	\$1,446,127



drill-down

Quarter	Metrics	Revenue												
	Customer City	Addison	Akron	Albany	Albert City	Alexandria	Allentown	Anderson	Annapolis	Arden	Arlington Heights	Arlington	Artesia	A
2005 Q1		\$7,713	\$30,140	\$4,626	\$6,686	\$29,042	\$4,579	\$1,948	\$40,066	\$23,341	\$10,481		\$10,922	\$1
2005 Q2		\$15,903	\$48,029	\$8,959	\$2,088	\$17,590	\$7,268	\$2,416	\$42,764	\$21,026	\$7,514	\$1,695	\$2,984	\$
2005 Q3		\$10,091	\$30,510	\$5,763	\$11,380	\$26,389	\$10,195	\$568	\$51,650	\$25,132	\$10,784	\$3,796	\$8,701	\$
2005 Q4		\$12,425	\$48,588	\$10,939	\$10,463	\$28,016	\$10,426	\$3,412	\$67,515	\$28,398	\$14,692		\$9,975	\$1
2006 Q1		\$7,256	\$26,183	\$7,998	\$5,603	\$35,959	\$10,273	\$2,732	\$50,121	\$26,351	\$7,276	\$1,593	\$6,208	\$1
2006 Q2		\$10,411	\$49,540	\$6,065	\$5,670	\$25,166	\$6,992	\$1,377	\$68,198	\$27,556	\$16,755	\$3,420	\$6,656	\$1
2006 Q3		\$10,325	\$30,414	\$9,100	\$7,760	\$33,170	\$16,970	\$921	\$69,050	\$33,579	\$10,235	\$4,319	\$7,636	\$1
2006 Q4		\$16,613	\$49,133	\$13,137	\$10,637	\$30,344	\$13,075	\$747	\$40,305	\$32,042	\$13,311	\$6,176	\$9,191	\$1

OLAP: operators

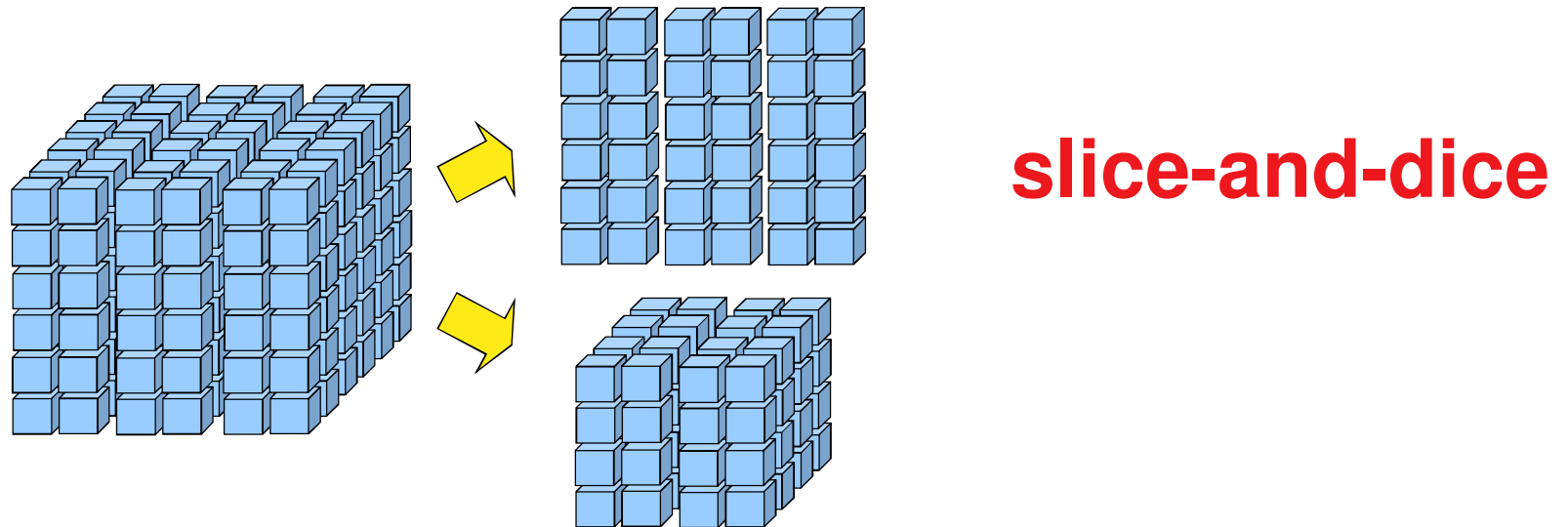
Category	Metrics	Dollar Sales	
	Year	1997	1998
Electronics		\$ 10.616	\$ 29.299
Food		\$ 5.300	\$ 5.638
Gifts		\$ 16.315	\$ 20.047
Health & Beauty		\$ 6.042	\$ 5.665
Household		\$ 38.383	\$ 50.391
Kid's Komer		\$ 2.559	\$ 2.943
Travel		\$ 4.497	\$ 4.792



drill-down
(adding a dimension)

Filter Details: Year = 1998												
Category	Metrics Customer Region	Dollar Sales										
		North-East	Mid-Atlantic	South-East	Central	South	North-West	South-West	England	France	Germany	Ca
Electronics		\$ 1.184	\$ 4.529	\$ 1.892	\$ 7.232	\$ 651	\$ 9.488	\$ 476	\$ 2.683	\$ 462	\$ 702	
Food		\$ 538	\$ 925	\$ 959	\$ 677	\$ 213	\$ 1.503	\$ 261	\$ 165	\$ 175	\$ 100	
Gifts		\$ 1.955	\$ 2.785	\$ 2.800	\$ 2.695	\$ 1.813	\$ 2.844	\$ 1.778	\$ 1.158	\$ 717	\$ 686	
Health & Beauty		\$ 611	\$ 887	\$ 566	\$ 382	\$ 499	\$ 1.162	\$ 1.044	\$ 273	\$ 72		
Household		\$ 5.787	\$ 5.320	\$ 5.416	\$ 6.812	\$ 4.334	\$ 5.008	\$ 7.588	\$ 2.139	\$ 3.649	\$ 2.791	
Kid's Komer		\$ 247	\$ 422	\$ 441	\$ 380	\$ 221	\$ 592	\$ 290	\$ 198	\$ 19	\$ 69	
Travel		\$ 608	\$ 559	\$ 1.096	\$ 611	\$ 464	\$ 316	\$ 573	\$ 257	\$ 198	\$ 55	

OLAP: operators



It produces “sub-cubes” starting from cubes. Typically, this is realized through simple or complex selections.

- **Slicing**: reduces the number of cube dimensions after setting one of the dimension (i.e., the most fine-grained attribute of a dimension) to a specific value
- **Dicing**: reduces the set of data being analyzed by a selection criterion

OLAP: operators

Category	Year	Metrics Customer Region	Revenue						
			Northeast	Mid-Atlantic	Southeast	Central	South	Northwest	Southwest
Books	2005		\$416,183	\$316,104	\$36,517	\$207,850	\$137,502	\$19,062	\$187,368
	2006		\$534,932	\$401,908	\$42,027	\$239,806	\$138,683	\$22,655	\$183,275
Electronics	2005		\$1,860,172	\$6,517,723	\$1,226,825	\$3,719,752	\$915,633	\$1,434,575	\$3,625,191
	2006		\$2,403,311	\$8,253,620	\$1,451,397	\$4,631,259	\$999,611	\$1,615,848	\$4,298,985
Movies	2005		\$112,560	\$138,611	\$118,179	\$153,556	\$119,566	\$27,060	\$362,858
	2006		\$148,785	\$188,567	\$147,445	\$203,547	\$145,434	\$35,878	\$463,470
Music	2005		\$78,507	\$99,631	\$25,528	\$383,911	\$38,373	\$27,933	\$95,083
	2006		\$104,925	\$126,851	\$35,215	\$485,174	\$43,424	\$33,860	\$110,689



slice-and-dice

Slicing based on the Year='2006' predicate

Report Filter (Local Filter):
Year = 2006

Category	Metrics Customer Region	Revenue						
		Northeast	Mid-Atlantic	Southeast	Central	South	Northwest	Sout
Books		\$534,932	\$401,908	\$42,027	\$239,806	\$138,683	\$22,655	\$1
Electronics		\$2,403,311	\$8,253,620	\$1,451,397	\$4,631,259	\$999,611	\$1,615,848	\$4,2
Movies		\$148,785	\$188,567	\$147,445	\$203,547	\$145,434	\$35,878	\$4
Music		\$104,925	\$126,851	\$35,215	\$485,174	\$43,424	\$33,860	\$1

Subcategory	Metrics Customer City	Revenue									
		Addison	Akron	Albany	Albert City	Alexandria	Allentown	Anderson	Annapolis	Arden	Arlin Heig
Art & Architecture		\$253	\$365	\$1,506	\$279	\$268	\$1,960		\$407	\$282	
Business		\$198	\$357	\$719	\$75	\$134	\$1,225	\$8	\$304	\$184	
Literature		\$92	\$116	\$277	\$54	\$66	\$503		\$137	\$128	
Books - Miscellaneous		\$216	\$95	\$830	\$120	\$73	\$605	\$4	\$233	\$71	
Science & Technology		\$363	\$943	\$3,271	\$578	\$491	\$2,547		\$834	\$366	
Sports & Health		\$220	\$153	\$416	\$165	\$128	\$1,476		\$409	\$204	
Audio Equipment		\$4,782	\$32,102	\$2,082	\$4,318	\$36,458	\$7,303	\$4,130	\$77,501	\$17,307	\$-
Cameras		\$7,671	\$39,381	\$5,810	\$3,898	\$12,045	\$8,124	\$405	\$33,744	\$11,840	\$-
Computers		\$1,531	\$14,810	\$1,935	\$3,022	\$7,967	\$3,354		\$5,097	\$7,087	\$-
Electronics - Miscellaneous		\$4,667	\$25,861	\$5,289	\$5,979	\$25,484	\$4,821	\$130	\$9,533	\$11,923	\$-
TV's		\$9,027	\$34,635	\$5,013	\$3,500	\$33,432	\$4,930		\$46,933	\$32,527	\$-
Video Equipment		\$7,422	\$10,078	\$5,540	\$2,479	\$5,500	\$7,470	\$540	\$55,552	\$35,723	\$-
Action		\$108	\$302	\$204	\$134	\$256	\$259	\$25	\$859	\$117	
Comedy		\$308	\$520	\$355	\$195	\$204	\$261		\$361	\$216	
Drama		\$424	\$548	\$225	\$243	\$313	\$399		\$511	\$415	
Horror		\$339	\$196	\$141	\$195	\$188	\$372		\$261	\$184	
Kids / Family		\$361	\$277	\$236	\$213	\$235	\$460	\$11	\$744	\$323	
Special Interests		\$700	\$670	\$353	\$289	\$179	\$695		\$816	\$288	
Alternative		\$1,077	\$236	\$114	\$484	\$234	\$139		\$365	\$173	\$-
Country		\$1,169	\$336	\$310	\$576	\$206	\$123	\$43	\$283	\$288	
Music - Miscellaneous		\$1,193	\$364	\$167	\$494	\$254	\$281	\$18	\$410	\$132	\$-
Pop		\$541	\$286	\$231	\$302	\$130	\$178	\$26	\$350	\$173	
Rock		\$1,184	\$324	\$202	\$490	\$161	\$341	\$14	\$285	\$127	
Soul / R&B		\$760	\$225	\$182	\$1,496	\$223	\$293	\$23	\$463	\$159	\$-

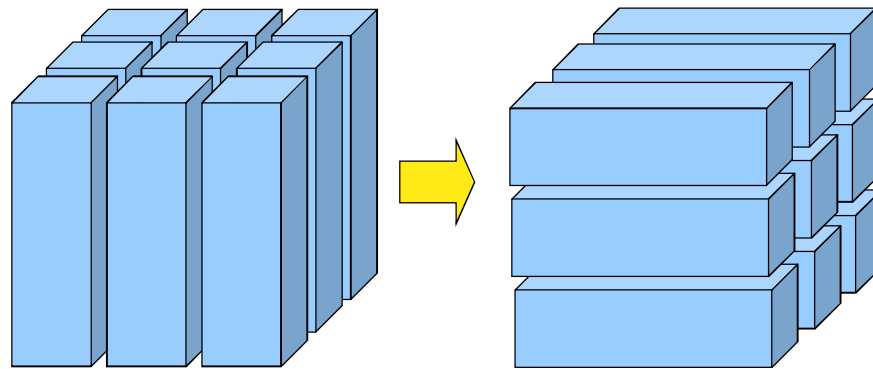


slice-and-dice

Subcategory	Metrics Customer City	Revenue				
		Bellevue	Bothell	Caldwell	Cheyenne	Coulee City
Audio Equipment		\$9,945	\$23,211	\$2,531	\$2,347	\$30,857
Cameras		\$31,245	\$5,613	\$5,592	\$1,240	\$11,250
Computers		\$12,751	\$4,771	\$1,443	\$713	\$8,181
Electronics - Miscellaneous		\$13,940	\$29,072	\$3,722	\$2,444	\$16,254
TV's		\$40,652	\$9,788	\$3,990	\$1,871	\$10,846
Video Equipment		\$36,423	\$26,363	\$3,543	\$3,947	\$5,480

Year = '2006' AND
 Category = 'Electronics' AND
 Revenue > 80 AND
 CustomerRegion = 'Northwest'

OLAP: operators



pivoting

It is used to change the structure of the cube, that is, when a different view of the data in the cube is needed.

OLAP: operators

Category	Year	Metrics	Revenue
Books	2005		\$1,320,585
	2006		\$1,563,287
Electronics	2005		\$19,299,870
	2006		\$23,654,030
Movies	2005		\$1,032,391
	2006		\$1,333,126
Music	2005		\$748,966
	2006		\$940,136



Category	Metrics	Revenue	
	Year	2005	2006
Books		\$1,320,585	\$1,563,287
Electronics		\$19,299,870	\$23,654,030
Movies		\$1,032,391	\$1,333,126
Music		\$748,966	\$940,136

pivoting

OLAP: operators

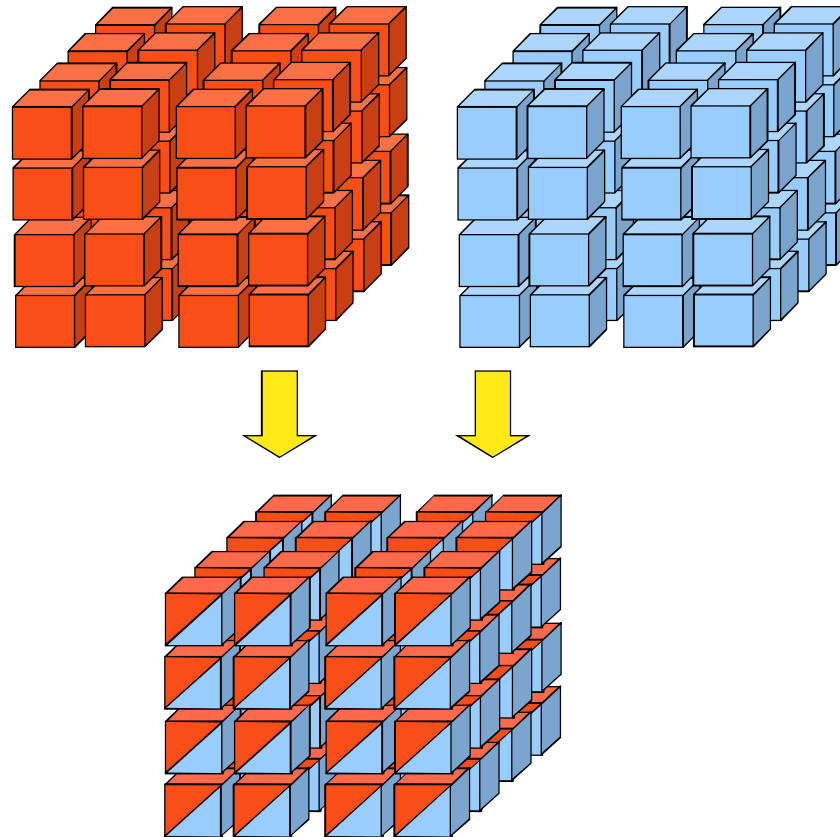
Category	Year	Metrics Customer Region	Revenue						
			Northeast	Mid Atlantic	Southeast	Central	South	Northwest	Southwest
Books	2005		\$416,183	\$316,104	\$36,517	\$207,850	\$137,502	\$19,062	\$187,368
	2006		\$534,932	\$401,908	\$42,027	\$239,806	\$138,683	\$22,655	\$183,275
Electronics	2005		\$1,860,172	\$6,517,723	\$1,226,825	\$3,719,752	\$915,633	\$1,434,575	\$3,625,191
	2006		\$2,403,311	\$8,253,620	\$1,451,397	\$4,631,259	\$999,611	\$1,615,848	\$4,298,985
Movies	2005		\$112,560	\$138,611	\$118,179	\$153,556	\$119,566	\$27,060	\$362,858
	2006		\$148,785	\$188,567	\$147,445	\$203,547	\$145,434	\$35,878	\$463,470
Music	2005		\$78,507	\$99,631	\$25,528	\$383,911	\$38,373	\$27,933	\$95,083
	2006		\$104,925	\$126,851	\$35,215	\$485,174	\$43,424	\$33,860	\$110,689



pivoting

Category	Metrics Customer Region Year	Revenue										
		Northeast		Mid Atlantic		Southeast		Central		South		Northwest
		2005	2006	2005	2006	2005	2006	2005	2006	2005	2006	2005
Books		\$416,183	\$534,932	\$316,104	\$401,908	\$36,517	\$42,027	\$207,850	\$239,806	\$137,502	\$138,683	\$19,062
Electronics		\$1,860,172	\$2,403,311	\$6,517,723	\$8,253,620	\$1,226,825	\$1,451,397	\$3,719,752	\$4,631,259	\$915,633	\$999,611	\$1,434,575
Movies		\$112,560	\$148,785	\$138,611	\$188,567	\$118,179	\$147,445	\$153,556	\$203,547	\$119,566	\$145,434	\$27,060
Music		\$78,507	\$104,925	\$99,631	\$126,851	\$25,528	\$35,215	\$383,911	\$485,174	\$38,373	\$43,424	\$27,933

OLAP: operators



drill-across

It combines data in different cubes to obtain a new cube.

OLAP: operators

Category	Metrics	Revenue							
	Quarter	2005 Q1	2005 Q2	2005 Q3	2005 Q4	2006 Q1	2006 Q2	2006 Q3	2006 Q4
Books		\$319,767	\$313,339	\$336,862	\$350,617	\$348,483	\$387,849	\$407,392	\$419,563
Electronics		\$4,448,112	\$4,299,411	\$4,918,673	\$5,633,676	\$5,411,499	\$5,714,783	\$5,999,174	\$6,528,576
Movies		\$228,108	\$232,201	\$264,471	\$307,611	\$299,531	\$326,270	\$334,143	\$373,182
Music		\$168,843	\$169,462	\$193,234	\$217,427	\$212,438	\$228,289	\$239,112	\$260,298



+ discount data

drill-across

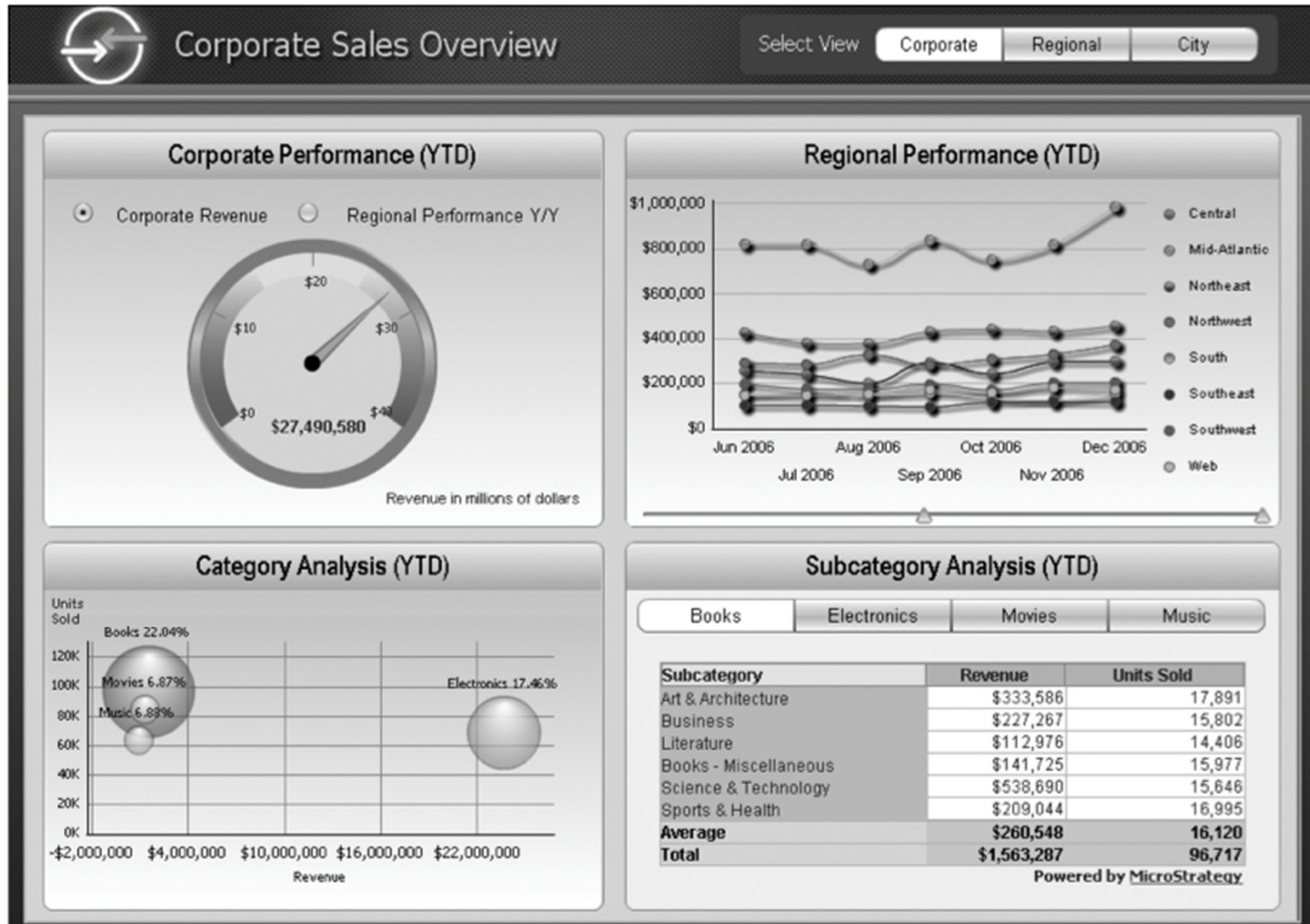
Category	Quarter	2005 Q1		2005 Q2		2005 Q3		2005 Q4		2006 Q1		
		Metrics	Discount	Revenue	Discount	Revenue	Discount	Revenue	Discount	Revenue	Discount	Revenue
Books			\$ 0	\$319,767	\$ 10,845	\$313,339	\$ 9,497	\$336,862	\$ 18,279	\$350,617	\$ 0	\$348,483
Electronics			\$ 0	\$4,448,112	\$ 150,366	\$4,299,410	\$ 143,395	\$4,918,673	\$ 302,884	\$5,633,675	\$ 0	\$5,411,499
Movies			\$ 0	\$228,108	\$ 8,025	\$232,201	\$ 7,948	\$264,471	\$ 16,649	\$307,611	\$ 0	\$299,531
Music			\$ 0	\$168,843	\$ 6,143	\$169,462	\$ 5,563	\$193,234	\$ 11,047	\$217,427	\$ 0	\$212,438

Drilling across the sales cube (Revenue measure) and the promotion cube (Discount measure)

Dashboards

- Dashboards can provide a **real-time overview of the trends for a specific phenomenon** or for many phenomena that are strictly connected with each other.
- The term is a visual metaphor: the group of indicators in the GUI are displayed like a car dashboard.
- Dashboards are often used by senior managers who need a quick way to view information. However, to conduct and display very complex analyses of phenomena, dashboards must be matched with analysis tools.
- Dashboards are nothing but **performance indicators** behind GUIs. Their effectiveness is due to a careful selection of the relevant measures, while using data warehouse information quality standard.

Dashboards



Dashboard created with MicroStrategy Dynamic Enterprise

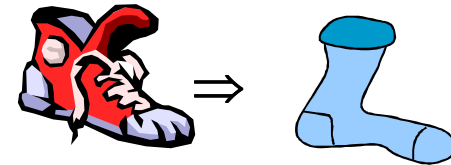
Data mining

- Technique aimed at discovering information hidden in the data
 - ✓ In the presence of large-scale databases, users are not always able to find out all significant patterns (models)
 - ✓ Data mining puts together a set of techniques and methodologies from artificial intelligence and pattern recognition, such as genetic algorithm, fuzzy logic, expert systems, neural networks, ecc.
 - ✓ To discover an hidden pattern it is sufficient to indicate what and where to look for.
 - ✓ Examples of applications are:
 - Market research
 - Study of the marketing effectiveness
 - Market segmentation
 - Analysis of the buying habits
 - business Planning
 - Modeling of investments
 - Detection of fraudulent activities
 - risk category assessment
 - Recognition of similarities between sequences of events
 - Assessment of clinical cases and study of epidemiological models

Data mining: association rules

- Association rules make it possible to determine the logical implication rules that exist in databases
- Example: purchasing transactions

{shoes} \Rightarrow {socks}



The transactions containing shoes tend to contain also socks.

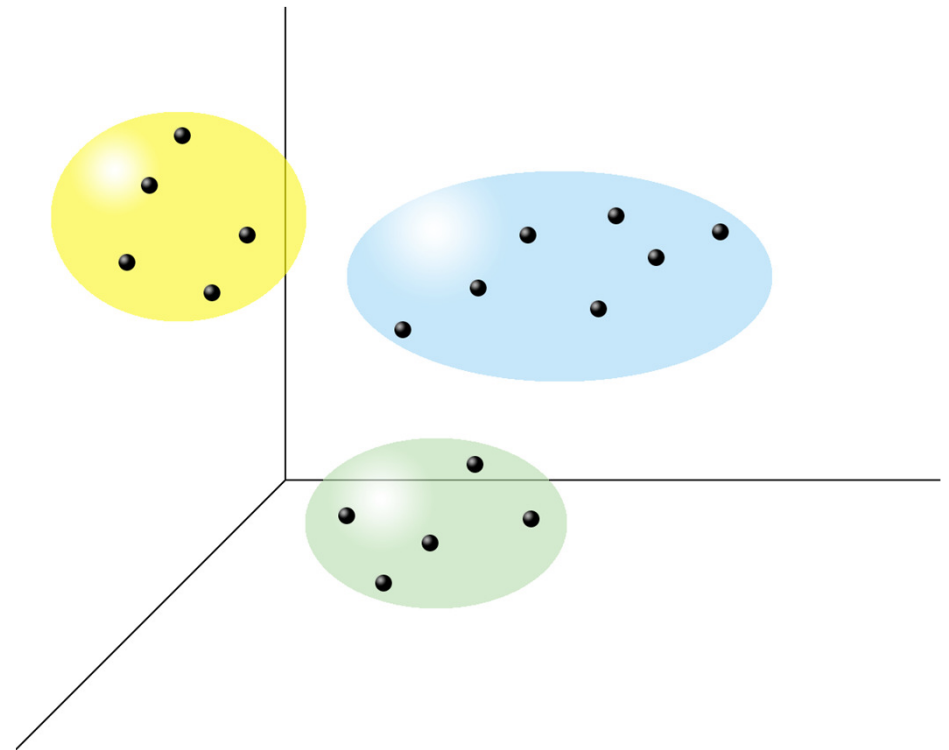
- Two measures are commonly added to evaluate how useful a rule of the form $\{X\} \Rightarrow \{Y\}$ is:
 - ✓ **Support:** The percentage of transactions containing both X and Y
 - ✓ **Confidence:** Percentage of transactions that also contain Y among those containing X

Data mining: association rules

- Extracting association rules normally involves determining all the rules whose support and confidence figures are above a certain threshold.
- Various algorithms for this exist, e.g., the Apriori algorithm by Agrawal & Srikant
- **Main applications:**
 - ✓ Studies on buying habits for targeted advertising
 - ✓ Product placement on supermarket shelves (*market-basket analysis*)
 - ✓ Variations in sales in case of lacking products

Data mining: clustering

- Given a population of objects as points in a multidimensional space, in which each dimension corresponds to a particular feature, clustering refers to grouping these objects together in a reduced number of sets (**clusters**) better characterizing the population. Clustering identifies the similarities between items.
- **Main applications:**
 - ✓ Segmenting customers into categories
 - ✓ The evaluation of clinical cases on the basis of symptoms
 - ✓ Epidemiological analysis



Data mining: decision trees

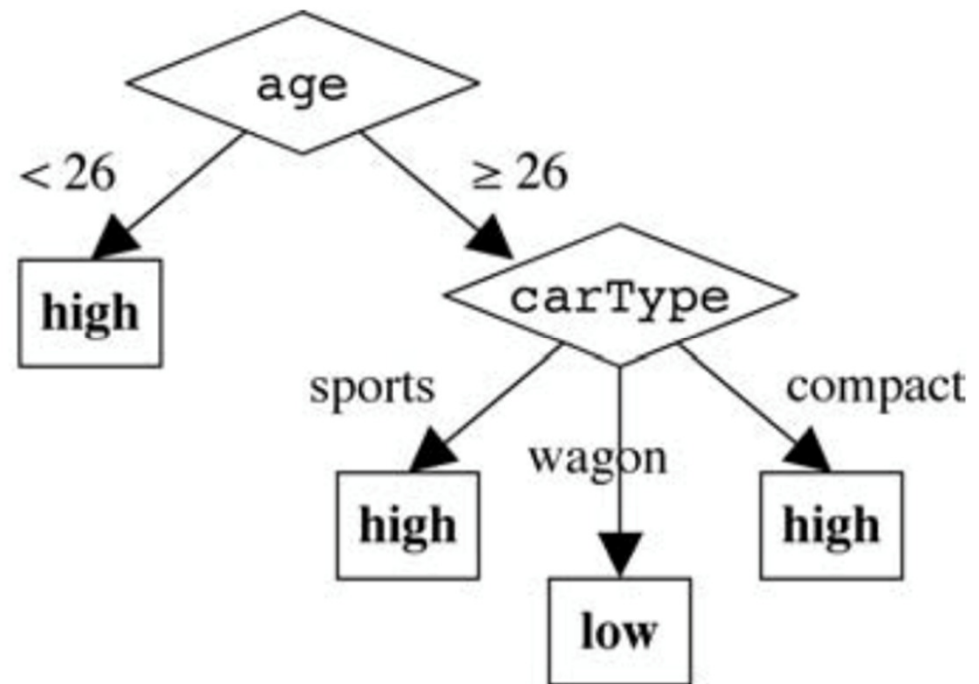
- Decision trees are a particular type of *classifiers*, i.e., algorithm that can create a descriptive profile for each category, based on the features of the data in the training set
- They are used to understand a specific phenomenon because they can prioritize the causes leading to a specific event.

Data mining: decision trees

■ Main Applications:

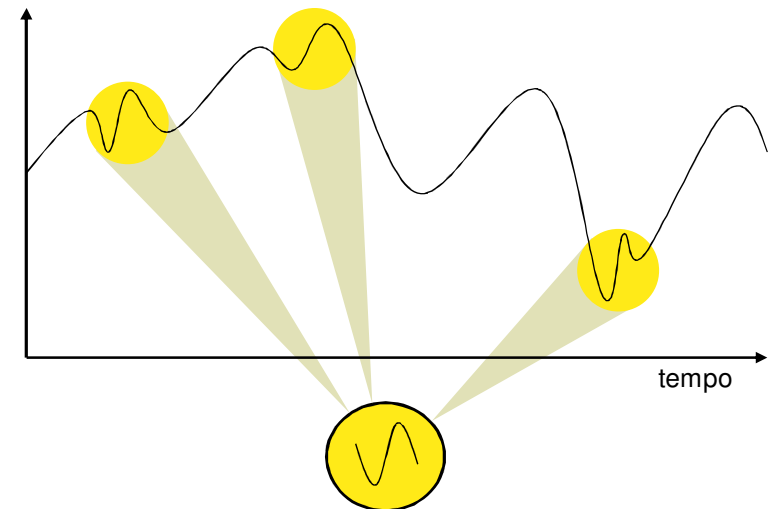
- ✓ Assessment of customer risk categories for companies granting mortgages, loans and insurances

age	carType	riskClass
40	wagon	low
65	sports	high
20	compact	high
25	sports	high
50	wagon	low



Data mining: time series

- The goal here is to identify recurring or atypical patterns in complex data sequences
- **Applications:**
 - ✓ Identifying associated growth schemata in shares on stock exchanges,
 - ✓ revealing anomalies in monitoring systems,
 - ✓ studying the relationships between different time series,
 - ✓ identifying companies with similar developmental models,
 - ✓ analyzing navigation routes on web sites

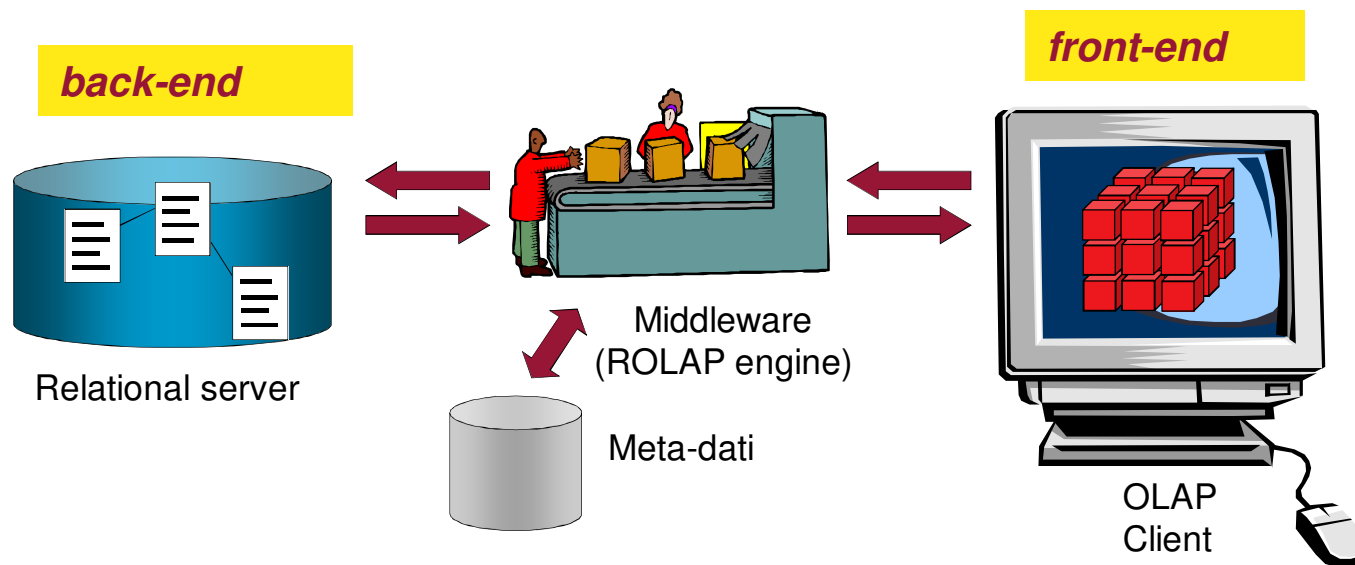


ROLAP

- Stands for *Relational OLAP*: an implementation based on relational DBMSs
- Motivated by the huge amount of literature written about the relational model, the broadly available corporate experience with relational database usage and management, and the top performance and flexibility standards of relational DBMSs. However,
 - ✓ the relational model does not include the concepts of dimension, measure, and hierarchy: you must create specific types of schemata that allow one to represent multidimensional schemata in terms of relational ones => **Star Schema**.
 - ✓ Performance problems (costly join operations over large tables) => **denormalization**.
- Since ROLAP uses a relational database, it requires more processing time and/or disk space to perform some of the tasks that multidimensional databases are designed for. However, ROLAP supports larger user groups and greater amounts of data.
- As is typical of relational databases, some queries are created and stored in advance (in materialized views)

ROLAP

- Data processing may take place within the database system, a middleware server, or the client.
- In a two-tiered architecture, the user submits an SQL query to the database and receives back the requested data.
- In a **three-tiered architecture**, the user submits a request for multidimensional analysis and the ROLAP engine converts the request to SQL for submission to the database. Then, the engine converts the resulting data from SQL to a multidimensional format before it is returned to the client for viewing.



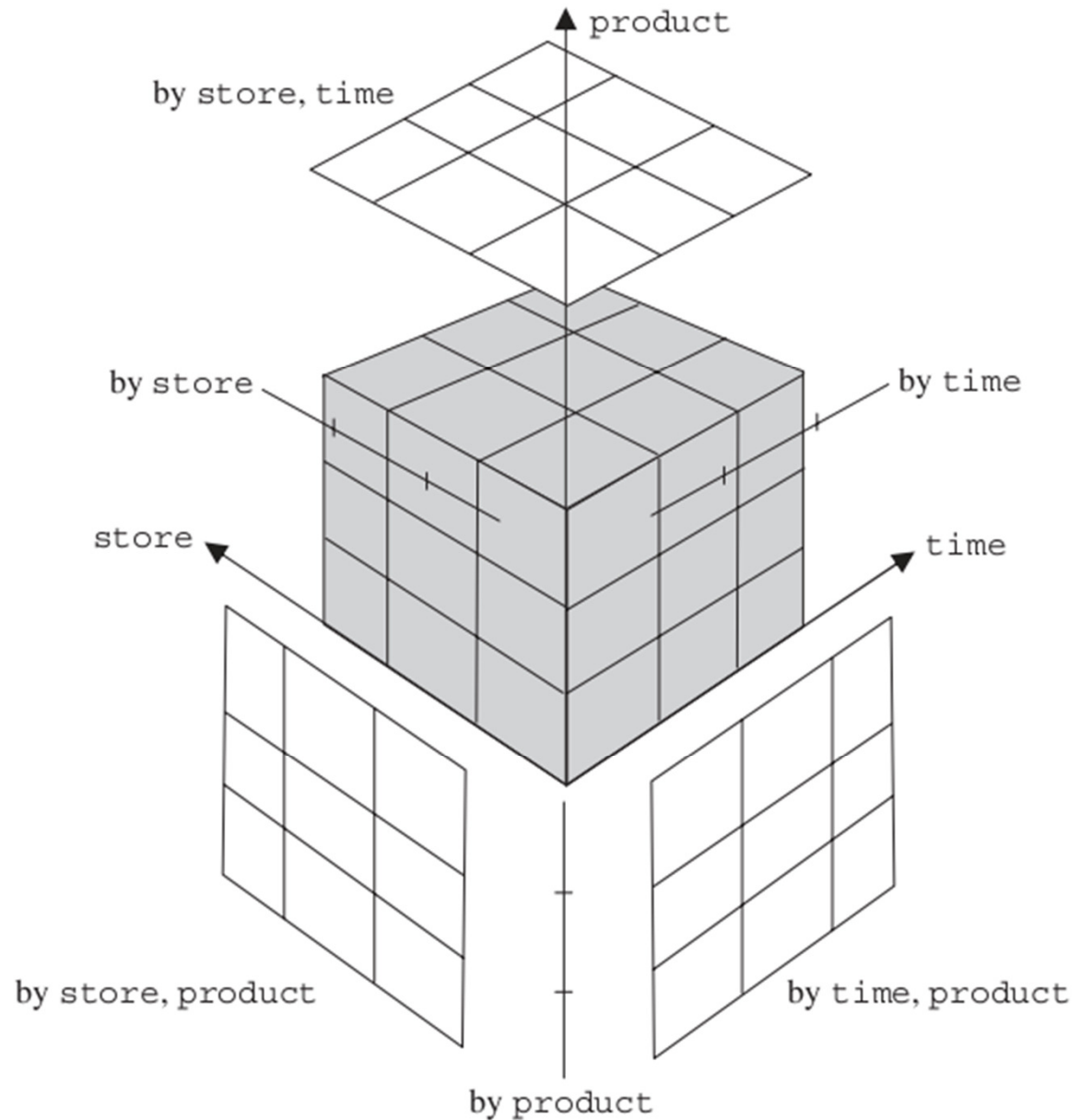
We will see more on ROLAP solutions later on in these slides

MOLAP

- It is based on an ad-hoc logical model, where **multidimensional data and operations can be natively represented**.
- Given a primary cube including all the primary events, i.e., the facts, its **data cube** consists of the primary cube surrounded by a collection of secondary coarse-grained cubes, which aggregate the primary cube along one or more dimensions and include secondary events.
- **All the secondary cubes do not need to be physically available because** you can calculate them by aggregating their primary cube along one. Nevertheless, **you may need to store some or all the secondary cubes to improve the query response time**.
- Data cubes thus act as a logical structure for the multidimensional model.
- **Data cubes are stored in arrays**. Each array element is associated with a set of coordinates on a value space. **Access to data is positional**, through the array coordinates.

MOLAP

The **data cube** for a simple SALE fact containing three dimensions (store, time, and product), without any hierarchy. Along with the primary cube (in gray), this figure shows three two-dimensional secondary cubes (by store, product; by store, time; by time, product). The 0-dimensional cube is not shown.



MOLAP

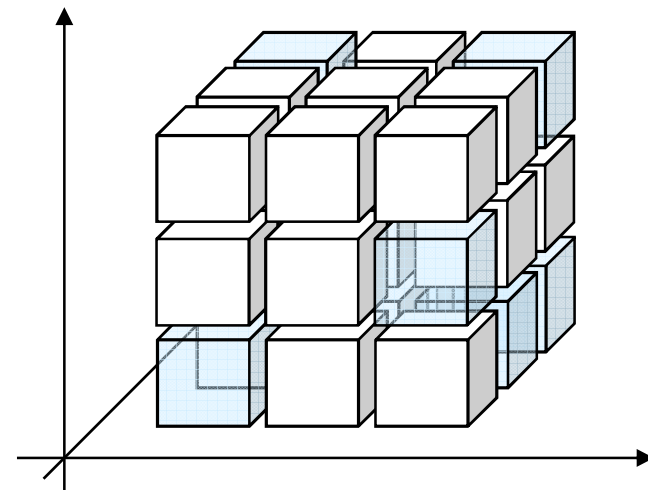
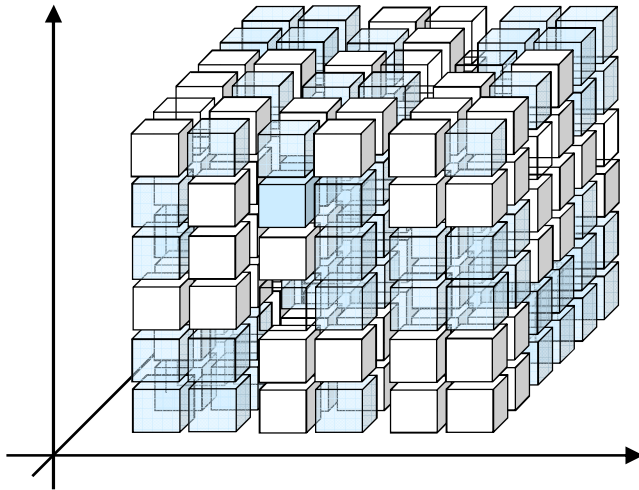
- ✓ The big advantage of the MOLAP approach with respect to the ROLAP one is that **multidimensional operations can be simply and naturally realized**, without the need of resorting to joins. For this reason, MOLAP system performance is excellent.
- ✓ However, **no standard for the logical model** at the basis of MOLAP approach currently **exists**, and the various implementations present very many differences one another: They simply share the usage of optimization techniques specifically designed for *sparsity management*.
- ✓ **No standard query language**: each vendor tends to propose a proprietary one (but *Microsoft Multidimensional Expressions* is imposing as standard de facto)
- ✓ Loading process is often slow.
- ✓ Some products may handle only up to ten dimensions with limited cardinality

MOLAP: The problem of sparsity

- In the multidimensional model, a set of coordinates corresponds to a possible event even if this does not actually take place.
- Typically, the number of occurred events is far less than possible events (around 20%).
- Keeping track of events that did not occur leads to a waste of resources and reduces the performance of the system.
 - ✓ ROLAP: can store only useful cells (occurred events).
 - ✓ MOLAP: calls for complex techniques to minimize the space required to keep track of not occurred events.

MOLAP: The problem of sparsity

- We can group cubes into “chunks”, and classify these as sparse or dense chunks. MOLAP systems try to materialize mainly dense chunks.



- Various strategies are possible (**Hybrid OLAP**):
 - Store dense chunks in MOLAP mode and sparse chunks in ROLAP mode.
 - Store primary cubes in ROLAP mode and secondary cubes in MOLAP mode (aggregation reduces sparsity).
 - Store frequently accessed data in MOLAP mode and the remaining data in ROLAP mode.

Need of a Design Methodology

- Many organizations lack the experience and skills required to meet the challenges involved in data warehousing projects.
- The lack of a **methodological approach** prevents data warehousing projects from being carried out successfully.
- Methodological approaches are created by closely studying similar experiences and minimizing the risks for failure by basing new approaches on a **constructive analysis of previous mistakes**.

Risk Factors

- **Risks related to project management:** Managers are often reluctant to share information one another. Also, managers cannot often convincingly report estimates of costs and benefits
- **Risks related to technology:** chosen architectures have to be scalable in terms of users and data; and expandible to implement new technology solutions; technical skills of implementers have to be adequate
- **Risks related to data and desing:** source instability or unreliability, as well as non-properly specified requirements may lead to low quality results. Inadequacy of initially developed prototypes may compromise the reputation of the DW project
- **Risks related to organization:** Inability to involve end users; difficulties in radically changing business culture; users's inability to take advantage of the results

A top-down approach

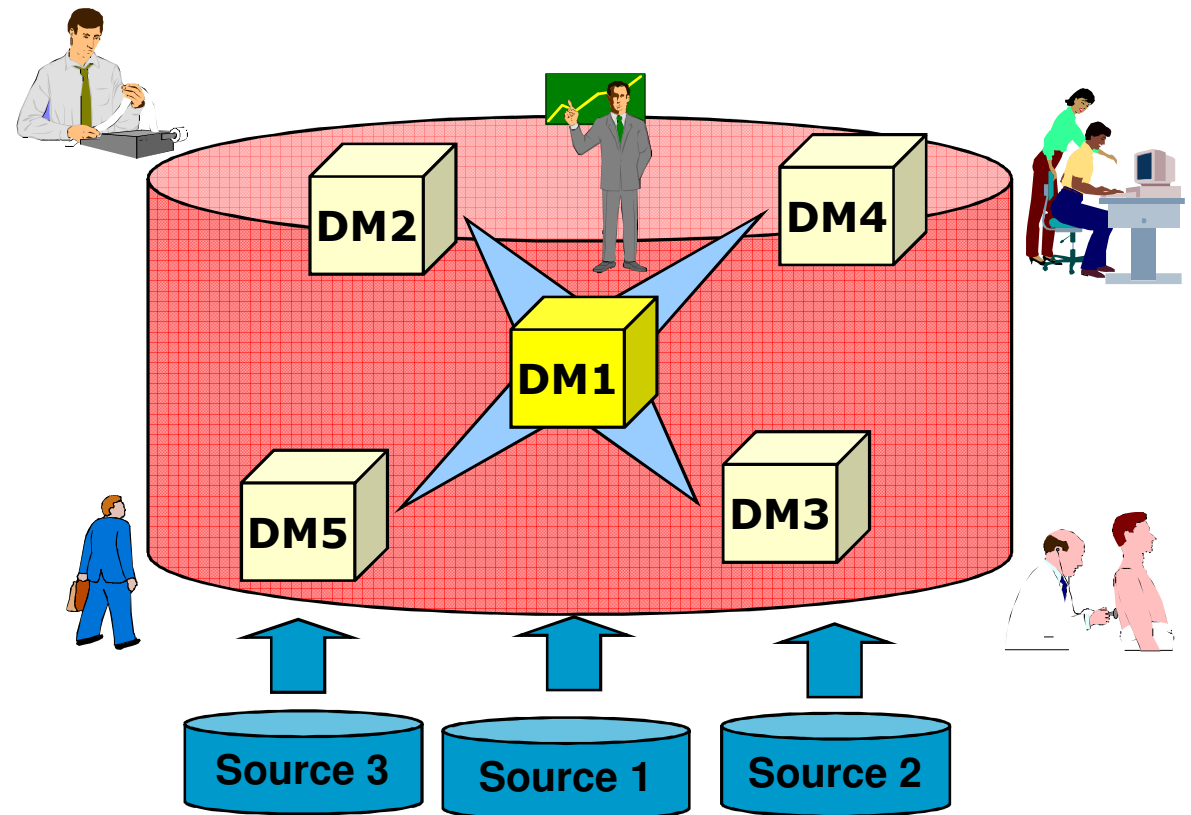
- Analyze global business needs, plan how to develop a data warehouse, design it, and implement it *as a whole*.
 - 👍 it will reach **excellent results** because it is based on a global picture of the goal to achieve, and in principle it ensures consistent, well integrated data warehouses.
 - 👎 high-cost estimates with **long-term implementations** discourage company managers from embarking on these kind of projects.
 - 👎 analyzing and **bringing together all relevant sources** is a very **difficult** task.
 - 👎 it is extremely difficult to **forecast the specific needs of every department** involved in a project, which can result in the analysis process coming to a standstill.
 - 👎 since **no prototype** is going to be delivered in the short term, users cannot check for this project to be useful, so they lose trust and interest in it

Bottom-up approach

- In a bottom-up approach, data warehouses are incrementally built and several data marts are iteratively created. Each data mart is based on a set of facts that are linked to a specific company department and that can be interesting for a user subgroup
 - 👍 Determines **concrete results in the short term**, especially if coupled with quick prototyping
 - 👍 **It does not require high financial investments** (at least, not from the very beginning of the project)
 - 👍 It allows to study **only the issues related to the data mart at hand**
 - 👍 Provides **immediate feedback to senior management** on the effectiveness of the system under construction
 - 👍 Attention on the project is steadily high
 - 👎 **Determines a partial view of the domain of interest!**

Which data mart to realize first?

- ✓ it should play a very strategic role in the company
- ✓ It should be a reference point for the whole data warehouse
- ✓ It should exploit consistent data already made available

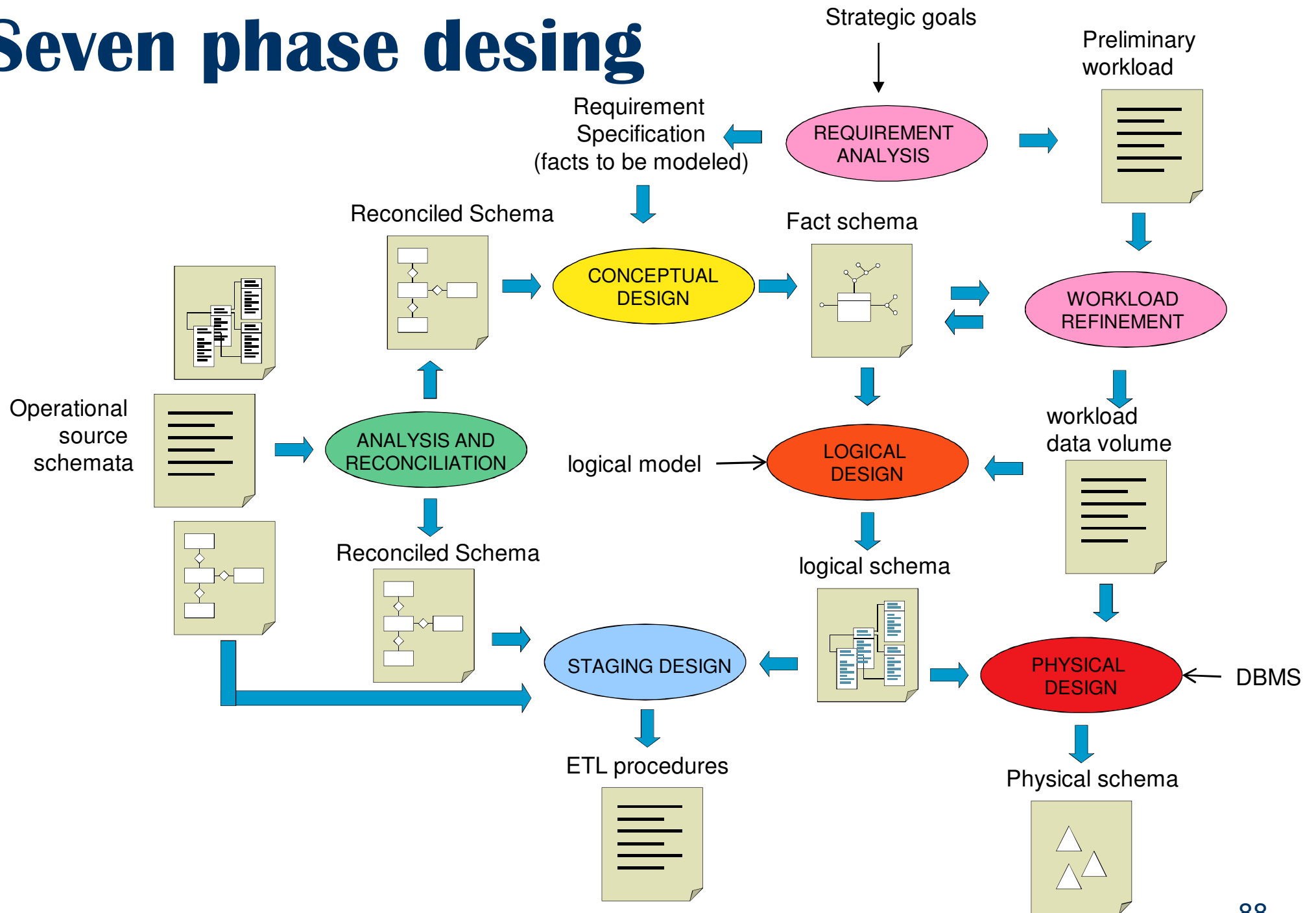


In an hospital DW, DM1 might be the data mart for hospital task collection and monitoring

Data Mart Design

Phase	Input	Output	People Involved
Analysis and reconciliation of data sources	Operational source schemata	Reconciled schema	Designer, data processing center staff
Requirement analysis	Strategic goals	Requirement specifications, preliminary workload	Designer, end users
Conceptual design	Reconciled schema, requirement specification	Fact schemata	Designer, end users
Workload refinement, validation of conceptual schemata	Fact schemata, preliminary workload	Workload, data volume, validated fact schemata	Designer, end users
Logical design	Fact schemata, target logical model, workload	Logical data mart schema	Designer
Data-staging design	Source schemata, reconciled schema, logical data mart schema	ETL procedures	Designer, database administrators
Physical design	Logical data mart schema, target DBMS, workload	Physical data mart schema	Designer

Seven phase desing



Facts

- **Facts** are concepts on which data mart end users base their decision-making process. Each fact describes a category of events taking place in enterprise. They have to be identified in the *Requirement Analysis phase*.
 - ✓ Designers should also have clear ideas on **fact dimensions**. Focusing on the dimensions of a fact leads to the definition of **fact granularity**, i.e., the highest detail level for data to be represented in a data mart. Selecting a granularity level for fact representation is the result of a delicate compromise between two opposite needs: the need for maximum usage flexibility, which implies the same granularity level as operational sources, and the need for top performance, which implies a concise synthesis of data.
 - ✓ Every fact needs **a historical interval** to be defined -- that is, a period of time covered by the events stored in a data mart.

Typical facts of different application fields

Application Field	Data Mart	Facts
Business, manufacturing	Supplies	Purchases, stock inventory, distribution
	Production	Packaging, inventory, delivery, manufacturing
	Demand management	Sales, invoices, orders, shipments, complaints
	Marketing	Promotions, customer retention, advertising campaigns
Finance	Banks	Checking accounts, bank transfers, mortgage loans, loans
	Investments	Securities, stock exchange transactions
	Services	Credit cards, bill payment through standing orders
Health service	Division	Admissions, discharges, transfers, surgical operations, diagnosis, prescriptions
	Accident & emergency	Admissions, tests, discharges
	Epidemiology	Diseases, outbreaks, treatments, vaccinations
Transportation	Goods	Demand, supply, transport
	Passengers	Demand, supply, transport
	Maintenance	Operations
Telecommunications	Traffic management	Network traffic, calls
	Customer relationship management	Customer retention, complaints, services

Other requirements

Need to be analyzed and specified:

- Requirements for logical and physical design (e.g., available space to optimize data mart performances).
- Requirements for data staging design (frequency of data staging)
- Requirements for data quality (especially end-users expectation on it)
- Data warehouse architecture (type of architecture, number of layers, independent or dependent data marts, and reconciled layer materialization)
- Requirements on data analysis applications to adopt (on the basis of the types of queries end users expect to issue to their data marts and the analytic reports IT staff members are usually required to draw up)
- Startup schedule
- Training schedule

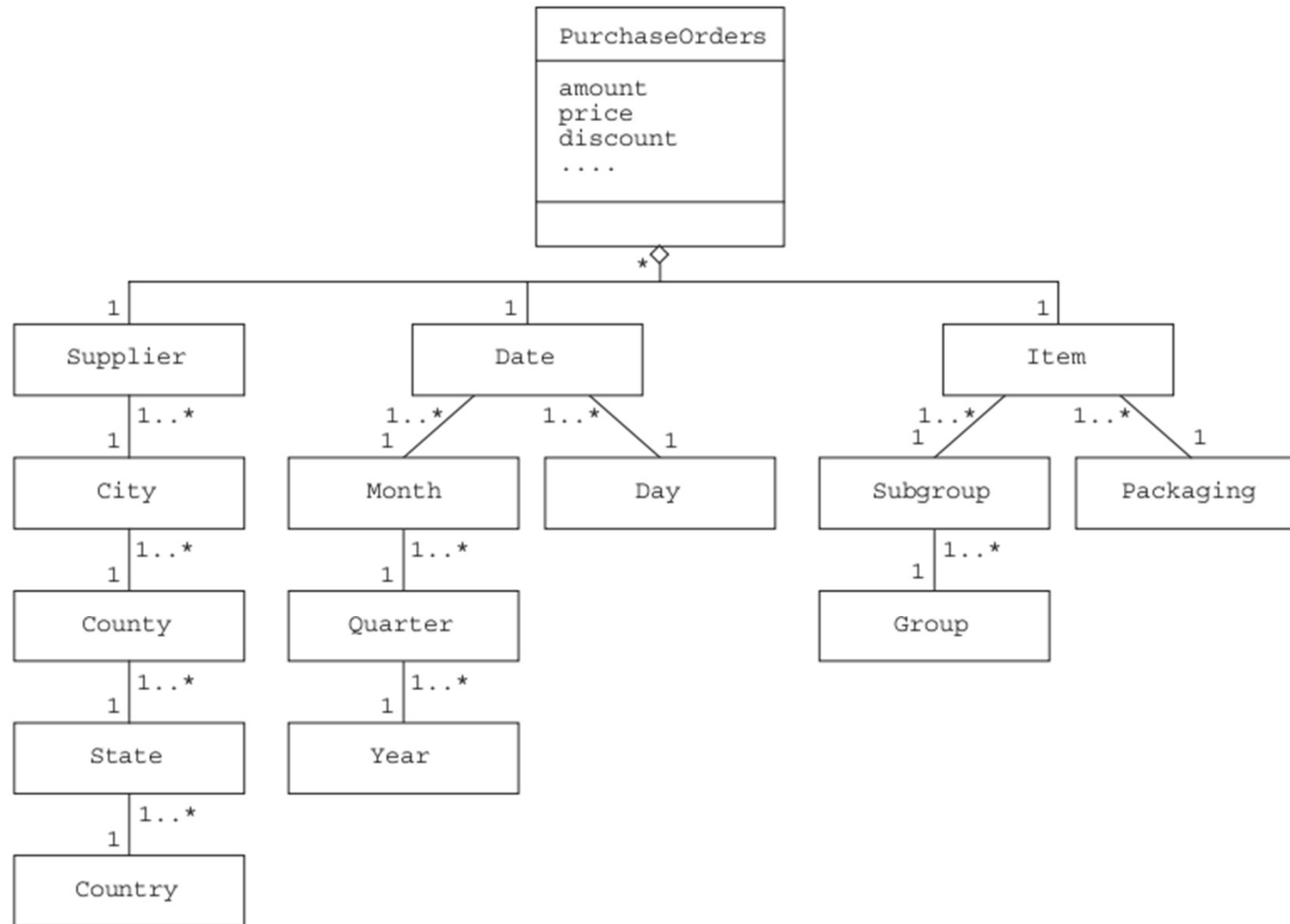
Conceptual Modeling for DWs

- It is well known that an accurate conceptual design is the fundamental requirement for the construction of a well-documented database that completely meets the requirements.
- While it is universally acknowledged that a DW relies (at the logical level) on the multidimensional model, there is **no agreement on the methodology for conceptual modeling in DWs.**
- The Entity/Relationship (**ER**) model is widely used in organizations for the documentation of information systems based on the relational model (even though not always in a formal and completely understood way), it **cannot be used to model a DW.**

Conceptual Modeling for DWs

- Indeed, ER model is oriented to queries that follow associations among data rather than summarize the data.
- The basic ER model is not able to accurately highlight the distinctive features of the multidimensional model (e.g, it does not directly allow for easy modeling of hierarchies on dimensions)
- It is not very efficient from the notational viewpoint.
- UML class diagrams suffer of the same drawbacks.

UML-based Conceptual Modeling for DW: an example [Lujan-ora-2006]



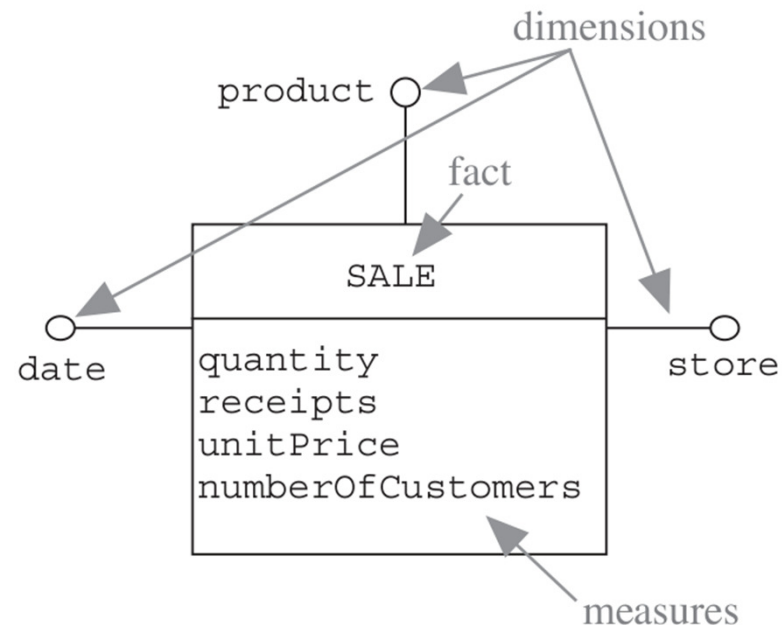
A UML class diagram for purchase order analysis

The Dimensional Fact Model

- DFM is a graphical conceptual model for data marts, designed for:
 - ✓ Effectively support the conceptual design.
 - ✓ create an environment in which user queries may be formulated intuitively.
 - ✓ make communication possible between designers and end users to refine requirement specifications.
 - ✓ build a stable platform from which starting the logical design (*independent from the target logical model*).
 - ✓ provide clear and expressive design documentation.
- The conceptual representation generated by the DFM consists of a set of **fact schemata**. Fact schemata basically model facts, measures, dimensions, and hierarchies.

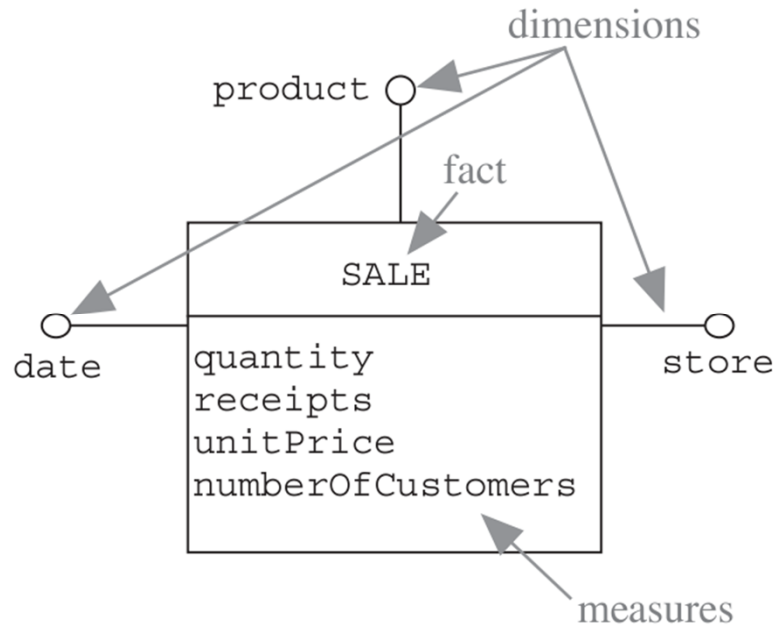
DFM: basic constructs

- A **fact** is a concept relevant to decision-making processes. It typically models a set of events taking place within a company (e.g., sales, shipments, purchases, and complaints, etc.). It is essential that a fact have dynamic properties or evolve in some way over time
- A **measure** is a numerical property of a fact and describes a quantitative fact aspect that is relevant to analysis (e.g., each sale is measured by the number of units sold, the unit price, and the total receipts)
- A **dimension** is a fact property with a finite domain and describes an analysis coordinate of the fact (typical dimensions for the sales fact are products, stores, and dates)

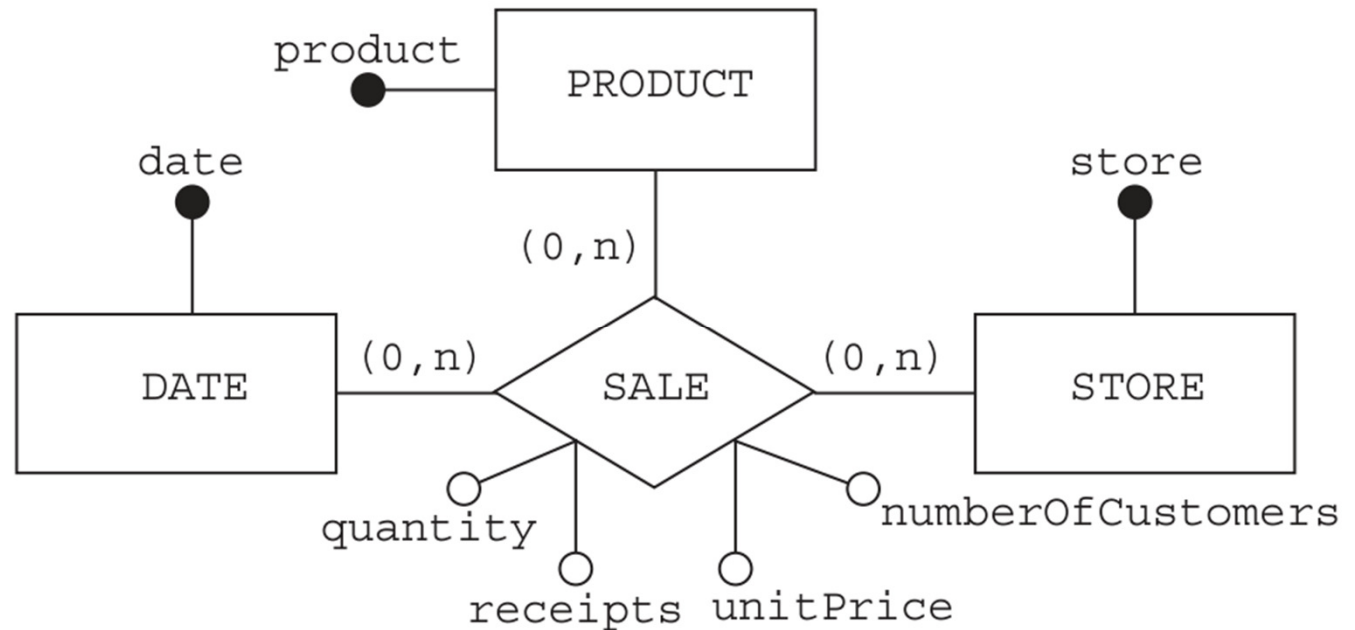


Note: A fact can also have no measures, as in the case when you might be interested in recording only the occurrence of an event

DFM: basic constructs

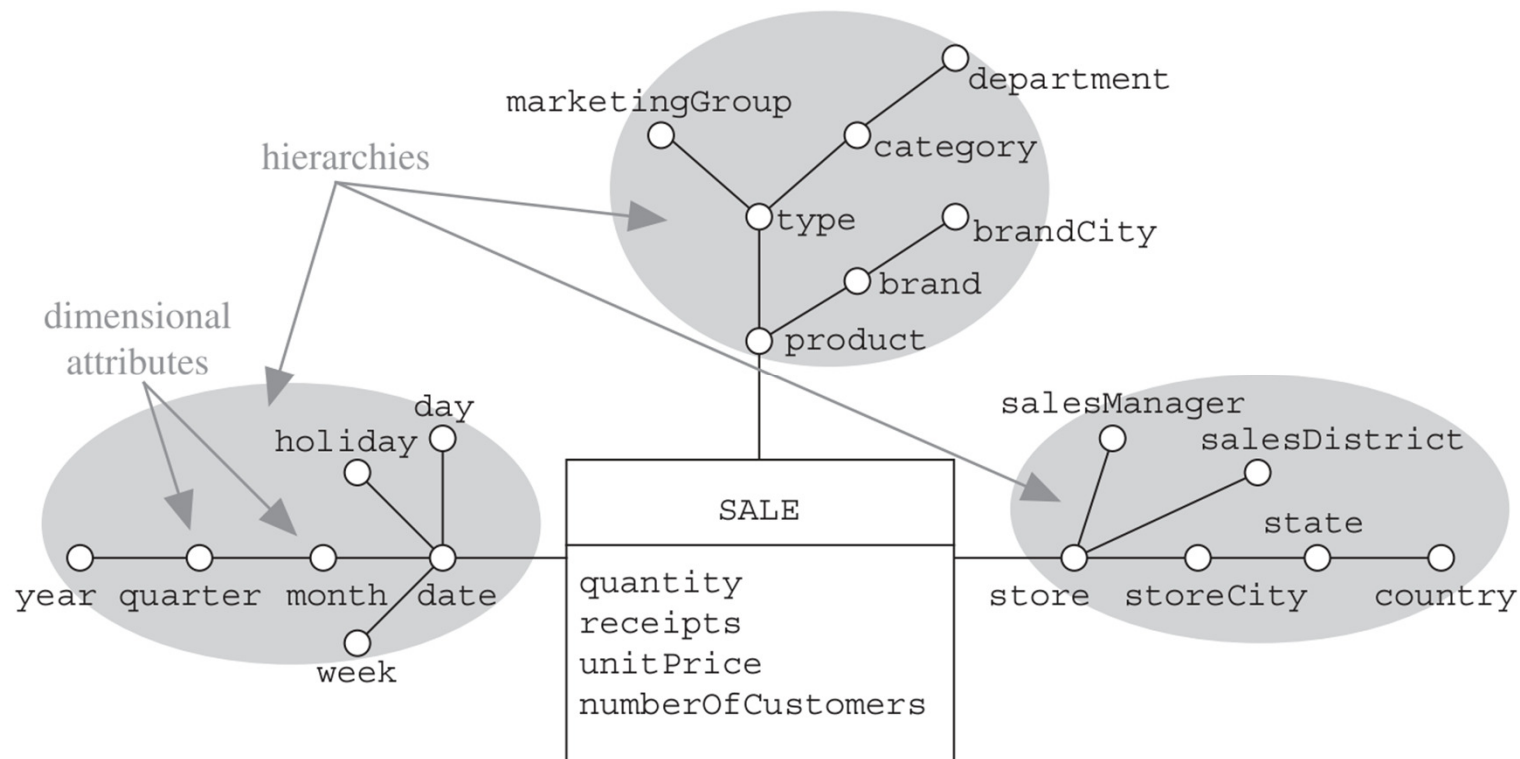


A fact expresses a many-to-many association between dimensions



DFM: basic constructs

- The general term **dimensional attributes** stands for the dimensions and other possible attributes, always with discrete values, that describe them (for example, a *product* dimension is described by its *type*, the *category* it belongs to, its *brand*, the *department* in which it is sold)
- A **hierarchy** is a directed tree whose nodes are dimensional attributes and whose arcs model many-to-one associations between dimensional attributes. It includes a dimension, positioned at the trees root, and all of the dimensional attributes that describe it.



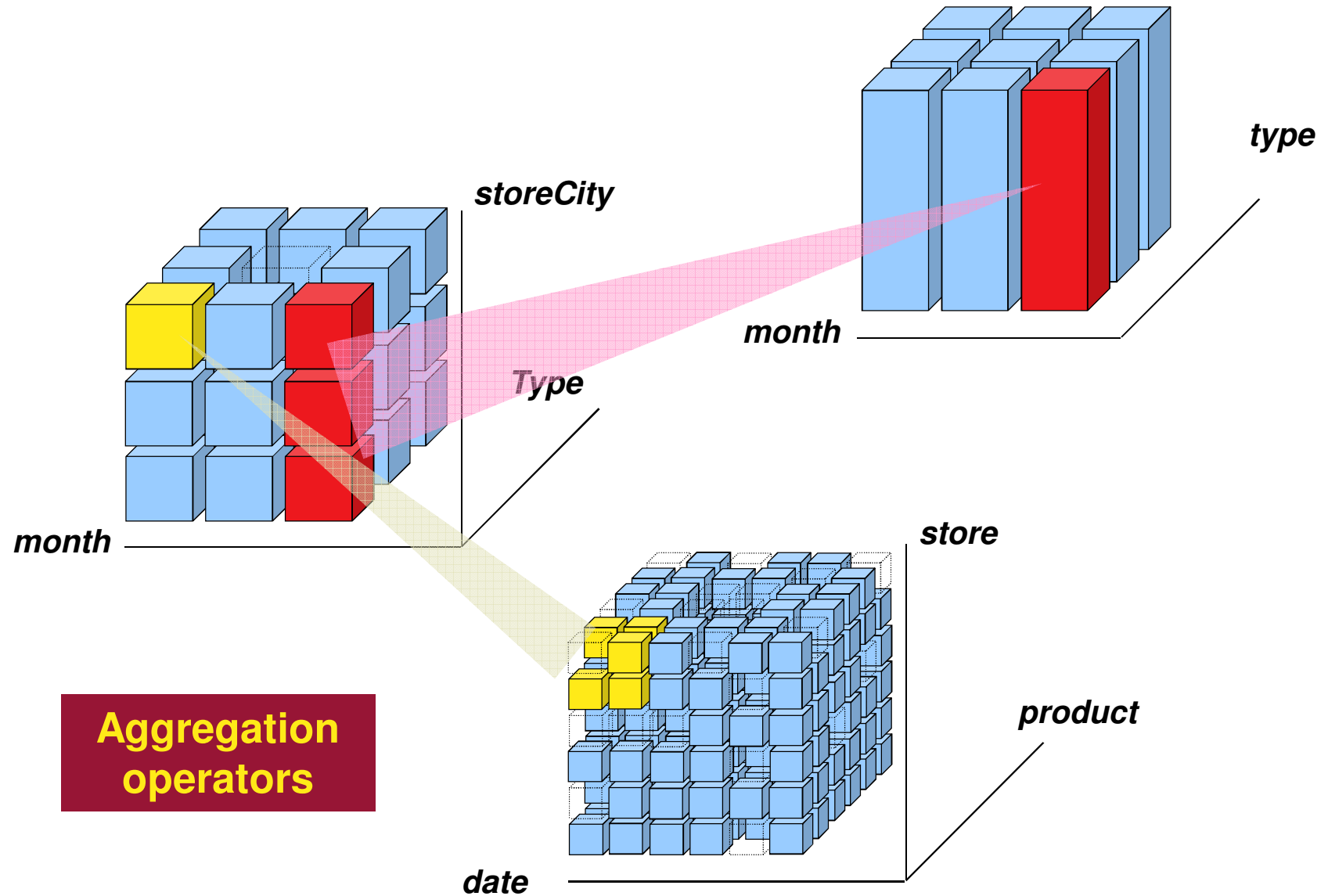
“Naming conventions”

- All dimensional attributes in a fact schema must have different names.
- Similar names can be differentiated by qualifying them with the name of the dimensional attribute that comes before them in hierarchies.
 - ✓ For instance, *BrandCity* and *storeCity*.
- Names of dimensional attributes should not explicitly refer to the fact they belong to
 - ✓ For instance, *shippedProduct* and *shipmentDate* have to be avoided
- Attributes with the same meaning in different fact schemata must have the same name.

Events and Aggregations

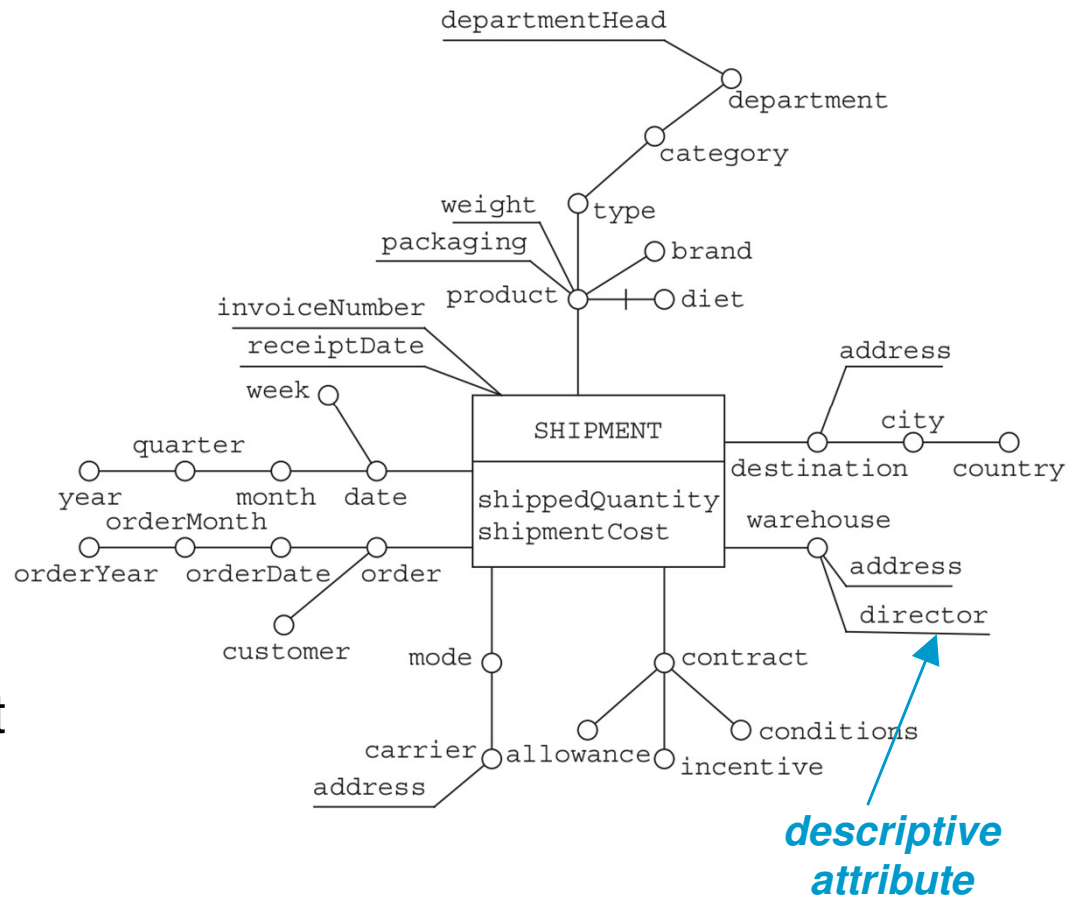
- A **primary event** is a particular occurrence of a fact, identified by one n-ple made up of a value for each dimension. A value for each measure is associated with each primary event.
 - ✓ In reference to sales, for example, a possible primary event records that 10 packages of Shiny detergent were sold for total sales of \$25 on 10/10/2008 in the SmartMart store.
- Given some dimensional attributes (*pattern*), each n-ple of their values identifies a **secondary event** which aggregates all corresponding primary events. To any secondary event a value is associated for each measure. Such a value sums up all the values of the same measure in the corresponding primary events.
 - ✓ Thus, hierarchies define the way you can aggregate primary events and effectively select them for decision-making process; while the dimension in which a hierarchy takes root defines its finest aggregation granularity, the other dimensional attributes correspond to a gradually increasing granularity.

Events and Aggregation



DFM: advanced constructs

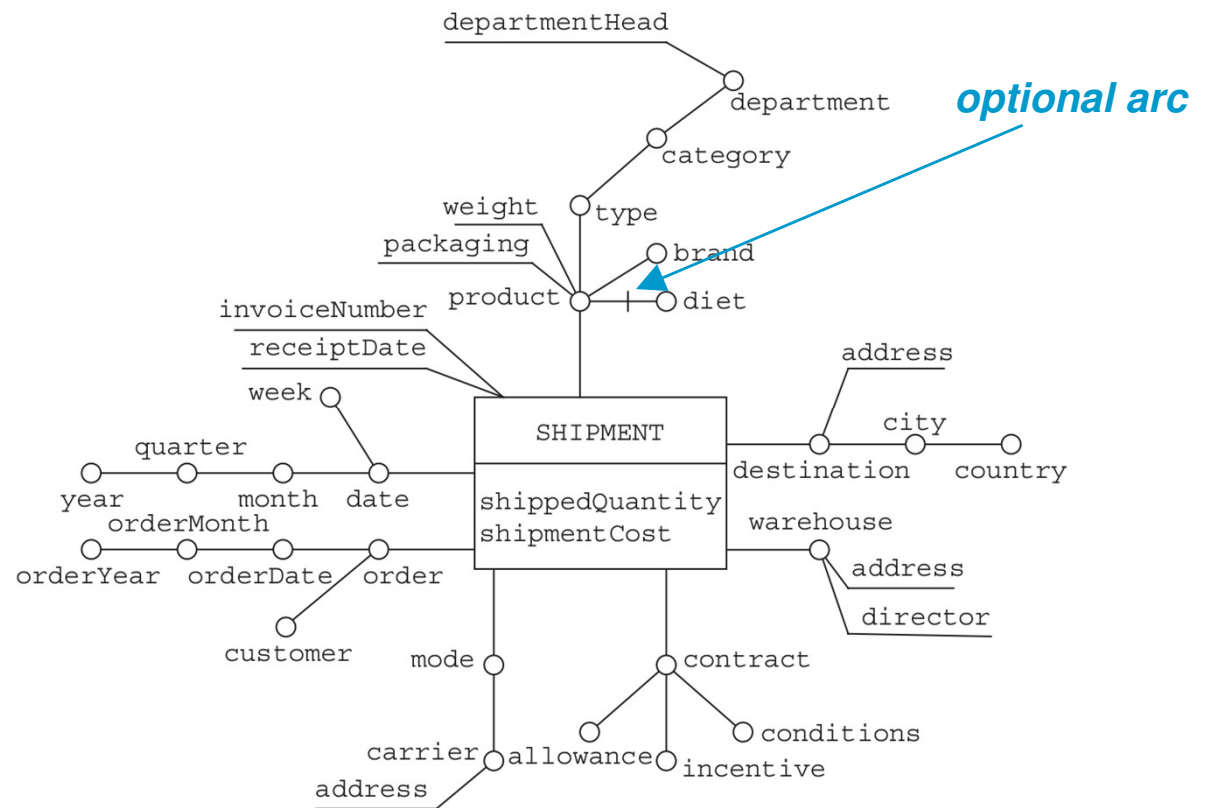
- A *descriptive attribute* is functionally determined by a dimensional attribute of a hierarchy and specifies a property of this dimensional attribute. Descriptive attributes often are tied to dimensional attributes by one-to-one associations and do not actually add useful levels of aggregation. It can also be directly connected to a fact if it describes primary events, but it is neither possible nor interesting to use it to identify single events or to make calculations



Note: In an ER representation of such model, you can model descriptive attributes as attributes of the entities corresponding to dimensional attributes or the relationship modeling the fact

DFM: advanced constructs

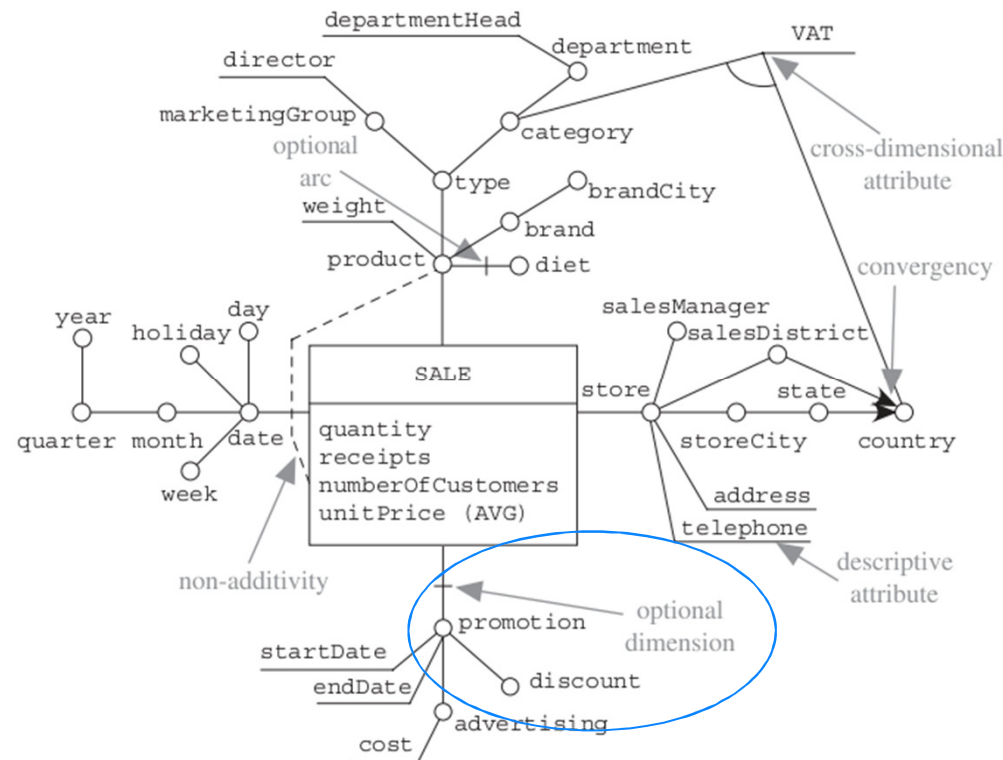
- Some arcs can be marked as *optional*.
- If the optionality is associated to a dimensional attribute *A* (as *diet* in the figure), this means that *A* and all possible descendents may be undefined for one or more instances of the fact to which the arc is (indirectly) associated



Note: In an ER representation of such model, the minimum cardinality of the association from *PRODUCT* to *DIET* is 0 rather than 1

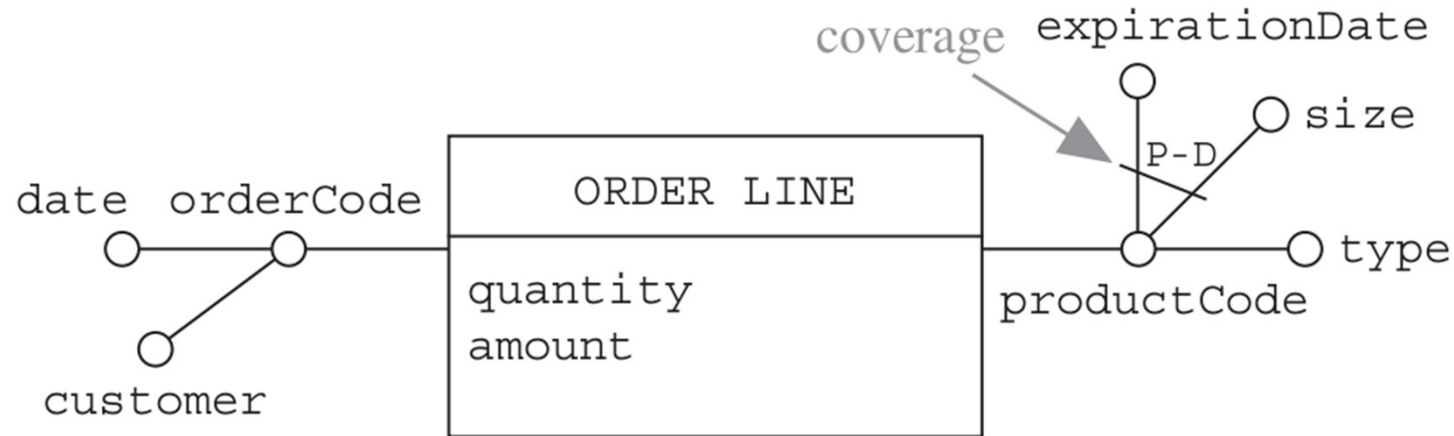
DFM: advanced constructs

- If the **optionality** is associated to a dimension d , this means that some primary events exist that are identified only by the other dimensions



Note: : In an ER representation of such model, the SALE has to be reified and identified by its participation to the relationships towards DATE, PRODUCT, and STORE. Also, it has a (0,n) participation to a relationship towards PROMOTION

DFM: advanced constructs



A **coverage** can be specified on two or more arcs that exit from the same dimensional attribute *A*. It can be:

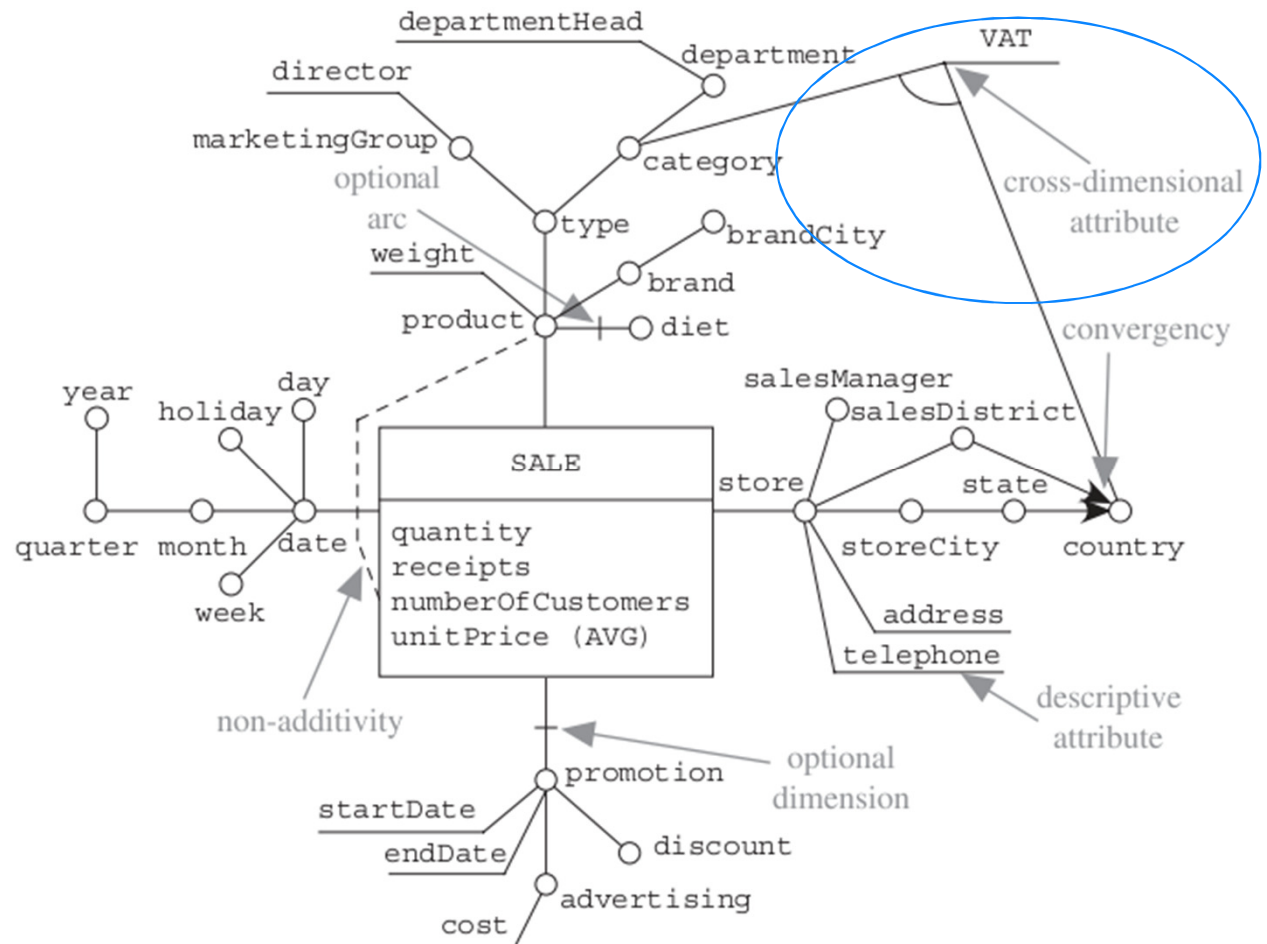
- **total** (T): every value of *A* is associated to at least a value for one of the child attributes of *A*
- **partial** (P): there are values for *A* for which all the children are undefined

and

- **Disjoint** (D): a value for at most one of the children is associated to *A*
- **Overlapping** (O): values of *A* exist linking to values of two or more children

DFM: advanced constructs

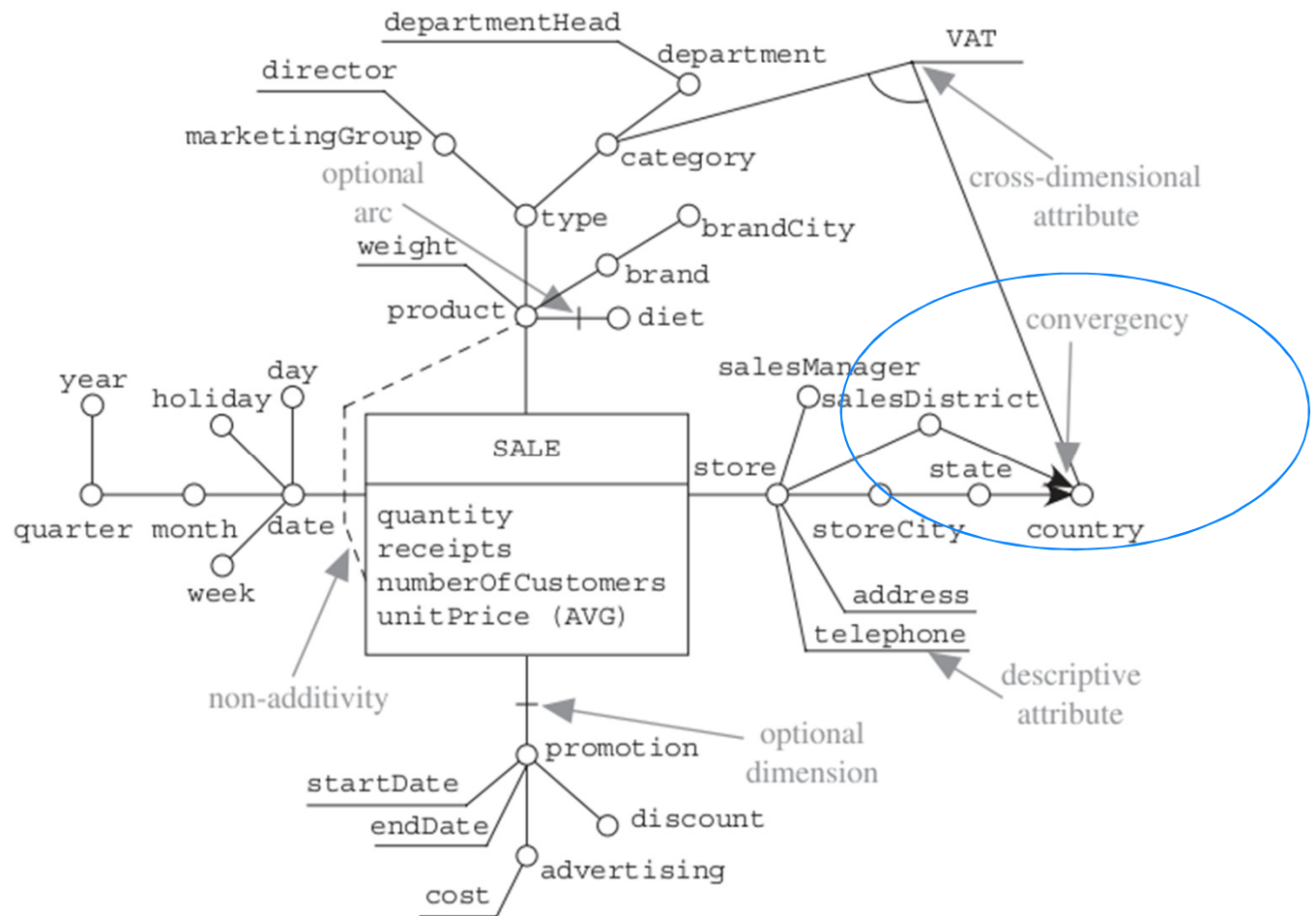
A **cross-dimensional attribute** is a *descriptive attribute* whose value is defined by the combination of two or more dimensional attributes. For example, a product Value Added Tax (VAT) depends both on the product category and on the country where the product is sold.



Note: : In an ER representation of such model, VAT is an attribute of a relationship between CATEGORY and COUNTRY, to which both such entities participate with cardinality (1,n)

DFM: advanced constructs

convergence is an integrity constraint. For example, the country of the state of the city of a store, has to be the same of the saleDistrict of the store.

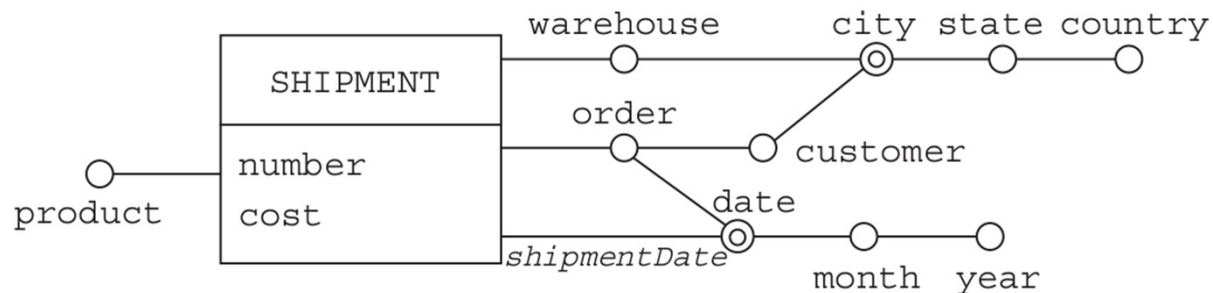
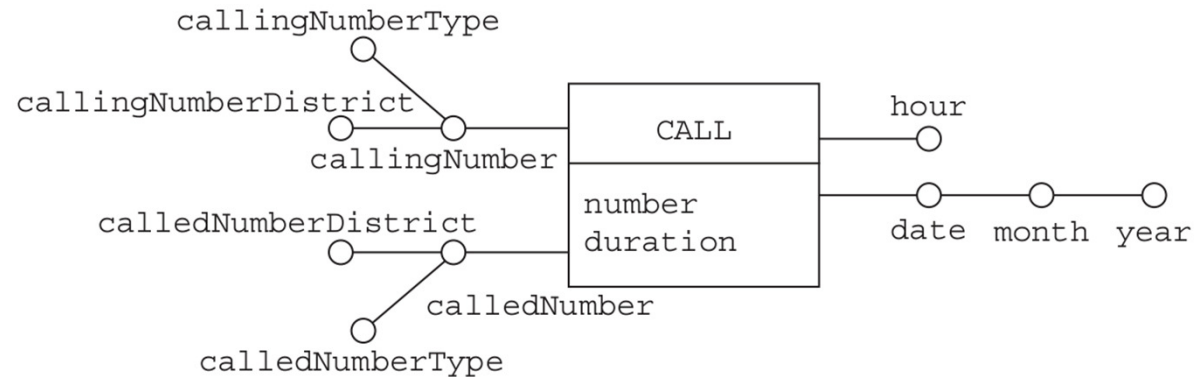
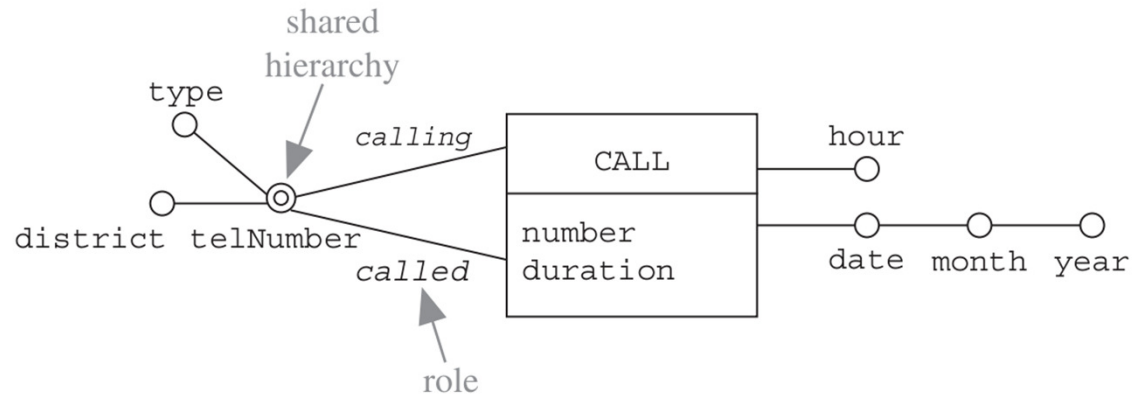


Note: In an ER representation of such model, convergence has to be specified through an external constraint

DFM: advanced constructs

Shared hierarchy:

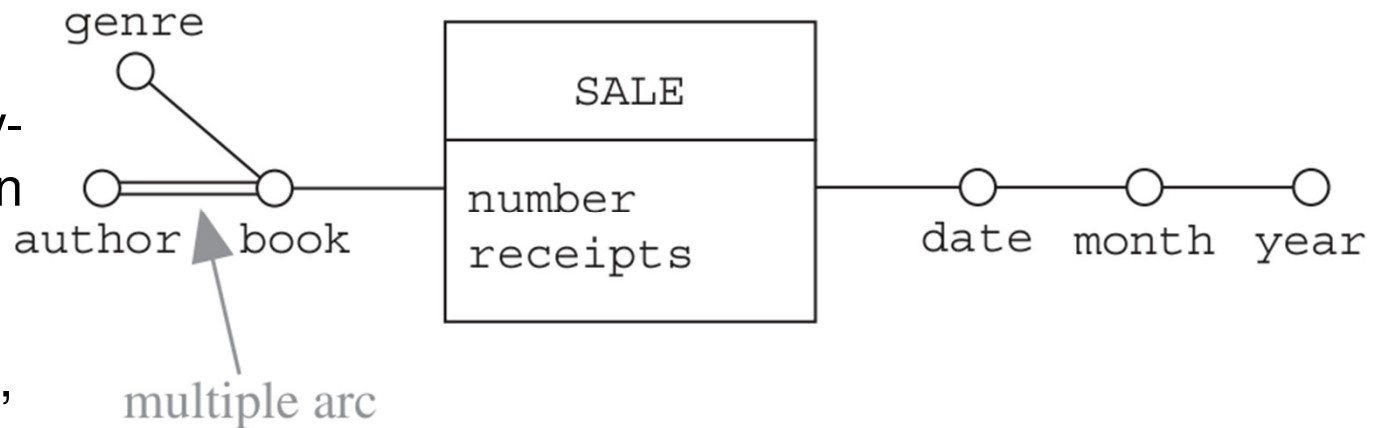
Avoids to replicate two or more times *entire portions* of hierarchies in a fact schema. If the sharing is on the overall hierarchy, a role is specified through an arc label



DFM: advanced constructs

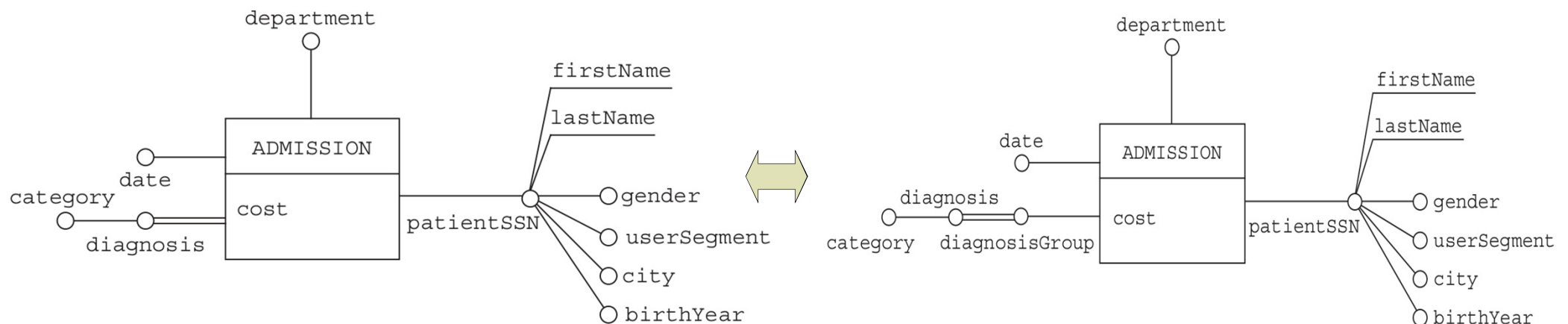
Multiple arc:

Allows for modeling many-to-many relations between dimensional attributes (instead of many-to-one relations). In the example, many different authors can write many different books.



Multiple arc entering a dimension:

A fact is identified by a tuple containing a value for each dimension. In the example, the value for diagnosis is, in fact, a “group of diagnosis”



Note: In an ER representation of such model, the relationship corresponding to a multiple arc is a many-to-many relationship.

Additivity

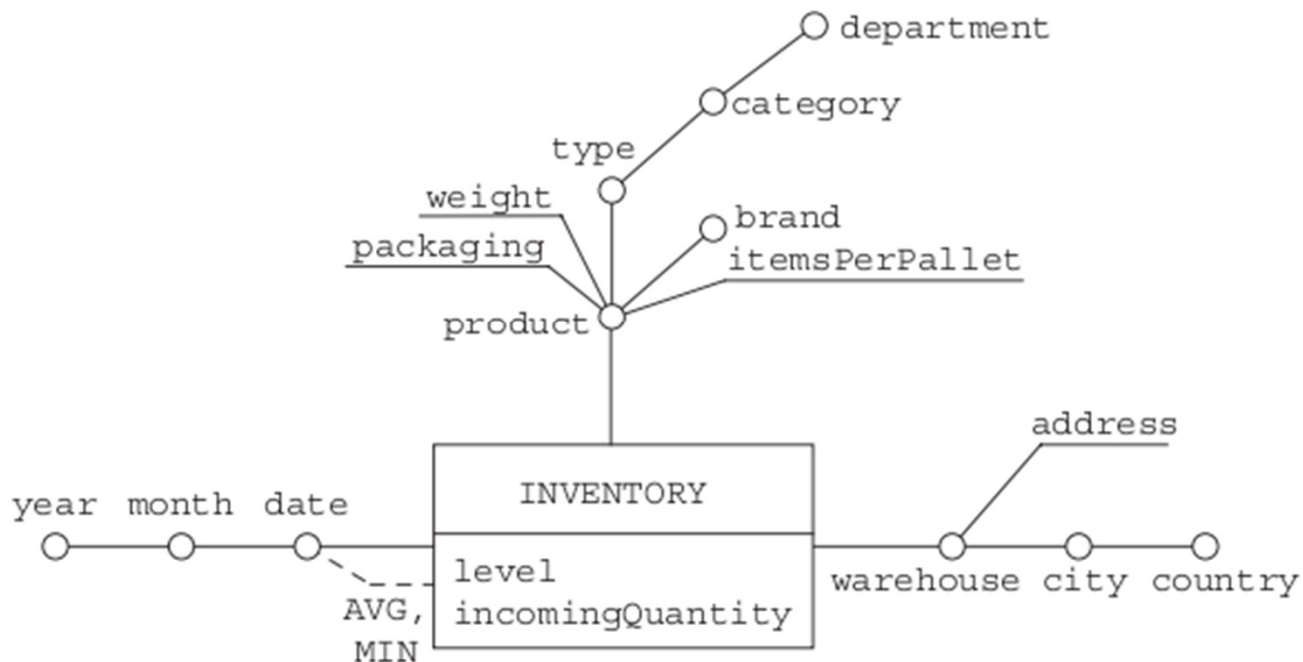
- Aggregation requires the definition of a suitable operator to compose the measure values that mark primary events into values to be assigned to secondary events.
- From this viewpoint, measures can be classified into three categories:
 - ✓ **Flow Measures:** refer to a timeframe, at the end of which they are evaluated cumulatively for such timeframe (the number of products sold in a day, monthly receipts, and yearly number of births).
 - ✓ **Level Measures:** are evaluated at particular times (the number of products in inventory, the number of inhabitants in a city).
 - ✓ **Unit Measures:** are evaluated at particular times but are expressed in relative terms (product unit price, discount percentage, currency exchange).

	Temporal Hierarchies	Nontemporal Hierarchies
Flow Measures	SUM, AVG, MIN, MAX	SUM, AVG, MIN, MAX
Level Measures	AVG, MIN, MAX	SUM, AVG, MIN, MAX
Unit Measures	AVG, MIN, MAX	AVG, MIN, MAX

Valid Aggregation Operators for Three Types of Measures

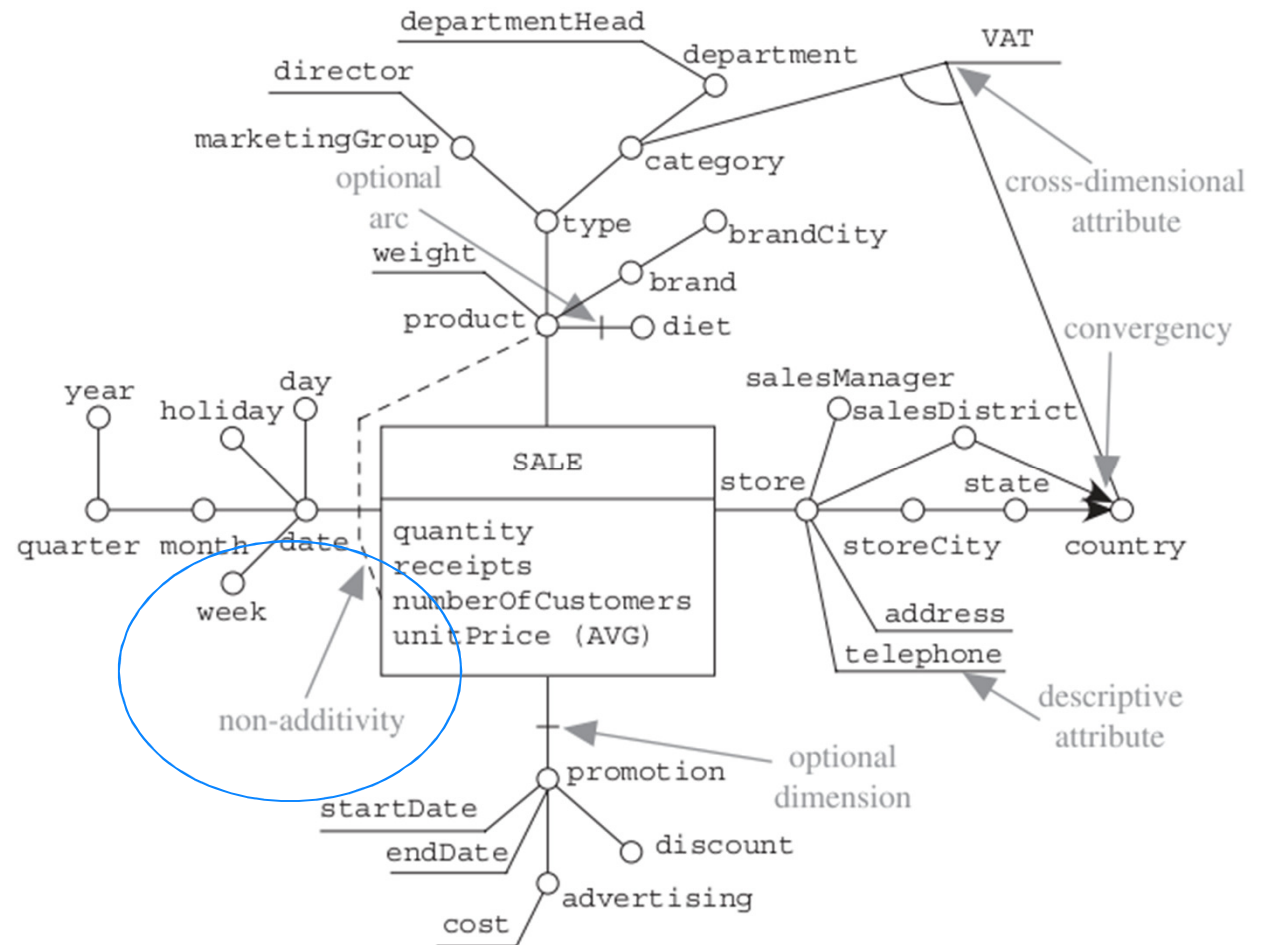
Additivity

- A measure is called **additive** along a dimension when you can use the SUM operator to aggregate its values along the dimension hierarchy. If this is not the case, it is called **non-additive**.
- A measure is connected to the dimensions along which it is non-additive via a dashed line labeled with the usable aggregation operators.
- Example: level is non-additive along date dimension, but it is additive along the other dimensions.



Additivity

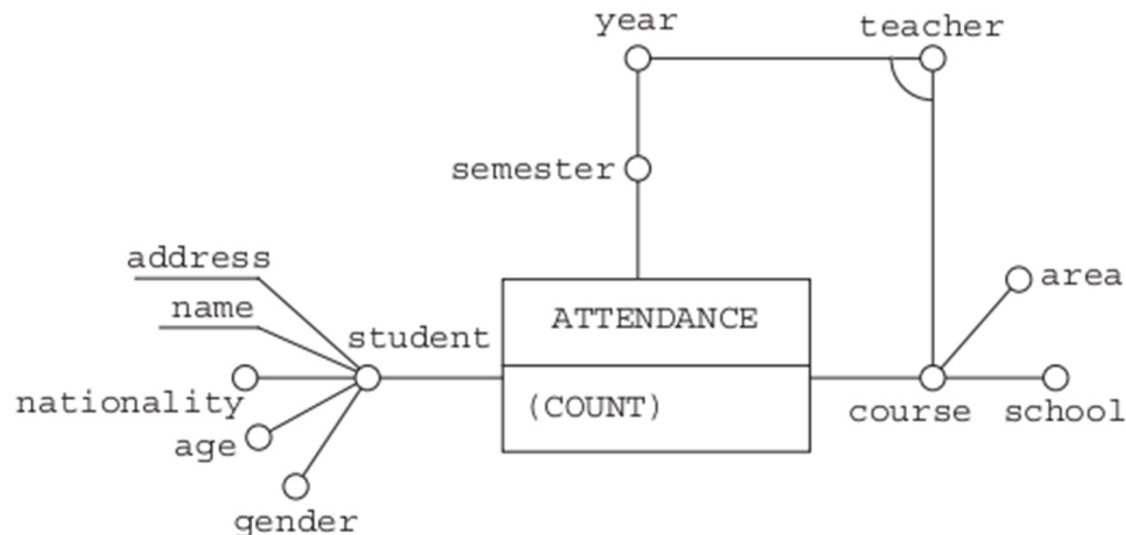
In the sales fact schema, the numberOfCustomers measure (i.e., estimated through the number of receipts issued for a product in a certain date in a certain store) is non-additive along the product dimension: Indeed, a single receipt refers to several products (possibly of different types). In this case it is also considered *non-aggregable* at all (missing labels)



If a measure shows the same type of additivity on all of the dimensions, you can mark the aggregation operator on its: unitPrice is a non-additive unit measure.

Empty fact schemata

- A fact schema is said to be **empty** if it does not have any measures
 - ✓ If this is the case, primary events only record the occurrence of events in an application domain

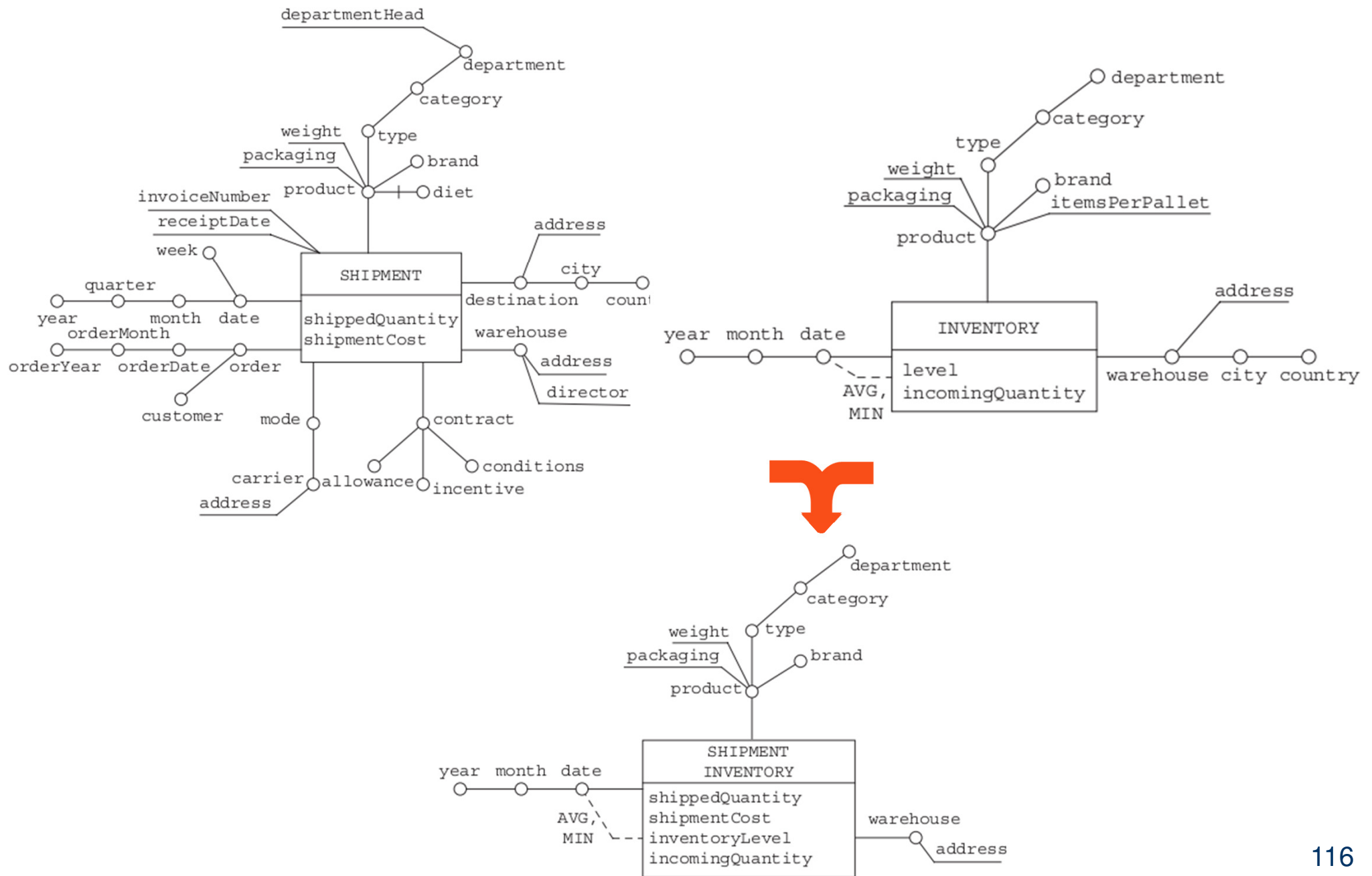


(COUNT) indicates that the information represented by a secondary event is the number of primary events corresponding to it

Overlapping fact schemata

- Separate fact schemata represent different facts in the DFM. However, part of the queries may require a comparison of the measures from two or more facts correlating to each other. In OLAP terminology, these are called **drill-across** queries
- To perform such queries we need to define which are the measures and which are the aggregation patterns that can be used when 2 or more fact schemata are queried. In an overlapping schema:
 - ✓ Measures are the union of the measures of the various schema
 - ✓ Hierarchies include only the attributes that occur in all hierarchies
 - ✓ The domain of each attribute is the intersection of the domains of the corresponding attributes in the original schema.

Overlapping fact schemata



Logical Models for Data Marts

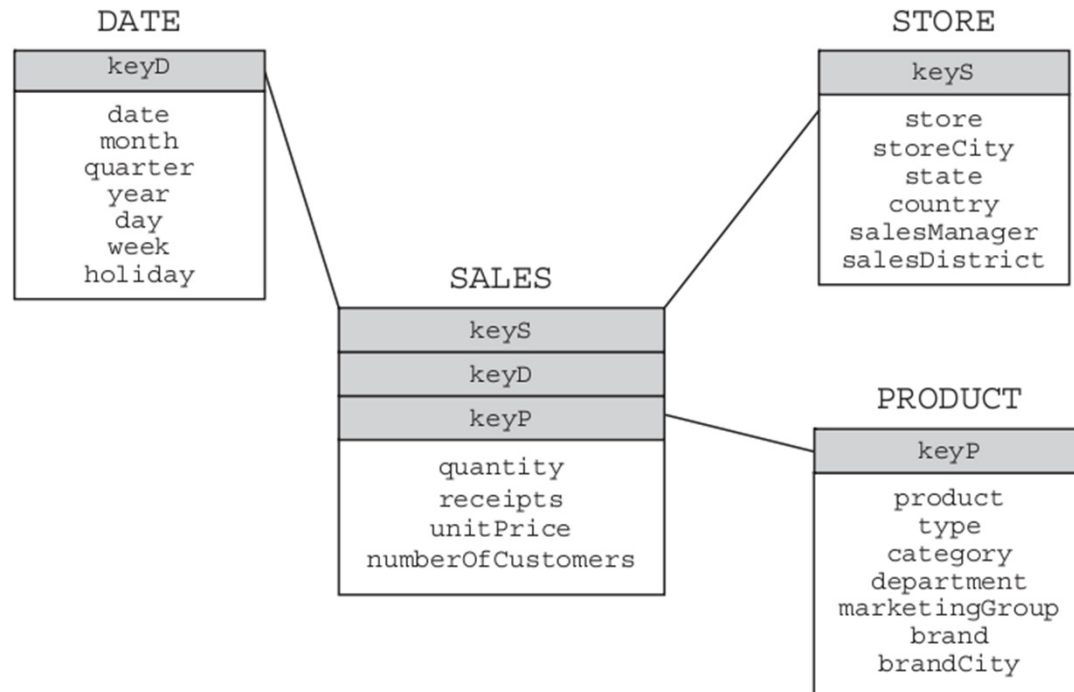
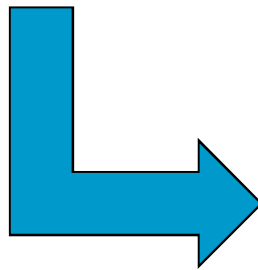
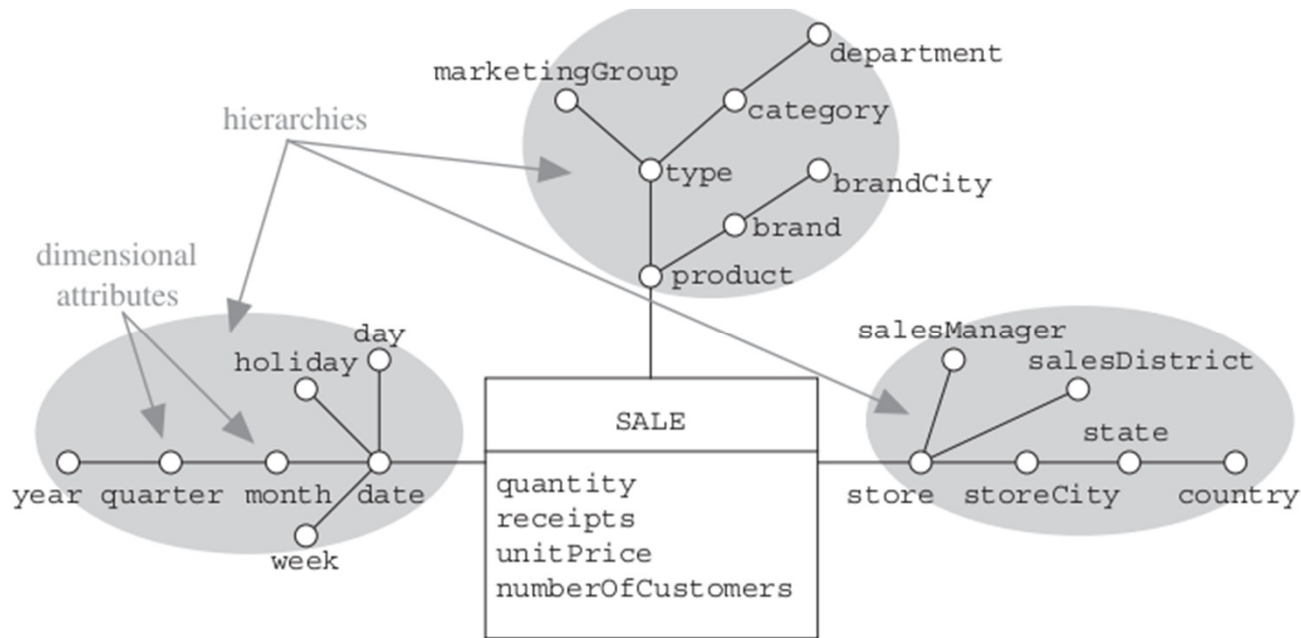
- Conceptual modeling does not depend on the logical model the designer has selected in the architecture design phase, but the topics related to logical modeling clearly do.
- As already said, Multidimensional structure can be represented using two distinct logical models:
 - ✓ MOLAP (*Multidimensional On-Line Analytical Processing*) stores data using structures that are intrinsically multidimensional (e.g., multidimensional vectors).
 - ✓ ROLAP (*Relational On-Line Analytical Processing*) makes use of the well-known relational model to represent multidimensional data.

ROLAP: The star schema

In what follows we concentrate on ROLAP system

- Multidimensional modeling in relational systems is based on the so-called **star schema** and star schema variants.
- A star schema is constituted by:
 - ✓ A set of relations DT_1, \dots, DT_n called **dimension tables**. Each of them corresponds to a dimension. Every DT_i features a primary (typically surrogate) key k_i and a set of attributes describing its dimension at different aggregation levels.
 - ✓ A **fact table** FT referencing all the dimension tables. An FT primary key is the composition of the set of foreign keys k_1 through k_n referencing dimension tables. An FT also contains an attribute for every measure.

Star schema: structure



Star schema: instances

SALES	<u>keyS</u>	<u>keyD</u>	<u>keyP</u>	quantity	receipts
	1	1	1	170	85
	2	1	2	320	160
	3	2	3	412	412

Fact Table

STORE	<u>keyS</u>	store	storeCity	state
	1	COOP1	Columbus	Ohio
	2	COOP2	Austin	Texas
	3	COOP3	Austin	Texas

Dimension Table

PRODUCT	<u>keyP</u>	product	type	category	brand
	1	Slurp Milk	Dairy product	Food	Slurp
	2	Fresh Milk	Dairy product	Food	Fresh
	3	Slurp Yogurt	Dairy product	Food	Slurp

Dimension Table

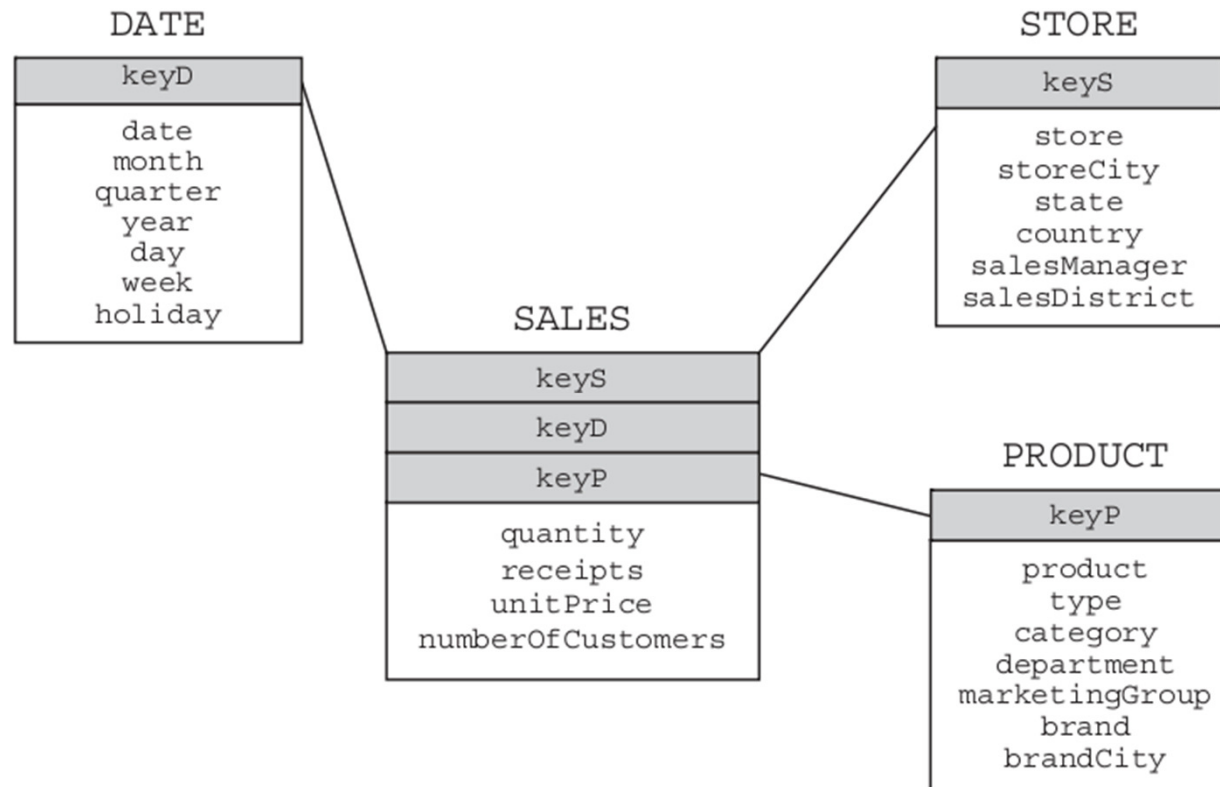
DATE	<u>keyD</u>	date	month	year
	1	9/2/2008	9/2008	2008
	2	10/3/2008	10/2008	2008
	3	10/5/2008	10/2008	2008

Dimension Table

Star schema: comments

- Dimension tables are completely denormalized (e.g., *product* → *type*; *type* → *category*)
 - 👍 A join is sufficient to obtain all data connected to a certain dimension
 - 👎 High redundancy in the data due to denormalization (even though redundancy is in the dimensional tables, whose size is often negligible)
- Fact Table contains tuples at the chosen level of aggregation
 - 👎 For very fine-grained facts, fact table can be very large and this can negatively affect access time to data
- No sparsity problems: we store only tuples corresponding to points in the multidimensional space for which an event occurred

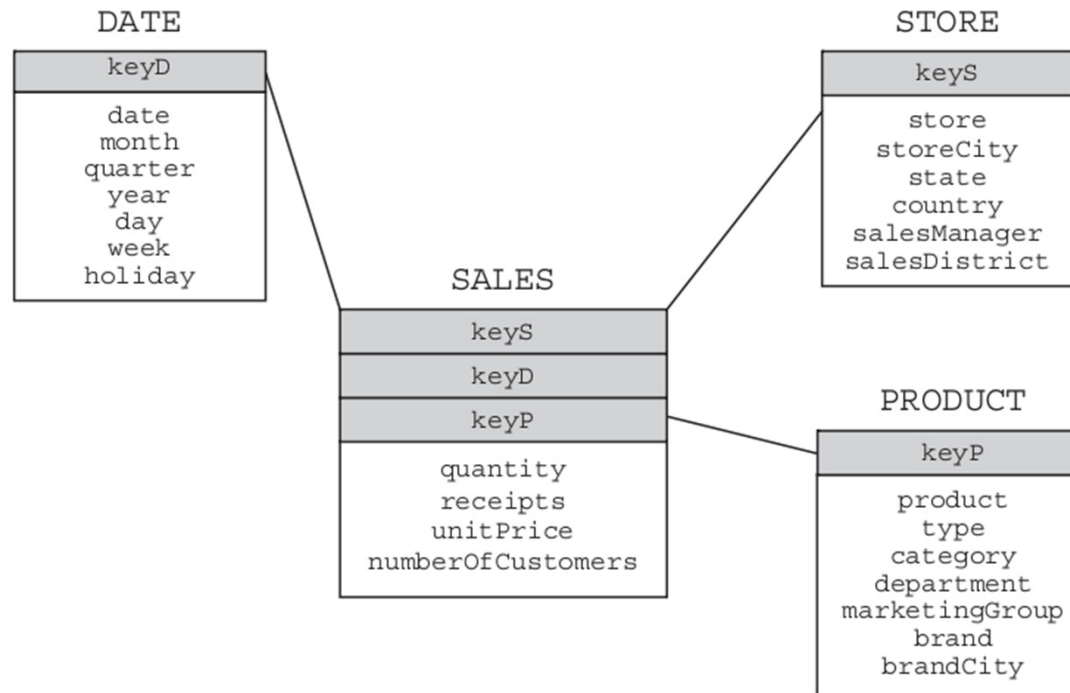
OLAP queries over star schemata



```
SELECT *  
FROM SALES AS FT, PRODUCT AS DT1, STORE AS DT2, DATA AS DT3  
WHERE FT.keyP = DT1.keyP AND FT.keyS = DT2.keyS AND  
FT.keyD = DT3.keyD
```

A multidimensional view of data is obtained when you join the fact table to its dimension tables

OLAP queries over star schemata



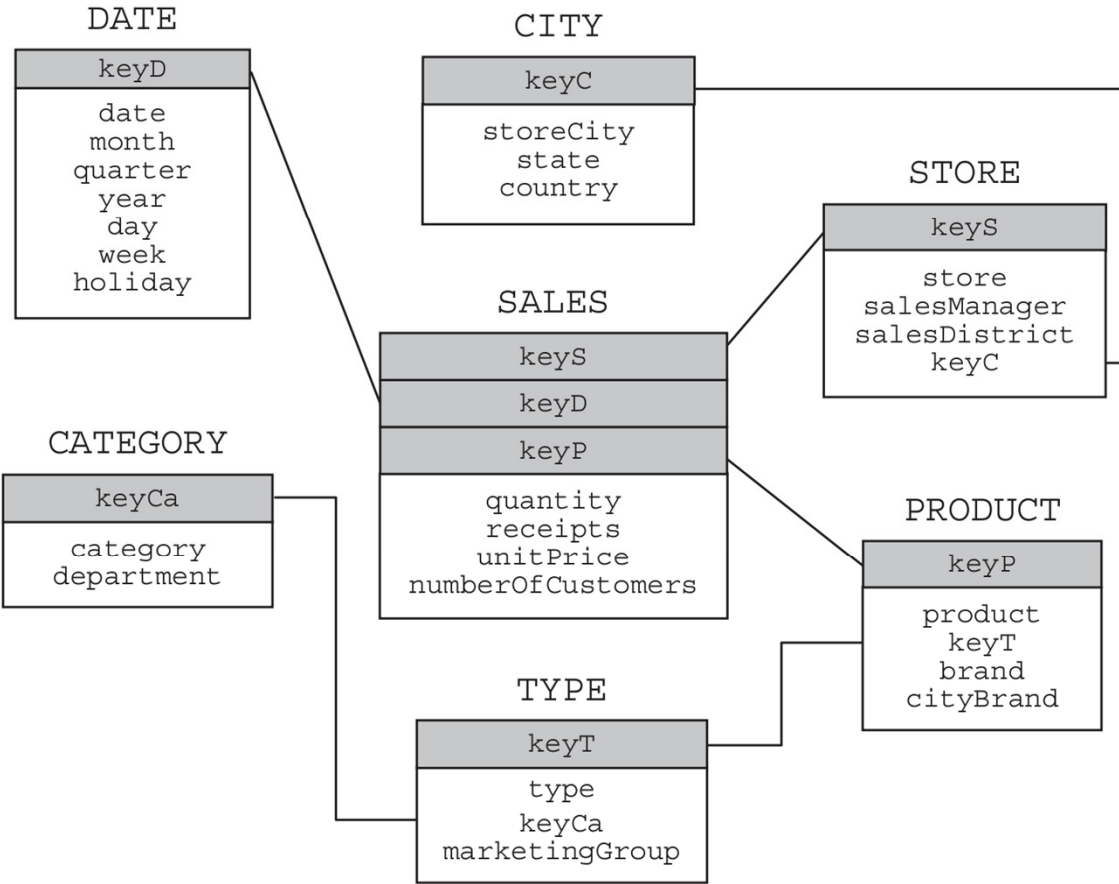
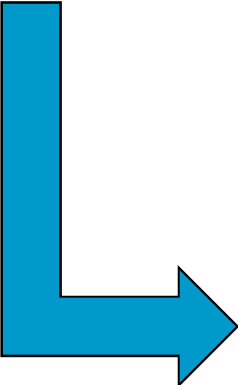
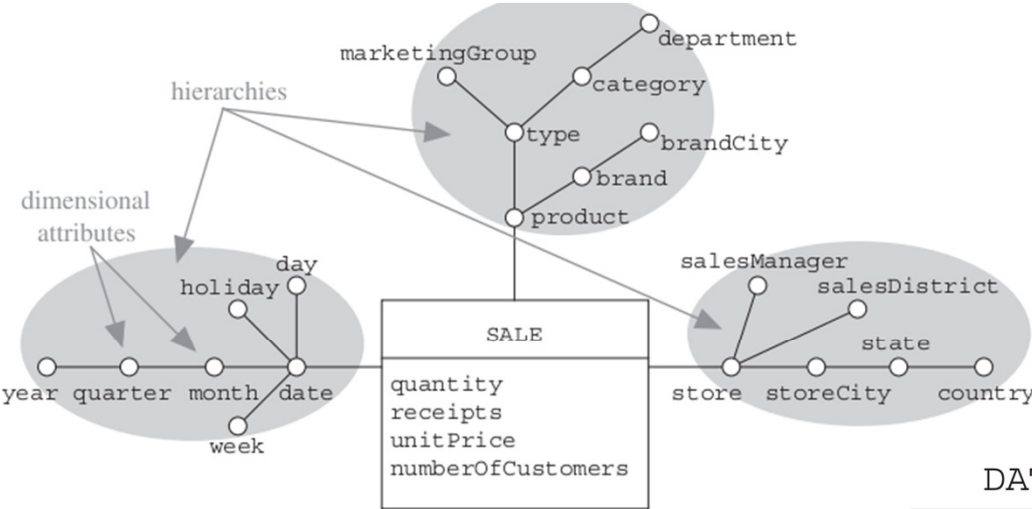
SALES (store.storeCity, date.week, product.type; product.category= 'Food').quantity

```
SELECT storeCity, week, type, sum(quantity)  
FROM SALES AS FT, PRODUCT AS DT1, STORE AS DT2, DATA AS DT3  
WHERE FT.keyP = DT1.keyP AND FT.keyS = DT2.keyS AND  
      FT.keyD = DT3.keyD AND DT1.category='Food'  
GROUP BY storeCity, week, type
```

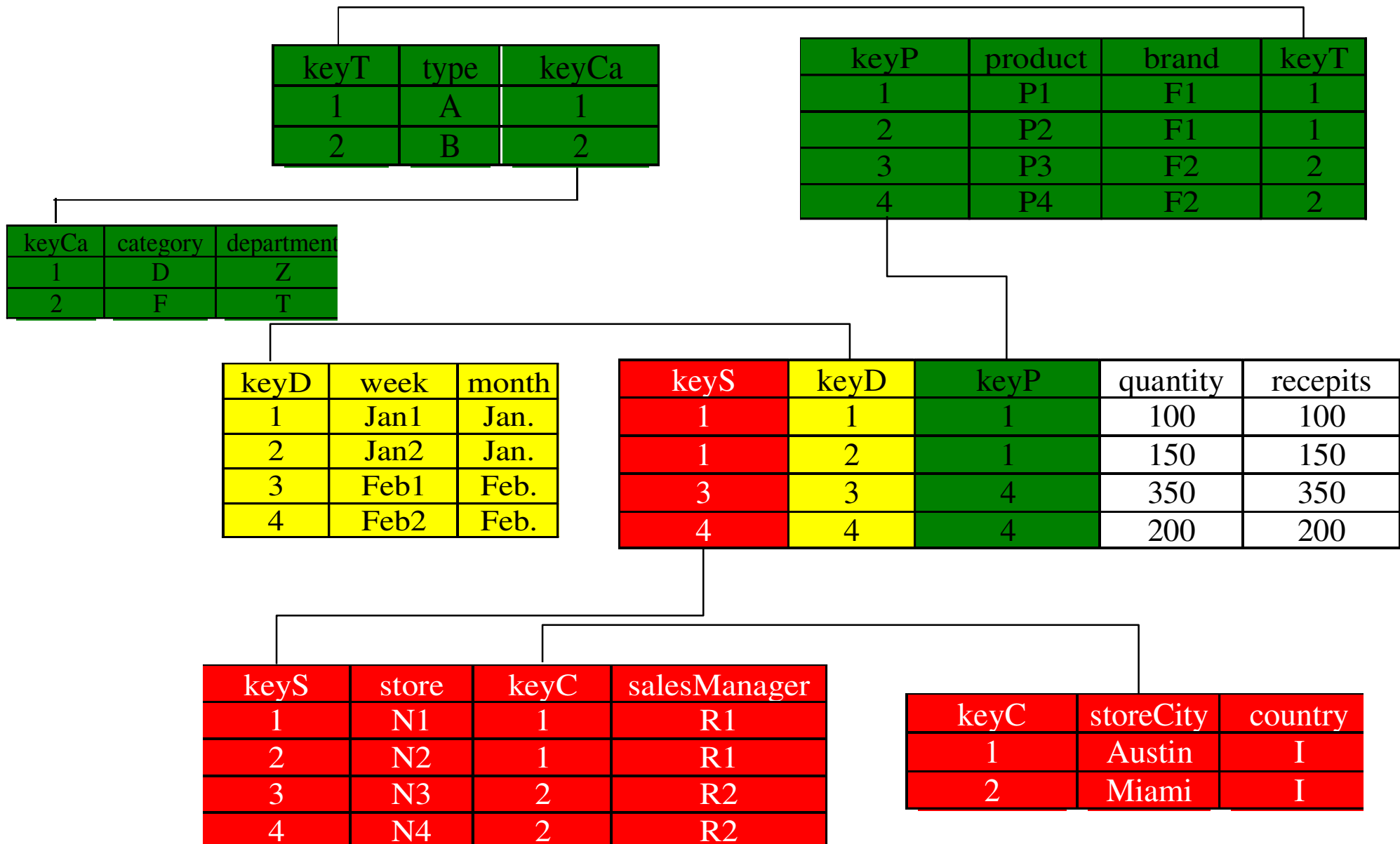
Snowflake schema

- Dimension tables are not in third normal form. For example, the STORE dimension table includes the transitive functional dependencies $\text{store} \rightarrow \text{storeCity}$, $\text{storeCity} \rightarrow \text{state}$.
- *Snowflake schema* reduces the denormalization of the dimension tables DT_i in the star schemata by eliminating some of the transitive dependencies that characterize them.
- A dimension table $DT_{i,j}$ in a snowflake schema is characterized by:
 - ✓ one primary key (typically surrogate) $d_{i,j}$
 - ✓ a subset of DT_i attributes functionally depending on $d_{i,j}$
 - ✓ some foreign keys, each referencing another $DT_{i,k}$ table, necessary for any DT_i information to be properly reconstructed
- We call *primary* the dimension tables whose keys are referenced in the fact table, *secondary* the remaining ones.

Snowflake schema: structure



Snowflake schema: instances



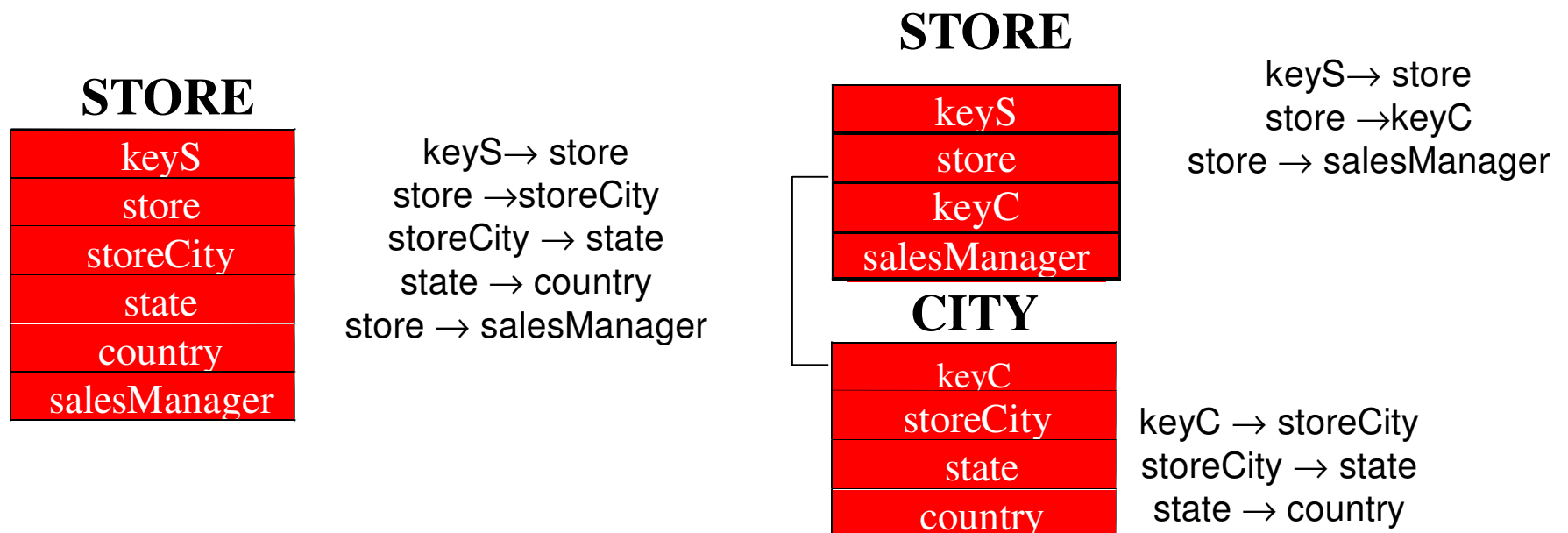
For ease of exposition, not all the dimensional attributes and measures are considered in this instantiation of the snowflake schema

Snowflake schema: comments

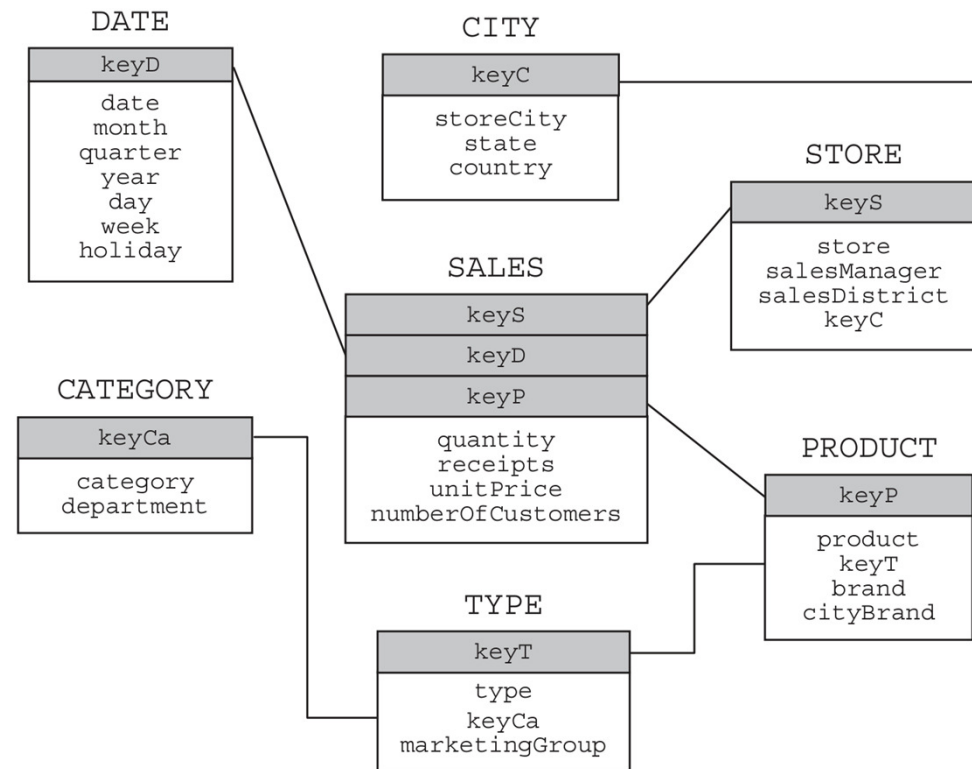
- The disk space required for data storage decreases by virtue of normalization.
- Snowflake schemata need new surrogate keys to be added in order to express the relationships between primary and secondary dimension tables.
- Processing the queries that involve only fact table attributes and primary dimension table attributes is streamlined because their joins involve smaller tables
- The time needed for queries of secondary dimension table attributes is longer because of a larger number of necessary joins.

Snowflake schema normalization

- When you decompose a dimension table to design a snowflake schema, you should check for an appropriate set of attributes to be inserted in the new relation
- To break down a schema effectively, you need for all those attributes that directly or transitively depend on the snowflaking attribute (that is, on the natural key of the new relation) to be part of the new relation.



OLAP queries over snowflake schemata



SALES (store.storeCity, date.week, product.type; product.category= 'Food').quantity

```
SELECT storeCity, week, type, sum(quantity)
FROM SALES AS FT, PRODUCT AS DT1, STORE AS DT2, DATA AS DT3,
TYPE AS DT4, CATEGORY AS DT5, CITY AS DT6
WHERE FT.keyP = DT1.keyP AND FT.keyS = DT2.keyS AND
FT.keyD = DT3.keyD AND DT6.keyC=DT2.keyC AND
DT4.keyT=DT1.keyT AND DT5.keyCa=DT4.keyCa AND
DT5.category='Food'
GROUP BY storeCity, week, type;
```

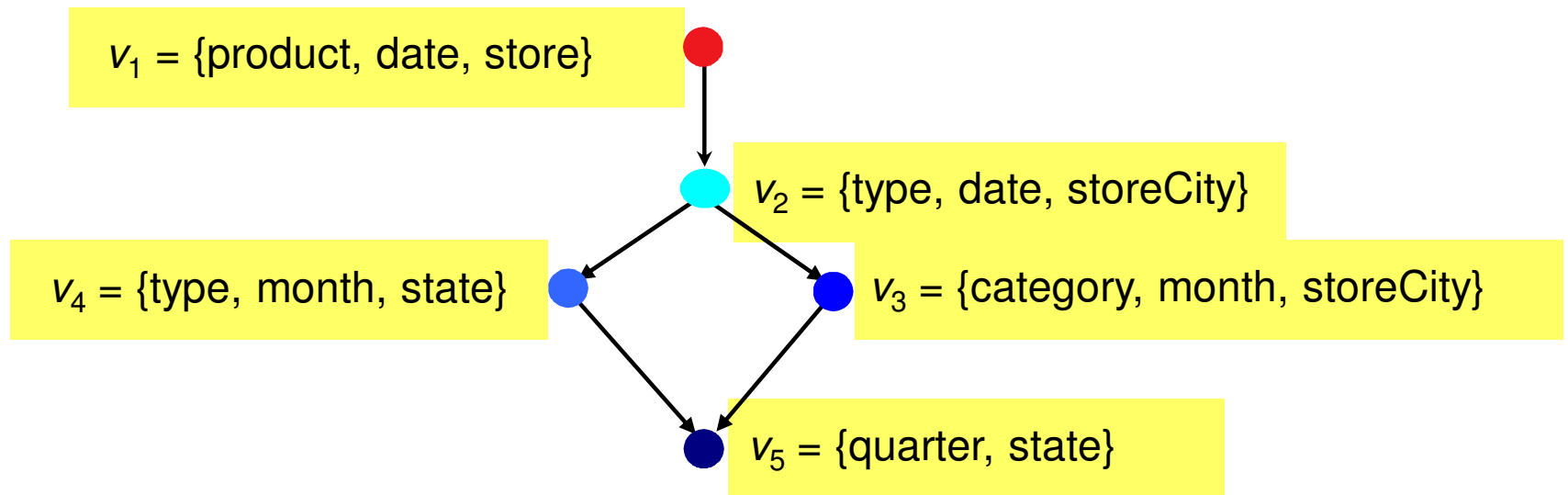
Views

- The huge amount of data stored in data warehouses makes users analyses difficult.
- For this reason users tend to apply selection and aggregation to decrease the parts of data they examine. But aggregations are costly operations...
- ...If you **calculate in advance** the most frequently used aggregate data, this can result in a significant increase in performance

The fact tables that include aggregate data are generally called views

Views

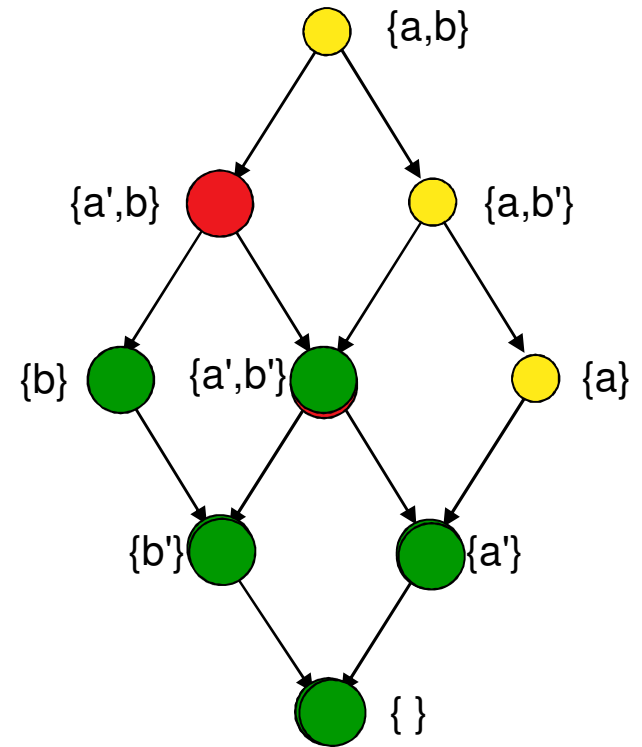
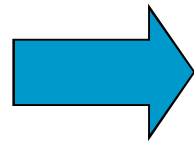
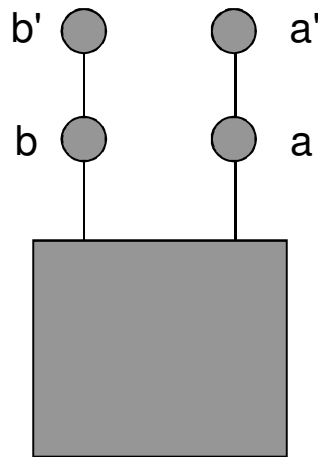
- Views can be classified on the basis of the aggregation level (*pattern*) that characterizes them



- Primary views:** correspond to the primary aggregation patterns (in fact non-aggregated, i.e., the finest, primary group-by set, which is the one defined by the fact schema dimensions)
- Secondary views:** correspond to the secondary aggregation pattern (aggregated)

Views and queries

- A view v over an aggregation pattern p is not only useful for queries with pattern p but also for all queries with a pattern p' that is more aggregated than p ($p \sqsubseteq p'$)



Multidimensional grid

Secondary views and aggregation


- Consider the following example, in which we on-the-fly calculate the total receipts starting from sold quantity and unitPrice

type	product	quantity	unitPrice	receipts
T1	P1	5	1,00	5,00
T1	P2	7	1,50	10,50
T2	P3	9	0,80	7,20
				22,70

Sum AVG

↓ ↓

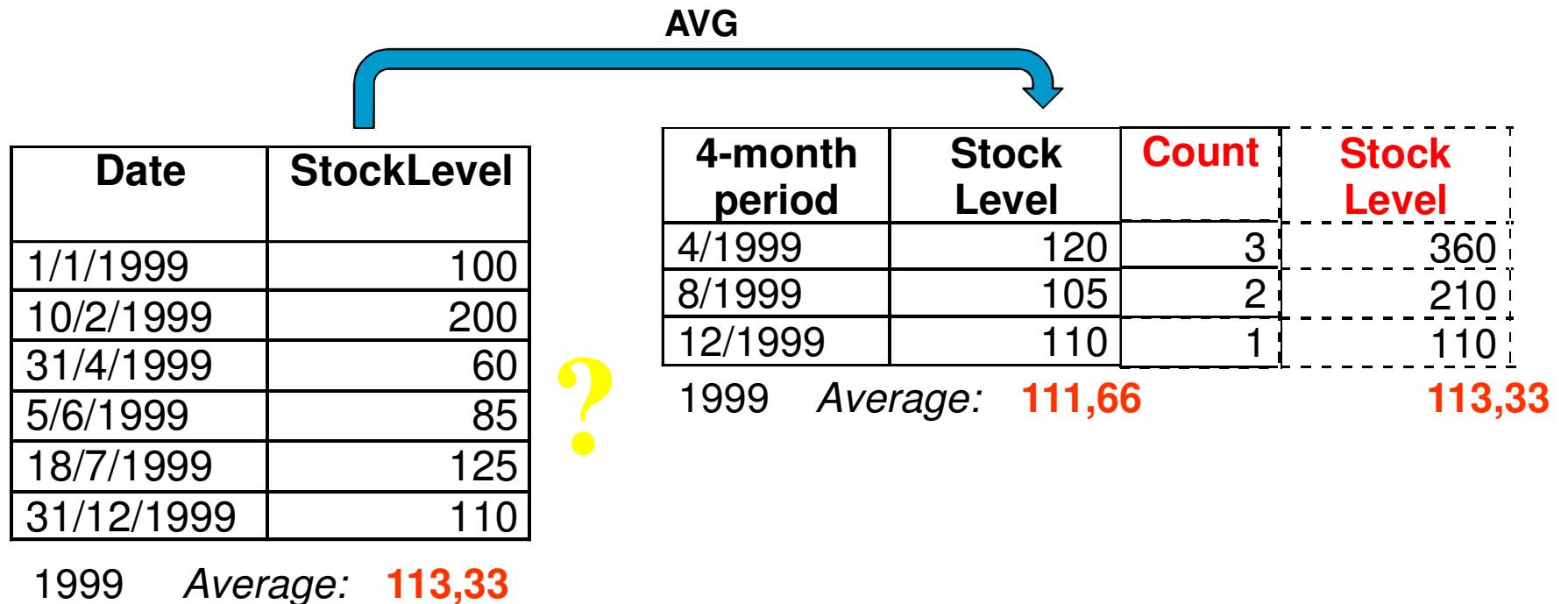
type	quantity	unitPrice	receipts
T1	12	1,25	15,00
T2	9	0,80	7,20
			22.20



- Due to the way in which we aggregated values, in the aggregated view it is not possible to obtain the correct value for total receipts.
- To correctly reconstruct information in secondary views it may be necessary to introduce new measures (**Derived Measures**)
- In the above example, the only way to obtain receipts is to explicitly store them in the aggregated view.

Secondary views and aggregation

- ✓ **Support Measures:** are necessary in the presence of non-distributive aggregate operators*



The correct solution is always the one that can be obtained by aggregating data directly from the primary view

*Distributive operators allow you to calculate aggregate data from data that is partially aggregated. The SUM, MIN, and MAX operators belong to this category

Relational schemata and views

- The simplest solutions consists in storing all (aggregate) data in a single fact table of the star schema
 - ✓ The size of the fact table increases (and performance can be negatively affected).
 - ✓ Dimension tables contain tuples referring to various aggregation levels. The NULL value is used to identify the aggregation level the dimension tuples refers to (each tuple in a dimension table will have a NULL value for all the attributes whose aggregation level is finer than the current one).

Relational schemata and views

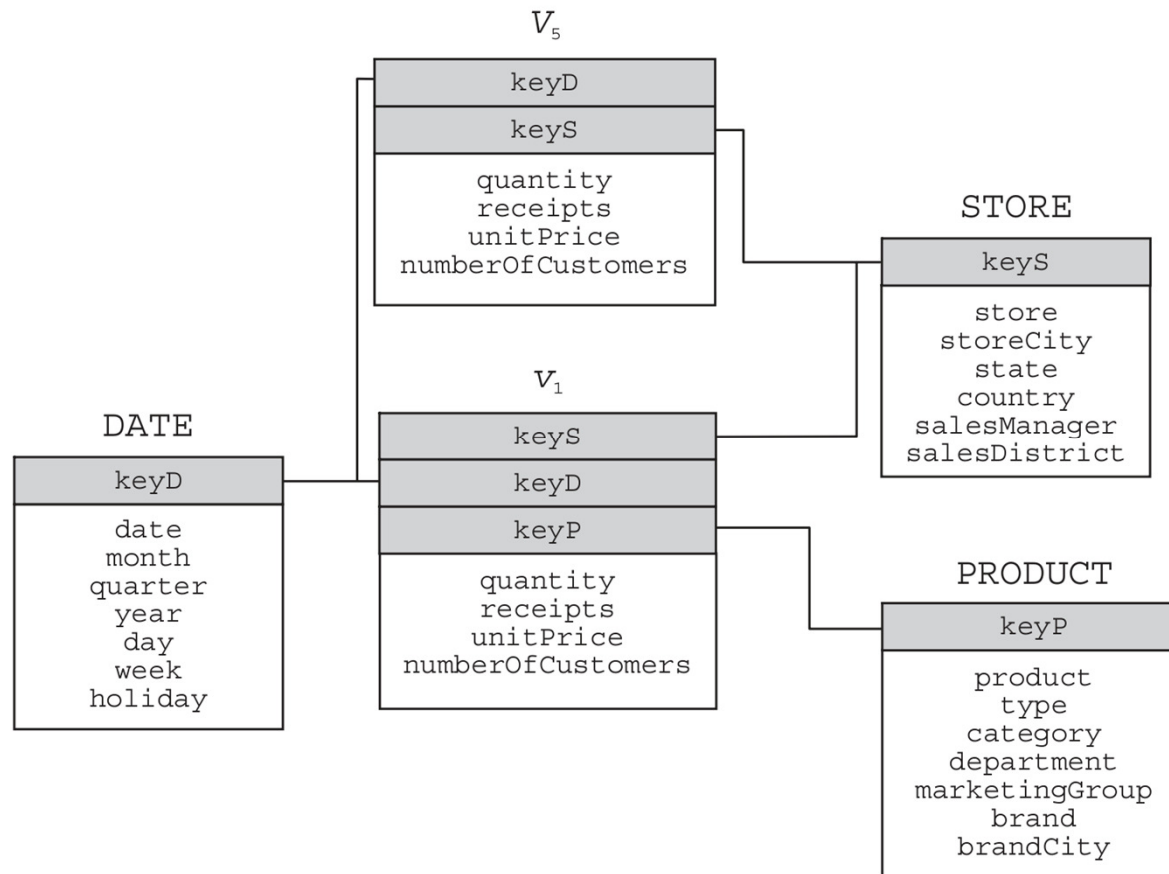
SALES	<u>keyS</u>	<u>keyD</u>	<u>keyP</u>	quantity	receipts
	1	1	1	170	85
	2	1	1	300	150
	3	1	1	1700	850

STORE	<u>keyS</u>	store	storeCity	state
	1	COOP1	Columbus	Ohio
	2	–	Austin	Texas
	3	–	–	Texas

Tuple 2 in SALES refers to the pattern {storeCity,date,product}, where storeCity is Austin, Tuple 3 in SALES refers to the pattern {state,date,product}, where state is Texas

Relational schemata and views

- Storing data related to different group-by sets into **separate fact tables** is another, more common solution.
- You can keep dimension tables merged, as in the previous solution, which leads to the so-called **constellation schema**



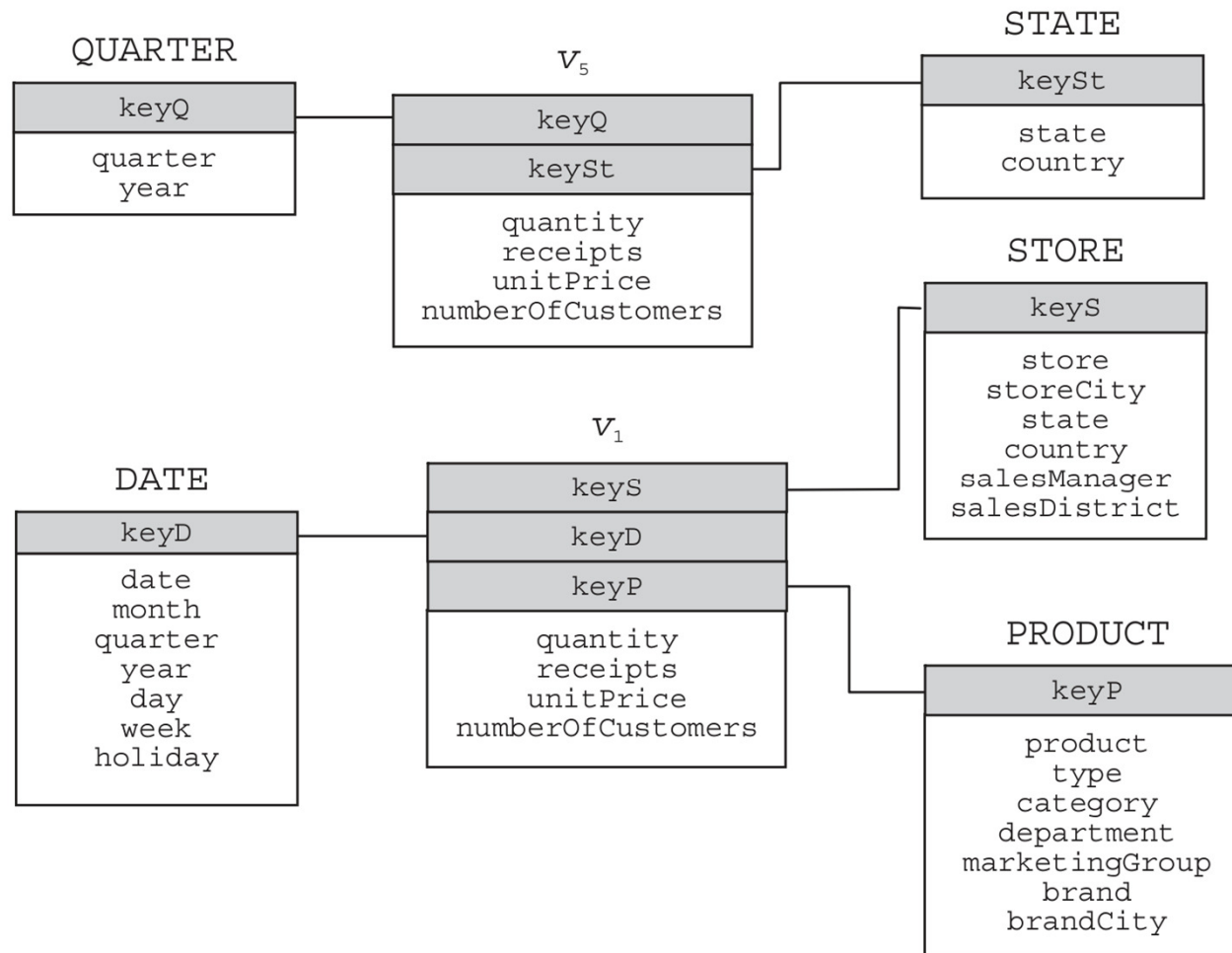
The fact tables that correspond to the group-by sets where one or more dimensions are completely aggregated do not have foreign keys referring to these aggregated dimensions

Relational schemata and views

- A constellation schema optimizes the access to fact tables because each of them includes only data at a specific aggregation level.
- This optimization is important, since the size of fact tables is far larger than the size of dimension tables
- However, dimension tables keep on including attributes at different hierarchy levels. The size of those dimension tables continues to be large, and NULL values still need to be inserted into invalid attributes for a specific aggregation level

Relational schemata and views

- You can choose to replicate dimension tables for views.
- Each dimension table will include only the set of attributes that are valid for the aggregation level at which that dimension table is created.

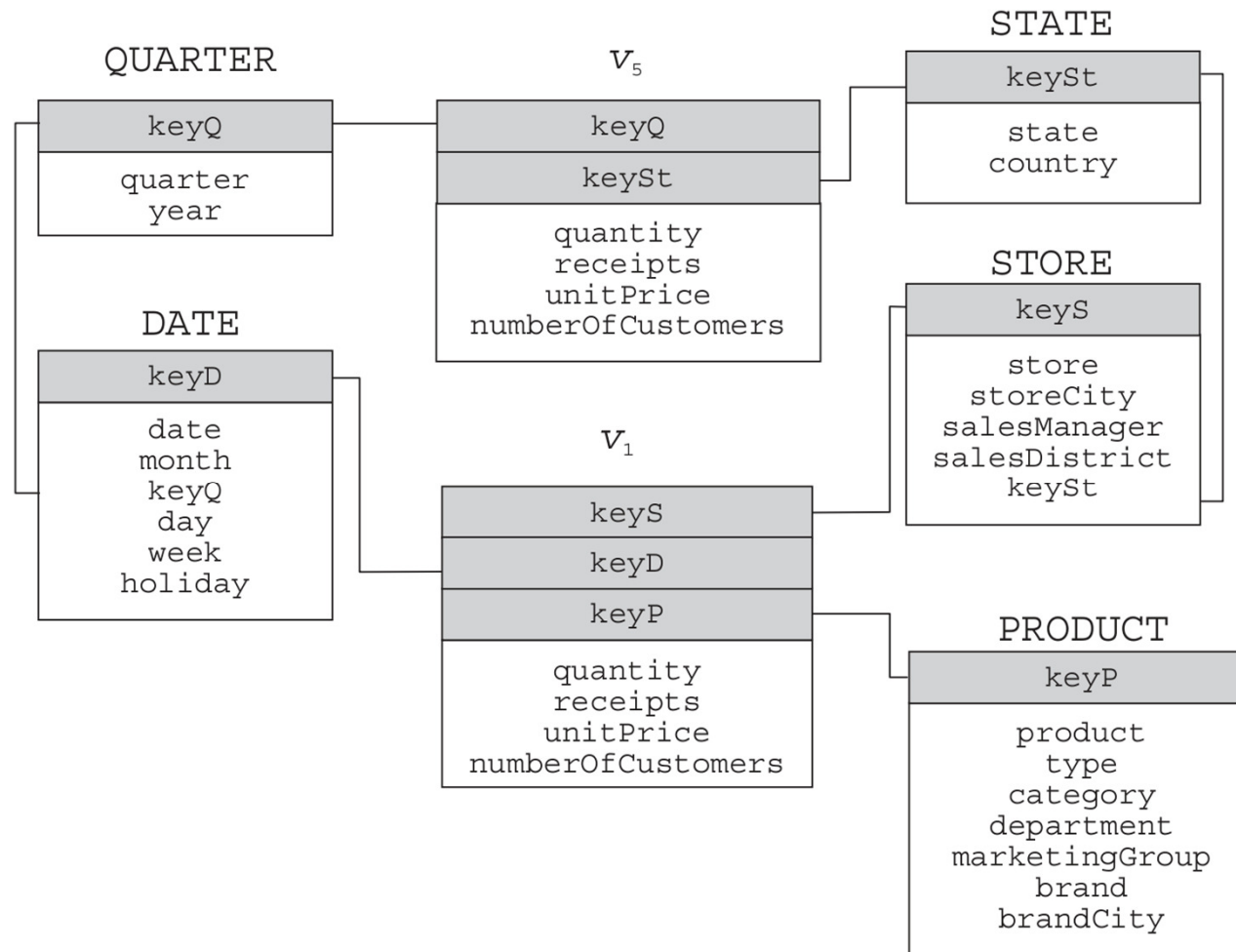


Relational schemata and views

- In general, this last solution is the one that is expected to deliver the best performance, since the access to fact tables and dimension tables has been completely optimized.
- However, maximizing performance is detrimental to the disk space required for data storage: more and more free space is needed not only to consolidate aggregate data, but also for redundant dimension tables.

Relational schemata and views

- A compromise between the solutions mentioned so far applies *snowflaking at the aggregation levels where aggregate views are materialized*. This solution takes advantage of the optimization achieved when you sort out aggregate data by aggregation level without replicating dimension tables.

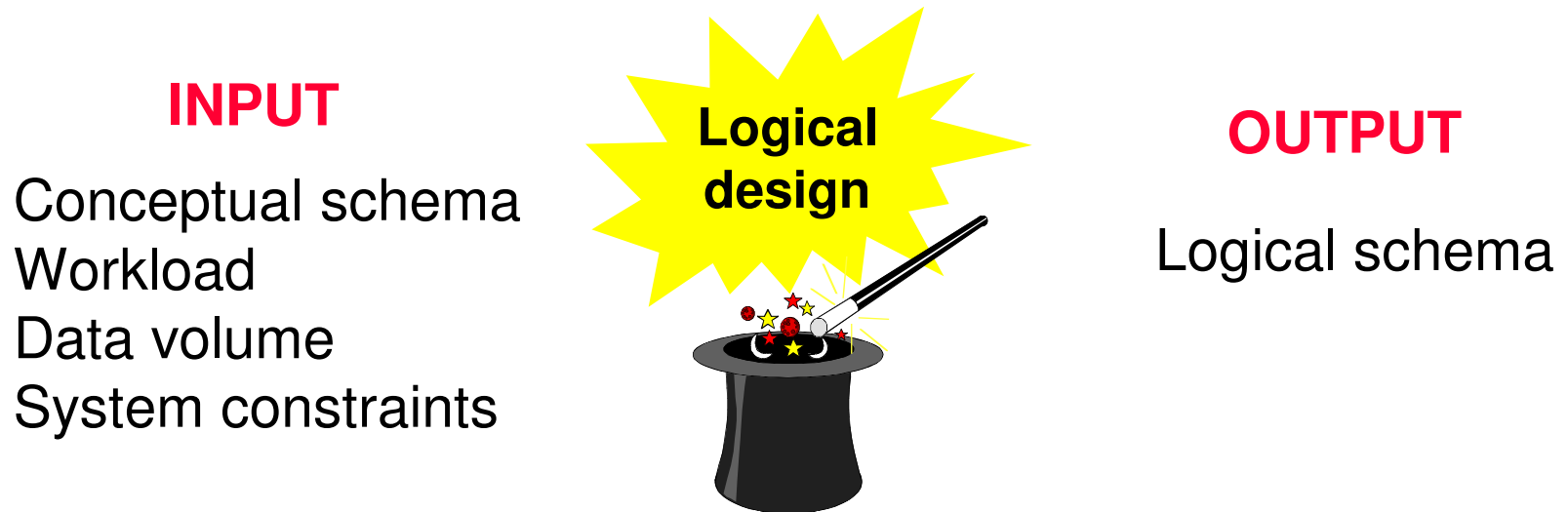


Aggregate navigators

- The presence of multiple fact tables that contain the data needed to solve a given query poses the problem of determining the view that will guarantee the minimum cost of execution.
- This role is played by the *aggregate navigator*, which are modules able to reformulate OLAP queries on the "best" available to view.
- The aggregate navigators of commercial systems currently manage only distributive aggregate operators, thus reducing the usefulness of the support measures.

Logical Design

- The logical design phase includes a set of steps that, starting from the conceptual schema, make it possible to define the logical schema of a data mart.



- Requires radically different techniques from those used for operational databases:
 - ✓ [Data redundancy](#)
 - ✓ [Denormalizations of relational tables](#)

Logical design

- Main steps in DW logical design are:
 1. Choosing the logical schema to adopt (e.g., star/snowflake schema)
 2. Translating fact schemata into logical schemata
 3. Materializing views
 4. Realizing some forms of optimization

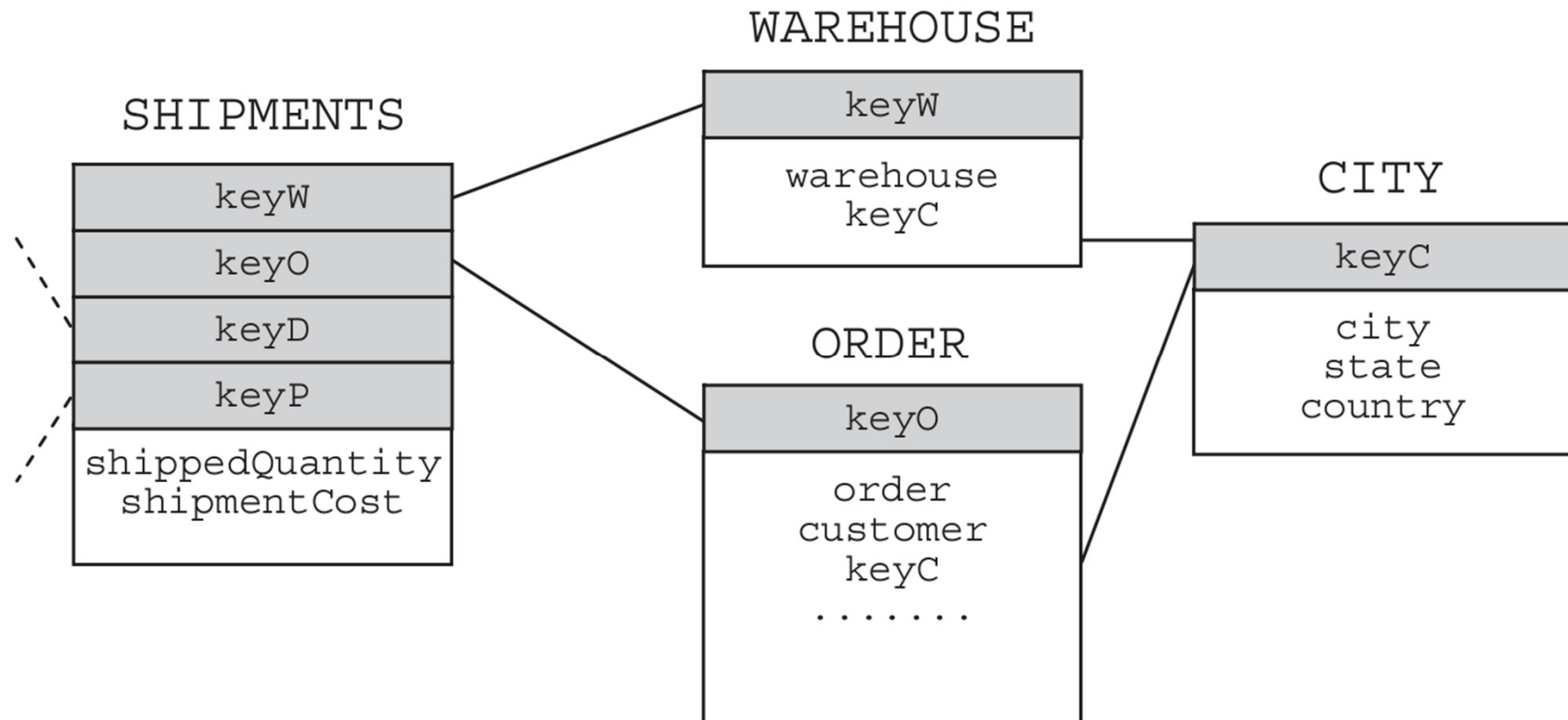
Star vs Snowflake

- There are discordant opinions on snowflaking:
 - ✓ Some authors consider it an embellishment that is in conflict with the founding philosophy of data warehousing

- It can be beneficial when:
 - ✓ the ratio between secondary dimension table cardinality and the primary dimension table one is very high, For example, if there were many products of one type, but few product types, normalization would save a remarkable amount of space, and the extra cost of a join, which is necessary to retrieve information on product types, would be small

Star vs Snowflake

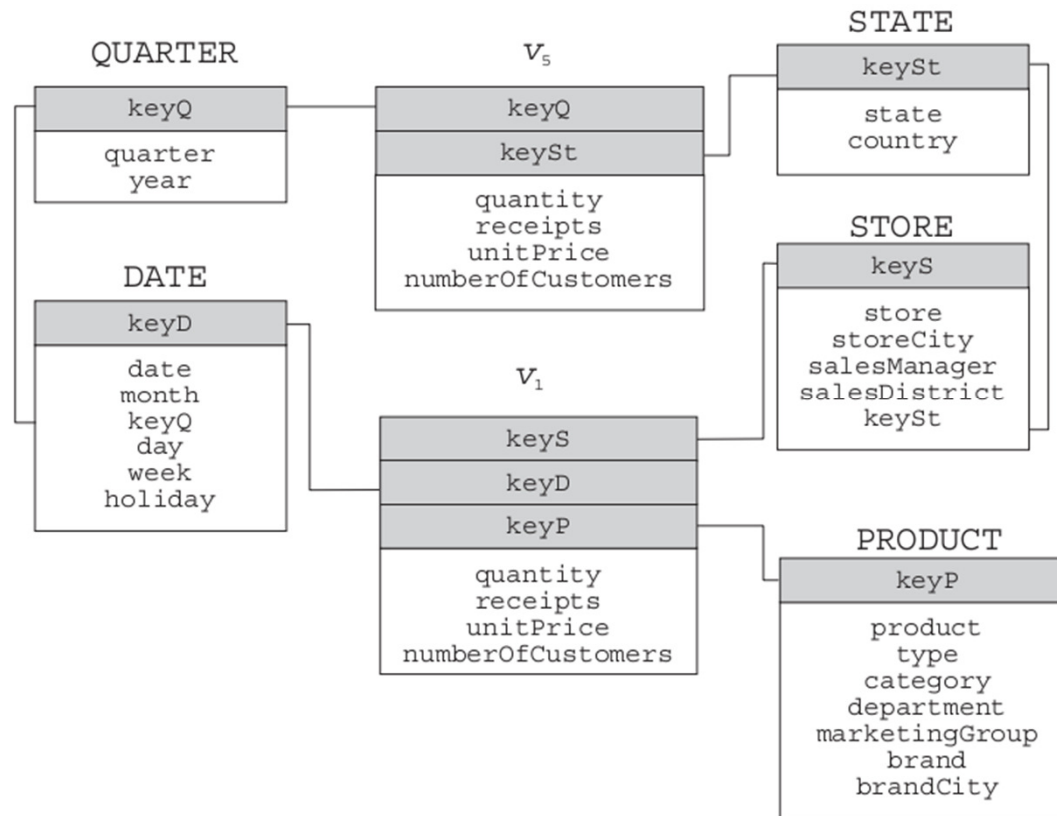
- It can be beneficial when
 - ✓ part of a hierarchy is common to more than one dimension



The secondary dimension table can be re-used for various hierarchies

Star vs Snowflake

- It can be beneficial when
 - ✓ There are materialized aggregate views



This means that secondary dimension tables are defined at the aggregation levels required for materialized views, so that secondary dimension tables of primary fact tables are primary dimension tables for aggregate fact tables.

From fact schemata to star schemata

- The basic rule for the translation of a fact schema into a star schema is the following

Create a fact table containing all the measures and descriptive attributes directly connected with the fact, and for each hierarchy, create a dimension table that contains all attributes.

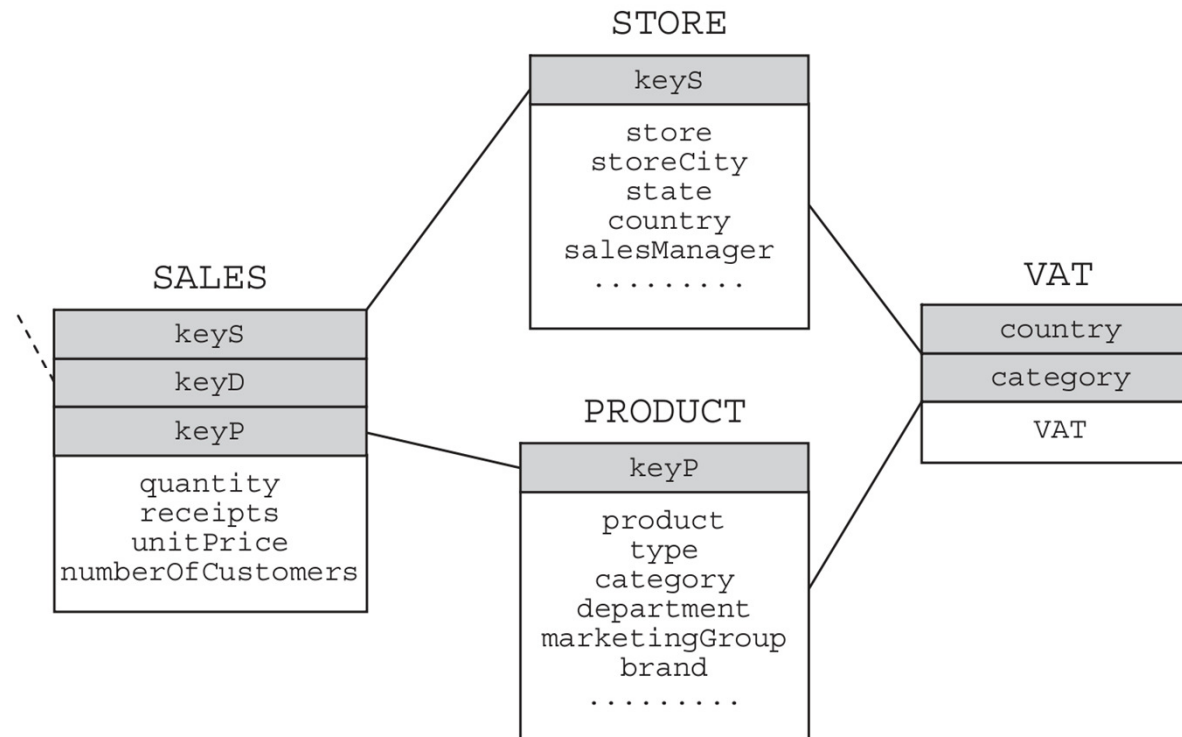
- In addition to this simple rule, the correct translation of a fact schema requires an in-depth discussion of the advanced constructs of the DFM.

Descriptive attributes

- Contain information that will not be used to perform aggregations but that is considered useful.
 - ✓ When connected to a dimensional attribute, it has to be included in the dimension table that contains the attribute.
 - ✓ If connected directly to the fact, it has to be included in the fact table.
- It only makes sense if it is compatible with the level of granularity of the event described in the fact table, so if connected directly to the fact table **it will be omitted in the aggregated views**. Also, if it is connected to a dimensional attribute, it can be included only in the dimensional tables that contain the attribute it refers to.

Cross-dimensional attributes

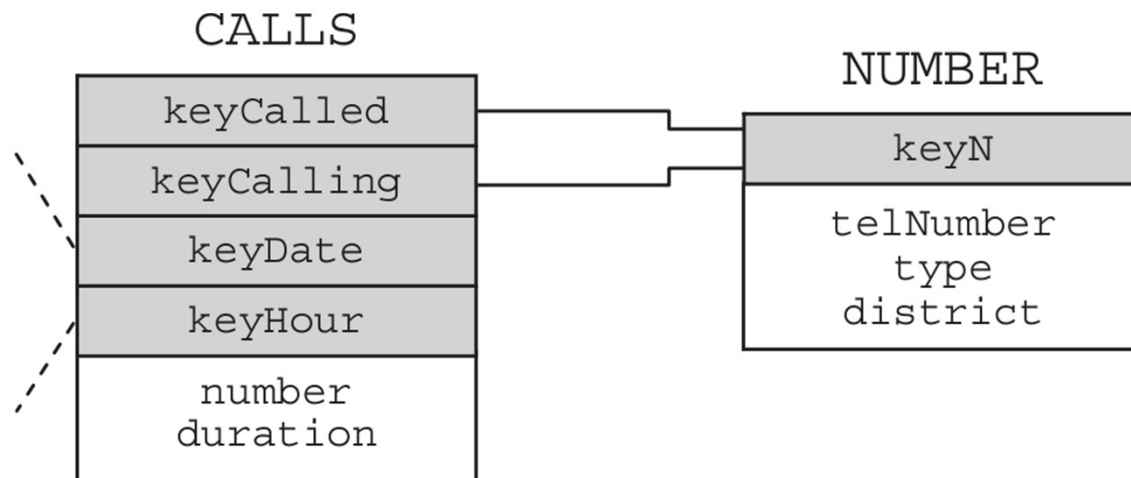
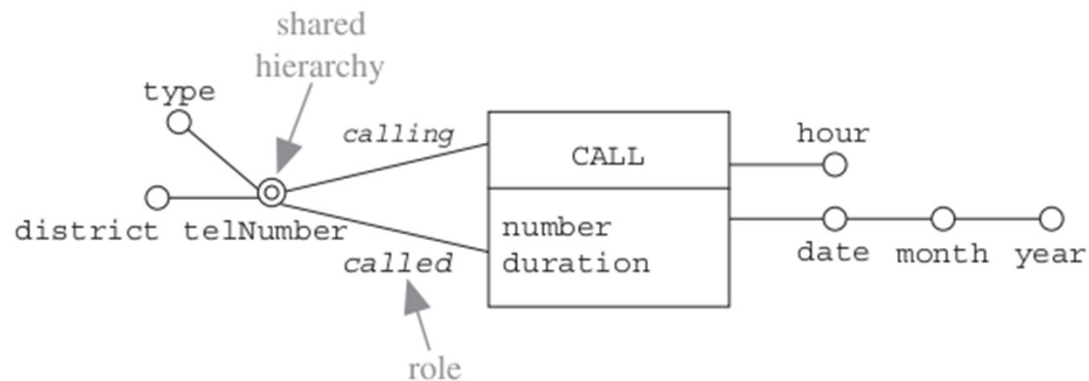
- From the conceptual point of view, a cross-dimensional attribute b defines a many-to-many relationship between two or more dimensional attributes a_1, \dots, a_m .
- Its translation into a logical schema requires you to enter a new table that includes b and has a_1, \dots, a_m as key attribute.



The choice between surrogate or non-surrogate keys for the new table should be made according to the space occupation that they cause

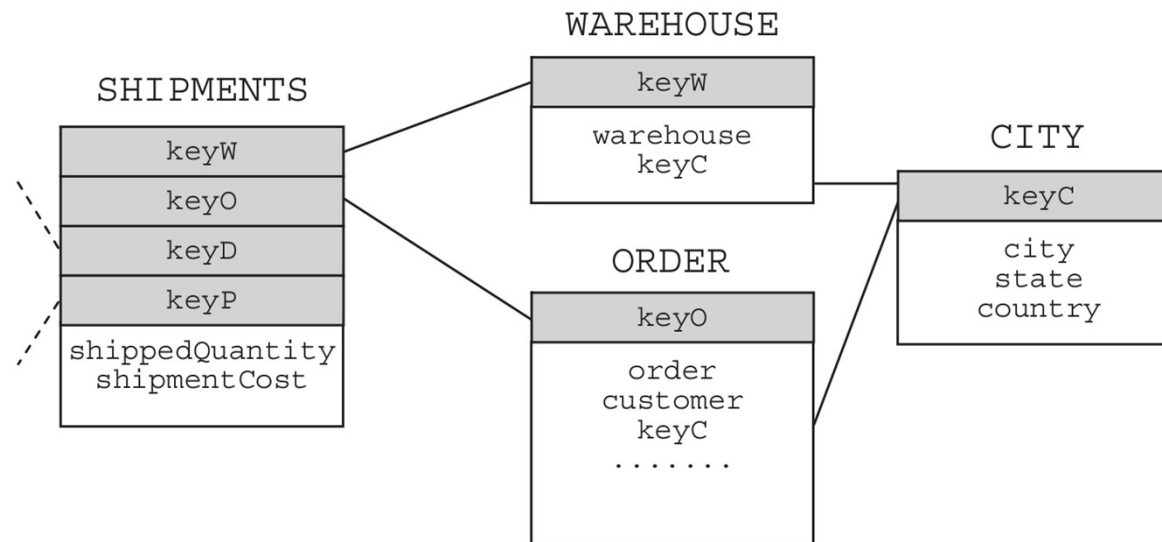
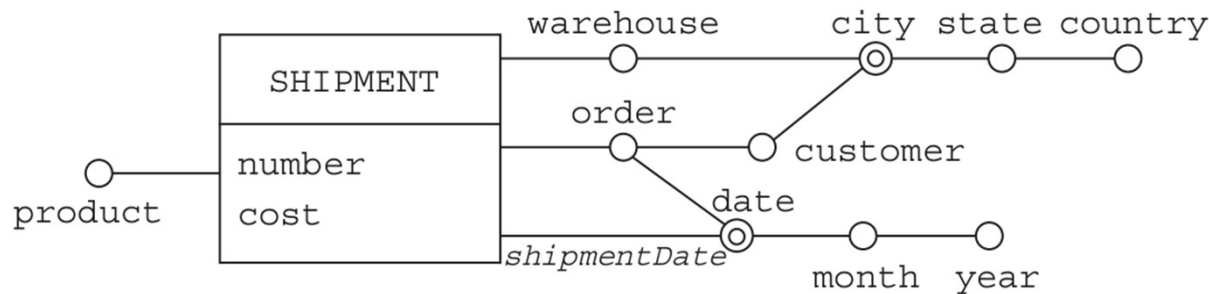
Shared Hierarchies

- If a hierarchy occurs several times in the same fact (or in different facts) it is not convenient to introduce redundant copies of the related dimension table.
- If two (or more) hierarchies contain exactly the same attributes it is sufficient to simply import the key twice (or more) in the fact table connected to them



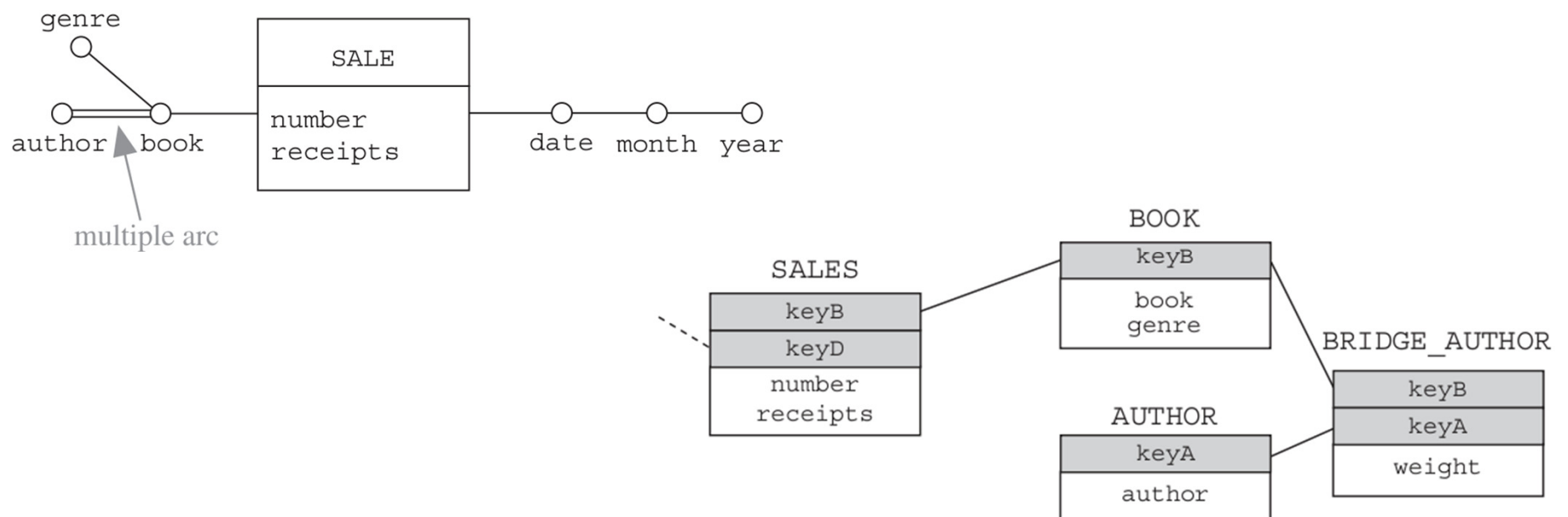
Shared hierarchies

- If the two hierarchies share only some of the attributes you need to decide if:
 - I. Introducing additional redundancy in the diagram by duplicating hierarchies and replicating the common fields.
 - II. Snowflaking the schema on the first shared attribute, and thus introducing a third table common to both the dimension table.



Multiple arcs

- Although it is not very common, some hierarchies can also model many-to-many relations.
- The most obvious design solution is to insert an additional table (**bridge table**) that models the multiple arc:
 - ✓ The key of the bridge table is the combination of the attributes connected by the multiple arc.
 - ✓ An (optional) *weight* attribute makes it also possible to consider to assign (in terms of a percentage) different relevance to the participating tuples



Multiple arcs

BOOK	<u>keyB</u>	book	genre
	1	Facts & Crimes	Technical
	2	Sounds Logical	Technical
	3	The Right Measure	Current affairs
	4	Facts: How and Why	Current affairs
	5	The 4 th Dimension	Science fiction

AUTHOR	<u>keyA</u>	author
	1	Matteo Golfarelli
	2	Stefano Rizzi

BRIDGE_AUTHOR	<u>keyB</u>	<u>keyA</u>	weight
	1	1	0.5
	1	2	0.5
	2	1	1.0
	3	2	1.0
	4	1	0.5
	4	2	0.5
	5	1	1.0

SALES	<u>keyB</u>	<u>keyD</u>	number	receipts
	1	1	3	150
	2	1	5	250
	3	1	10	300
	4	1	4	80
	5	1	8	400

Multiple arcs

- It may take up to 3 join to retrieve all the information contained in the hierarchy
- Two kinds of queries are possible :
 - ✓ **Weighted queries** : consider the weight of the multiple arc and thus provide the actual total

Total receipts for each author:

```
SELECT    AUTHOR.author, sum(SALES.receipts * BRIDGE_AUTHOR.weight)
FROM      AUTHOR, BRIDGE_AUTHOR, BOOK, SALES
WHERE     AUTHOR.keyA = BRIDGE_AUTHOR.keyA
AND       BRIDGE_AUTHOR.keyB = BOOK.keyB
AND       BOOK.keyB = SALES.keyB
GROUP BY AUTHOR.author
```

Multiple arcs

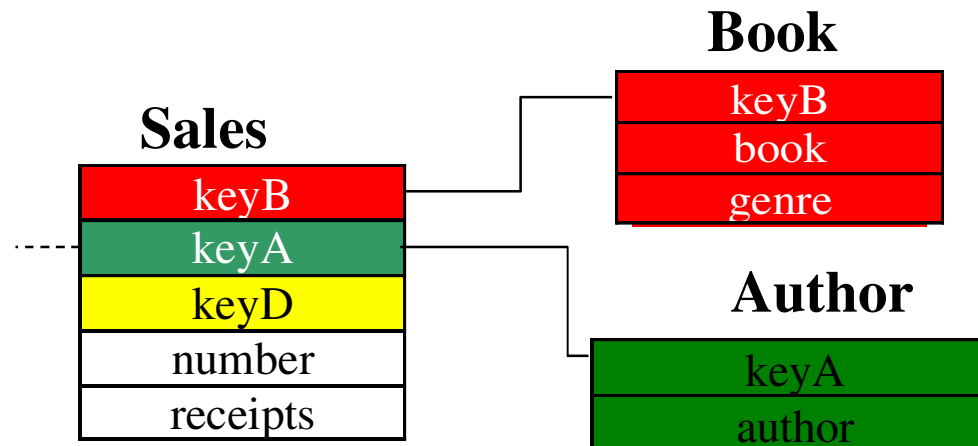
- It may take up to 3 join to retrieve all the information contained in the hierarchy
- Two kinds of queries are possible :
 - ✓ **Impact queries:** Resulting quantities are higher because the weight is not used

Number of books for each author:

```
SELECT    AUTHOR.author, sum(SALES.number)
FROM      AUTHOR, BRIDGE_AUTHOR, BOOK, SALES
WHERE     AUTHOR.keyA = BRIDGE_AUTHOR.keyA
AND       BRIDGE_AUTHOR.keyB = BOOK.keyB
AND       BOOK.keyB = SALES.keyB
GROUP BY  AUTHOR.author
```

Multiple arcs

- If you wish to continue using the star schema it is necessary to make finer the granularity of the fact thus shaping the multiple arc directly in the fact table (push-down).
- This solution requires the addition of a new dimension to the fact table corresponding to the terminal attribute of the multiple arc



Multiple arcs

SALES

<u>keyB</u>	<u>keyA</u>	<u>keyD</u>	number	receipts
1	1	1	1.5	75
1	2	1	1.5	75
2	1	1	5	250
3	2	1	10	300
4	1	1	2	40
4	2	1	2	40
5	1	1	8	400

An instance for the book sales fact table with author push-down

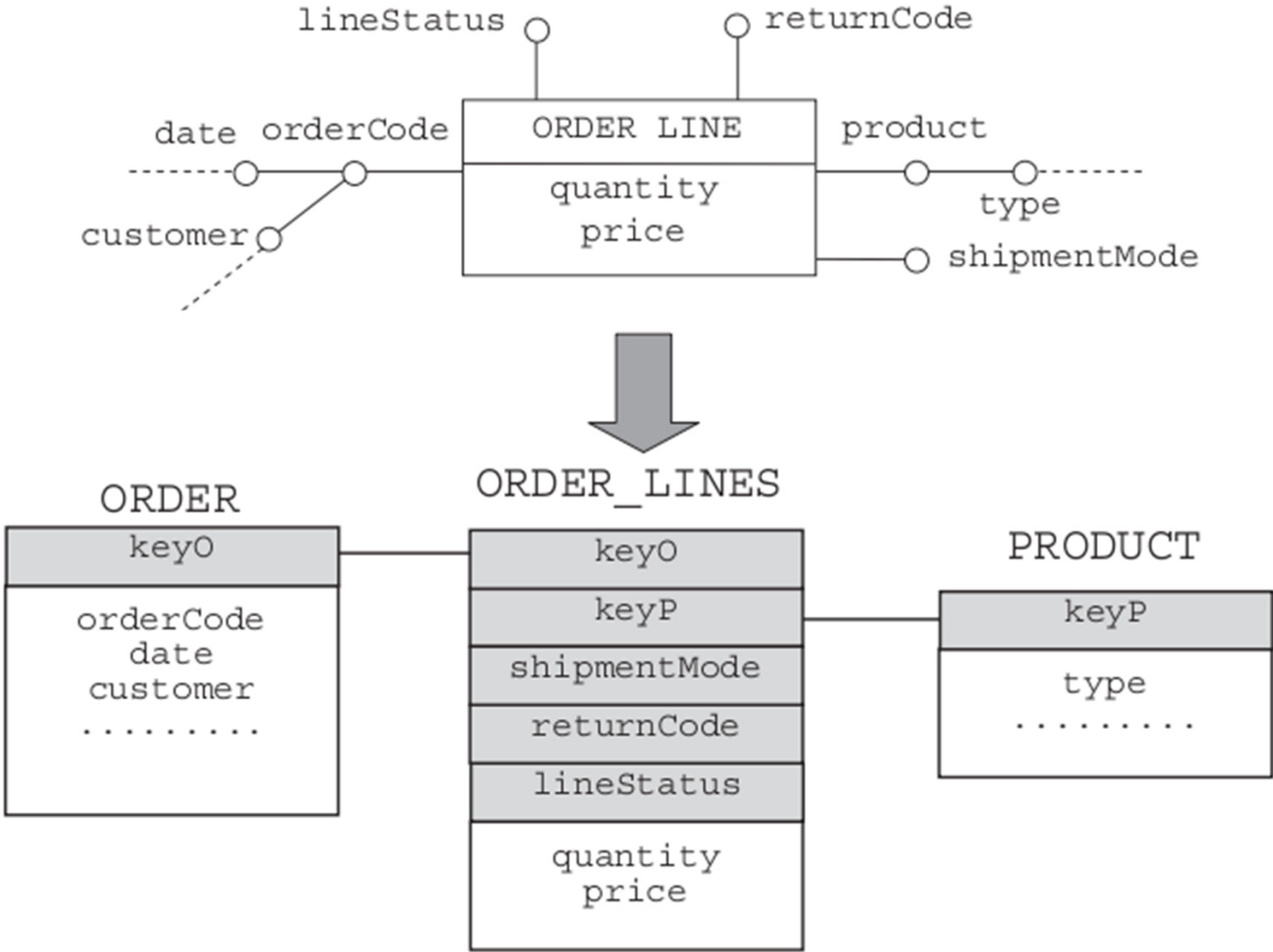
Multiple arcs: comparison

- From the informative point of view the two solutions are identical
- With the **push-down solution**:
 - ✓ It introduces a strong redundancy in the fact-table whose rows need to be replicated as many times as there are matches of the multiple arc
 - ✓ The weight is hardcoded in the fact table and its update can be very complex
 - ✓ Impact queries are very complex
 - ✓ The query execution cost is reduced since less joins are required
 - ✓ The calculation of the weighted events may occur during population of the data mart.
- With the **bridge-table solution**:
 - ✓ The query execution cost is reduced due to the lower number of tuples involved (but more joins are necessary)
 - ✓ The calculation of the weighted events occurs during the query

Degenerate Dimensions

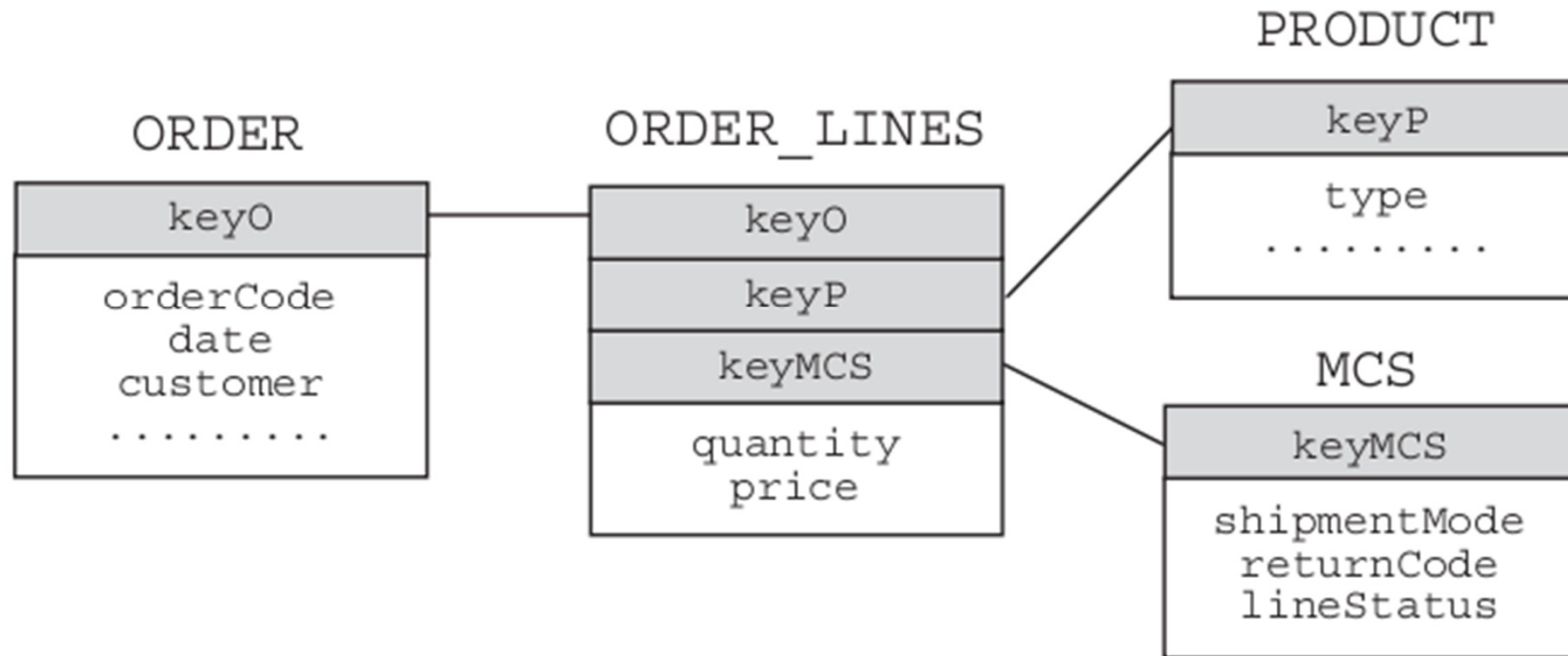
- The term degenerate dimension defines a hierarchy that includes only one attribute.
- If the byte length of the single attribute in the dimension is (much) greater than that of a surrogate key, then it may be reasonable to create a dedicated dimension table. We store degenerate dimensions directly in the fact table, otherwise. Indeed, this allows for reducing the number of joins in the queries.
- **Junk dimensions** are an alternative solution to the problem of degenerate dimensions
 - ✓ A junk dimension is a single dimension table that includes a set of degenerate dimensions
 - ✓ A junk dimension does not contain any functional dependencies between attributes. So all possible value combinations are valid, and thus a high number of tuples can be generated.
 - ✓ For this reason, this solution is viable only when the number of valid combinations for the attributes involved is not very large.

Degenerate dimensions



Solution with no dimension tables for lineStatus, returnCode, and shipmentMode

Degenerate dimensions



Solution with the junk dimension MCS