

Information Systems

Chapter 7:

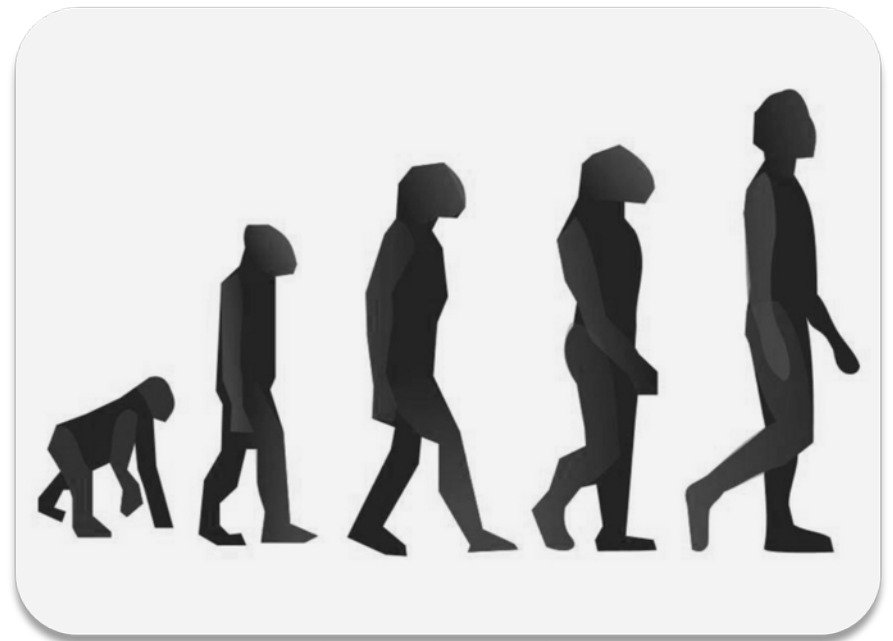
Data Warehousing

Rita Schindler | TU Ilmenau, Germany

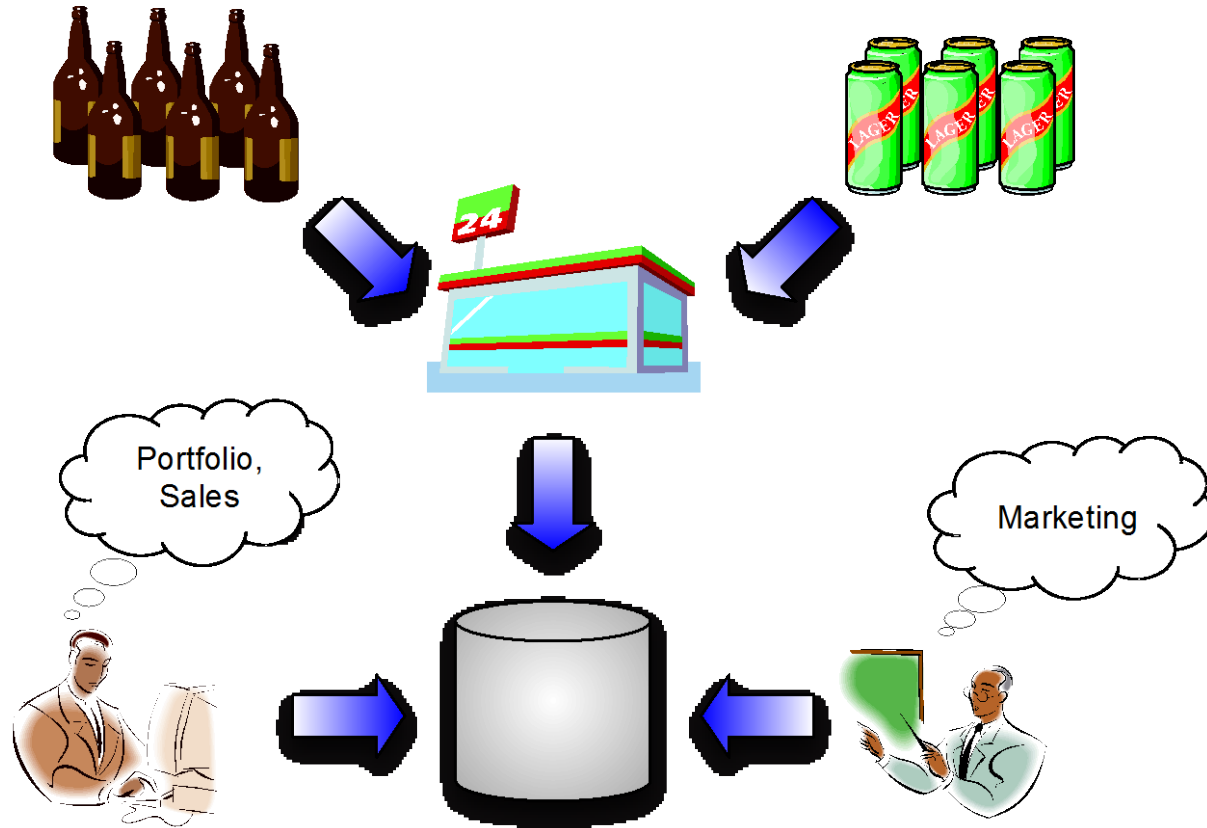
www.tu-ilmenau.de/dbis

Outline

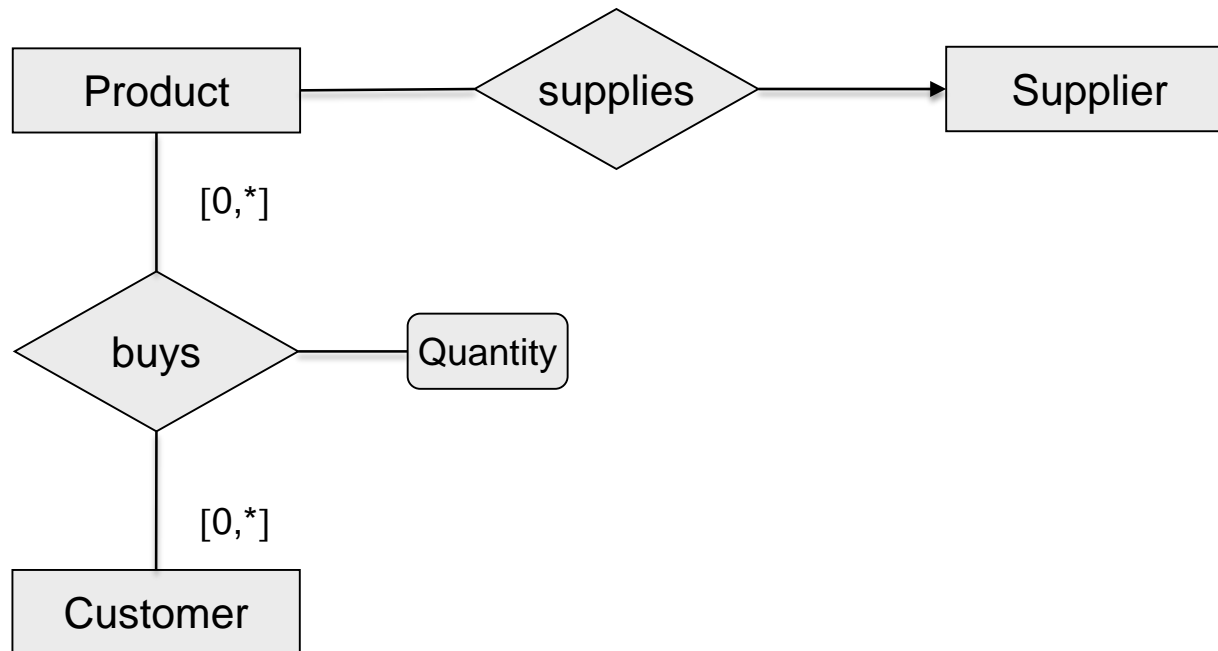
- ▶ DW Architecture
- ▶ Multidimensional Data Model
- ▶ Relational Mapping
- ▶ OLAP Operations
- ▶ Star Joins
- ▶ SQL Extensions for DW



Motivation: Retailer Database



Database Schema



Querying and Analytics

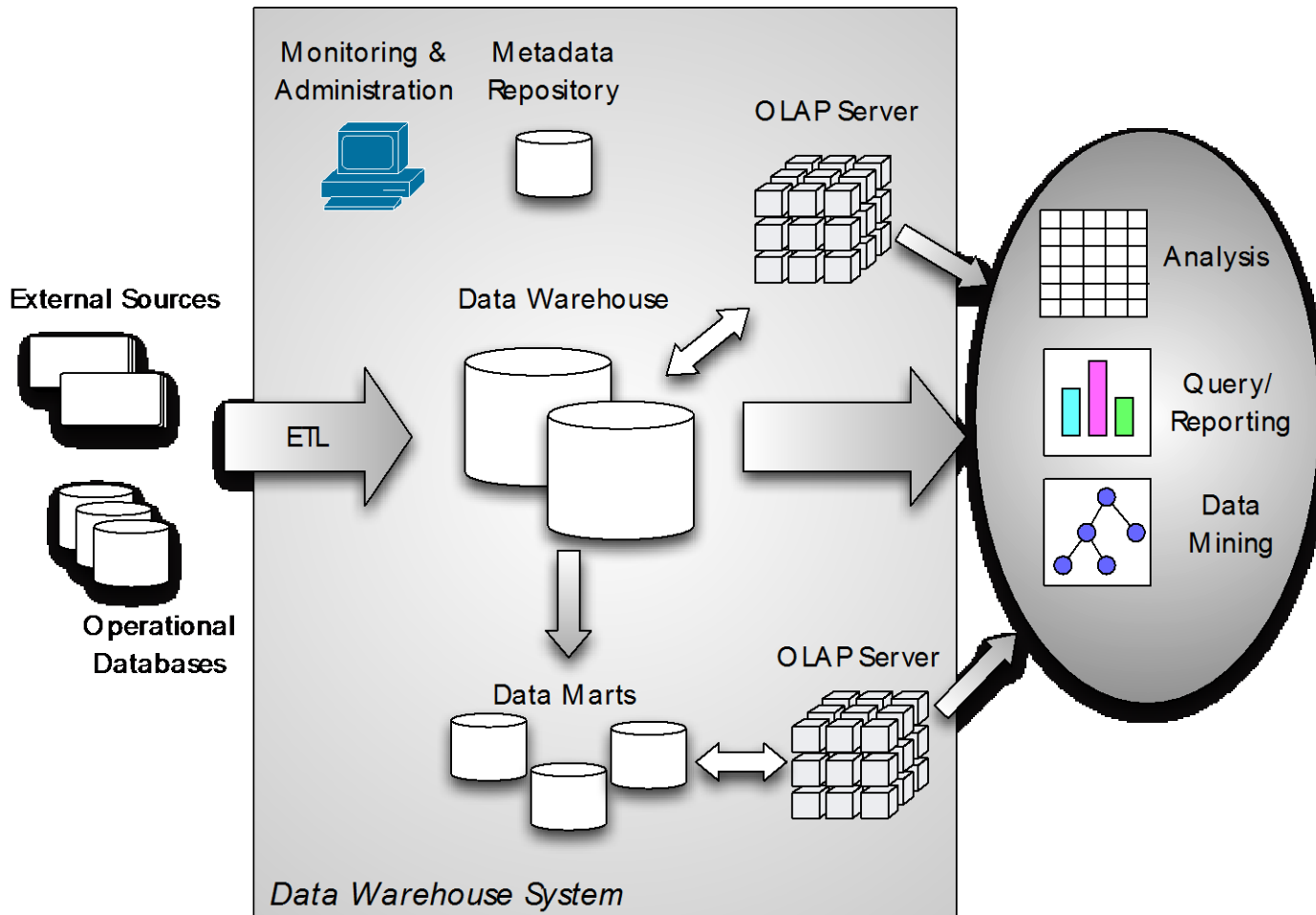
▶ Typical Questions:

- ▶ How many units of soft drinks did we sell last month?
- ▶ How was the sales trend of wine last year? Who are our gold customers?
- ▶ Which supplier delivers the largest quantities?
- ▶ Do we sell more beer in Ilmenau than in Erfurt?
- ▶ How many units of soft drinks did we sell last summer in Thuringia?
- ▶ More than bottled water?

▶ Problems

- ▶ Requires data from external sources (suppliers, customers)
- ▶ Data is time-related
- ▶ Querying multiple databases

Overview



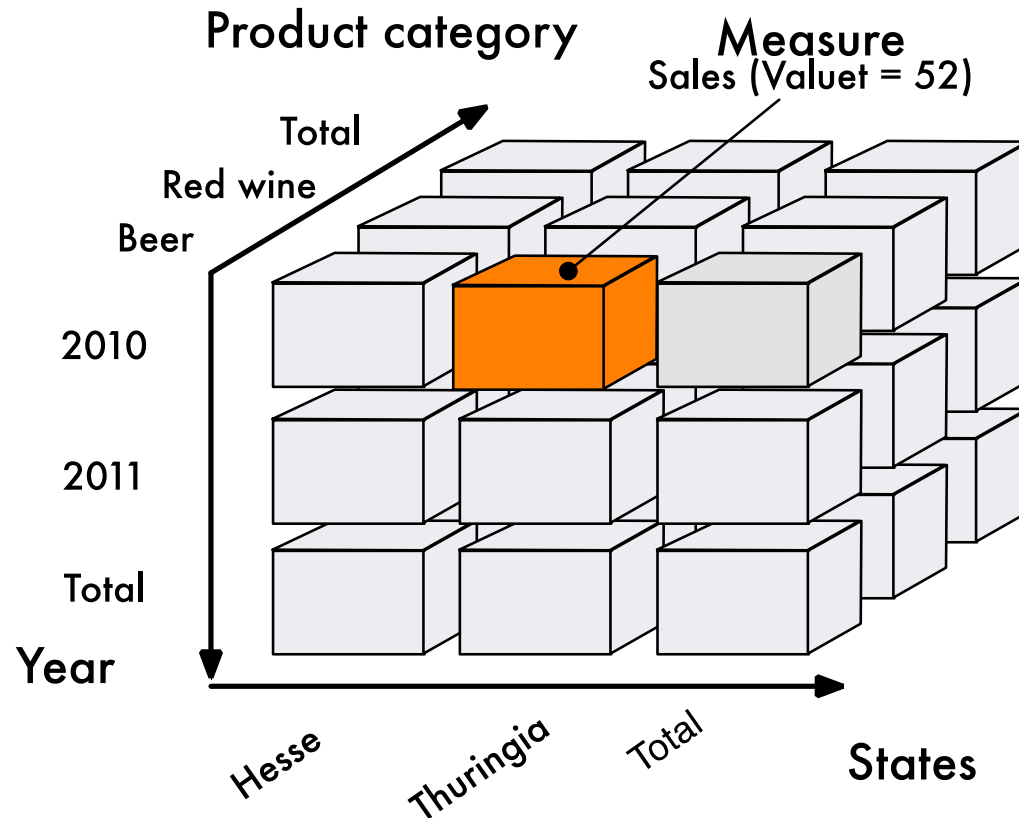
Application Example

- ▶ Wal-Mart (www.wal-mart.com): market leader in retail
- ▶ Enterprise-wide data warehouse
 - ▶ Database size: approx. 0,5 PB
 - ▶ Each day up to 20.000 queries
 - ▶ Very detailed data (daily analysis of sales, inventory, customer behaviour)
 - ▶ Foundation for market basket analysis, customer classification, ...
- ▶ Analysis questions
 - ▶ Checking assortment of goods (slow sellers, big seller, ...)
 - ▶ Analysing profitability of stores
 - ▶ Impact of marketing activities
 - ▶ Evaluating customer surveys and complaints
 - ▶ Inventory analysis
 - ▶ market basket analysis using sales slips

Example: Query

How many **units** did we sell in the **years** 2010 and 2011 in the **product categories** beer and red wines in the **states** Thuringia and Hesse?

Example: Query Result (Cube)



Example: Report (two-dimensional)

Sales		Beer	Red Wine	Total
2010	Hesse	45	32	77
	Thuringia	52	21	73
	Total	97	53	150
2011	Hesse	60	37	97
	Thuringia	58	20	78
	Total	118	57	175

OLTP vs. OLAP

- ▶ Conventional operational information systems
 - **Online Transactional Processing (OLTP)**
 - ▶ Collecting and management of data
 - ▶ Processing under the responsibility of each department
 - ▶ Transactions: short read / write access to few data records
- ▶ Data Warehouse
 - **Online Analytical Processing (OLAP)**
 - ▶ Focus: analysis
 - ▶ Transactions: long(-lasting) read transactions on many records
 - ▶ Integration, consolidation and aggregation of data

Example: OLTP vs. OLAP

```
SELECT CNo , CName , Account
FROM Customers C , Orders O
WHERE C.CNo = O.CNo
AND Product = 'Coffee'
```

```
SELECT store.name, year.description, SUM(quantities)
FROM sales, store, article, productgroup, day, month, year
WHERE sales.product_id = article.product_id           // Product-Dimension
AND article.group_id = productgroup.group_id
AND productgroup.description = 'soft drink'
AND sales.time_id = day.time_id                       // Time-Dimension
AND day.month_id = month.month_id
AND month.year_id = year.year_id
AND sales.store_id = store.store_id                   // Region-Dimension
GROUP BY store.name, year.description
```

Definition: Data Warehouse

A Data Warehouse is a **subject-oriented**, **integrated**, **non-volatile**, and **time variant** collection of data in support of managements decisions. (W.H. Inmon 1996)

- ▶ **subject-oriented**
 - ▶ Data is organized in a way that all information relating to the same real-world event or object are linked together
- ▶ **integrated data collection**
 - ▶ Combining data from different sources of all of an organization's operational systems (internal and external)
- ▶ **non-volatile data collection**
 - ▶ stable, persistent database
 - ▶ data is never removed or updated in DW
- ▶ **time-variant data**
 - ▶ allows to compare data over time (time series analysis)
 - ▶ storing data over a long period

More Definitions

- ▶ **Data Warehousing**

- ▶ Data Warehouse process, i.e. all steps of collecting & integrating data (extraction, transformation, loading) as well as storing and analysing

- ▶ **Data Mart**

- ▶ external (partial) view on the Data Warehouse
- ▶ by replicating or copying data
- ▶ user- or application-specific

- ▶ **OLAP (Online Analytical Processing)**

- ▶ explorative and interactive analysis based on the conceptual data model (cube)

- ▶ **Business Intelligence**

- ▶ Data Warehousing + Analytics (OLAP, Data Mining), Reporting,

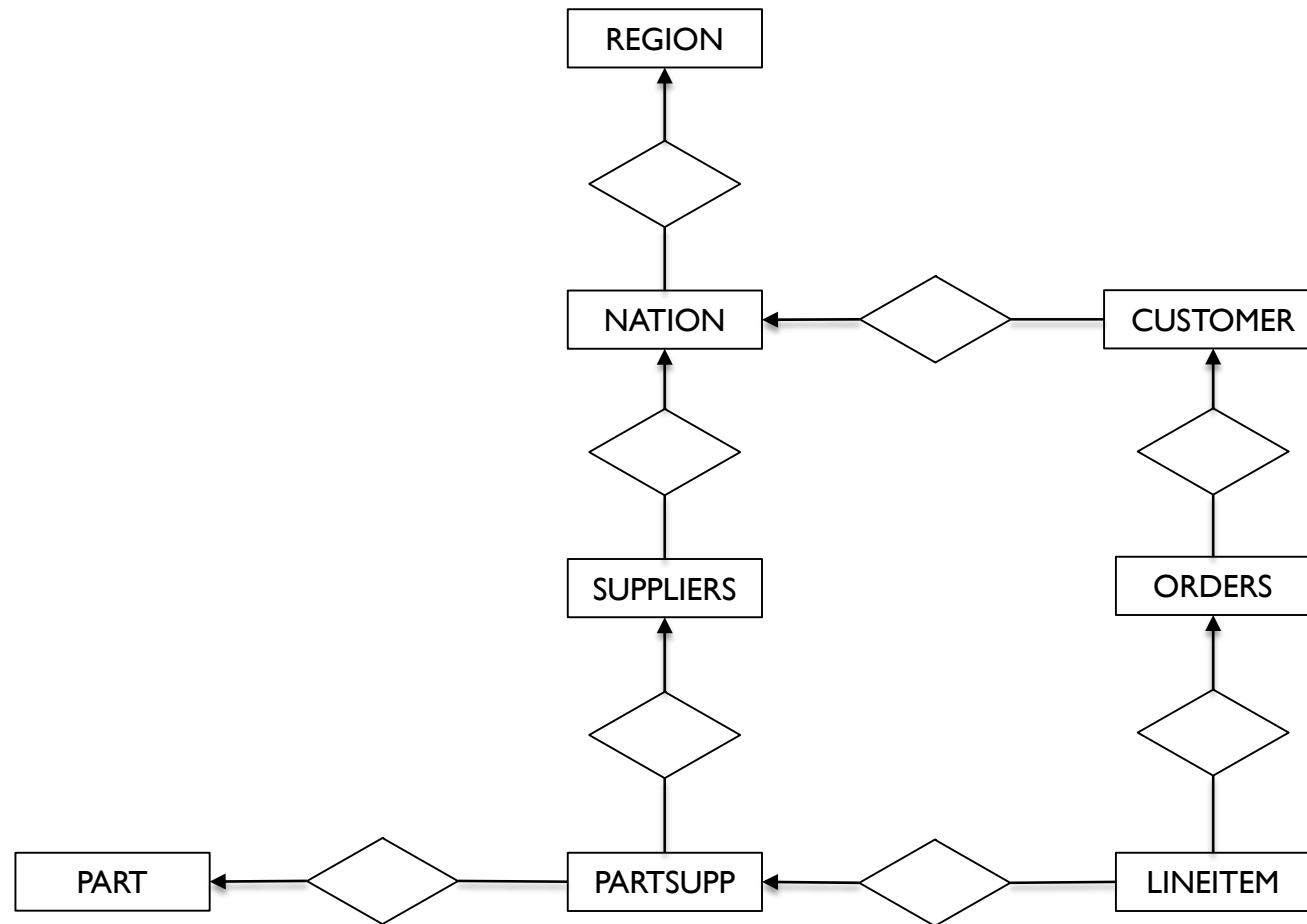
Separating Operational and Analytical Systems

- ▶ **Query response time:**
 - ▶ performing analytics on operational data source would result in poor performance
- ▶ **Time variant storage:**
 - ▶ time series analysis
- ▶ **Accessing data independently from operational data sources:**
 - ▶ availability, data integration
- ▶ **Normalizing/standardizing data formats in DW**
- ▶ **Guaranteeing data quality in DW**

TPC

- ▶ **Transaction Processing Performance Council (TPC)**
 - ▶ is a non-profit organization
 - ▶ founded in 1988
 - ▶ to define transaction processing and database benchmarks and to disseminate objective, verifiable TPC performance data to the industry
- ▶ TPC benchmarks are used in evaluating the performance of computer systems; the results are published on the TPC web site.

Benchmarking: TPC-H Schema




TPC-H: Example Query

▶ 22 queries

▶ Large Volume Customer Query (Q18)

- ▶ The Large Volume Customer Query ranks customers based on their having placed a large quantity order. Large quantity orders are defined as those orders whose total quantity is above a certain level

```
SELECT  c_name, c_custkey, o_orderkey, o_orderdate, o_totalprice,
        SUM(l_quantity)
FROM    customer , orders , lineitem
WHERE   o_orderkey IN ( SELECT l_orderkey
                        FROM lineitem
                        GROUP BY l_orderkey
                        HAVING SUM(l_quantity) > [QUANTITY] )
AND     c_custkey = o_custkey
AND     o_orderkey = l_orderkey
GROUP BY c_name, c_custkey, o_orderkey, o_orderdate, o_totalprice
ORDER BY o_totalprice desc, o_orderdate
```



TPC-H: Some Numbers

10,000 GB Results

Rank	Company	System	QphH	Price/QphH	Watts/KQphH	System Availability	Database	Operating System	Date Submitted	Cluster
1		Dell PowerEdge R710 using EXASolution 4.0	7,128,255	.53 USD	NR	10/01/11	EXASOL EXASolution 4.0	EXASOL EXACluster OS 4.0	04/05/11	Y
2		IBM System p 570	343,551	32.89 USD	NR	04/15/08	IBM DB2 Warehouse 9.5	IBM AIX 5L V5.3	10/15/07	Y
3		HP Integrity Superdome/Dual-Core Itanium/1.6 GHz	208,457	27.97 USD	NR	09/10/08	Oracle Database 11g Enterprise Edition	HP-UX 11.i v3 64 bit	03/10/08	N
4		IBM System p5 575 with DB2 UDB 8.2	180,108	47.00 USD	NR	08/30/06	IBM DB2 UDB 8.2	IBM AIX 5L V5.3	07/14/06	Y
5		HP Integrity Superdome-DC Itanium2/1.6GHz /64p/128c	171,380	32.91 USD	NR	04/01/07	Oracle Database 10g R2 Enterprise Edt w/Partitioning	HP-UX 11i v3 64 bit	11/30/06	N
6		HP Integrity Superdome - Itanium2/1.5 GHz-128p/128	86,282	161.24 USD	NR	04/06/05	Oracle Database 10g Enterprise Edition	HP UX 11.i V2 64 bit	10/07/04	Y
7		Unisys ES7000 Model 7600R Enterprise Server(16s)	80,172	18.95 USD	NR	02/17/09	Microsoft SQL Server 2008 Enterprise x64 Edition	Microsoft Windows Server 2008 Datacenter x64 Edition	02/17/09	N
8		HP Integrity Superdome	63,650	38.54 USD	NR	08/30/08	Microsoft SQL Server 2008 Enterprise Edition	Microsoft Windows Server 2008 Itanium based Systems	02/27/08	N
9		HP Integrity Superdome - Itanium2/1.5 GHz-64p/64c	49,104	118.13 USD	NR	03/25/04	Oracle Database 10g Enterprise Edition	HP-UX 11.i 64-bit Base OS	01/05/04	N

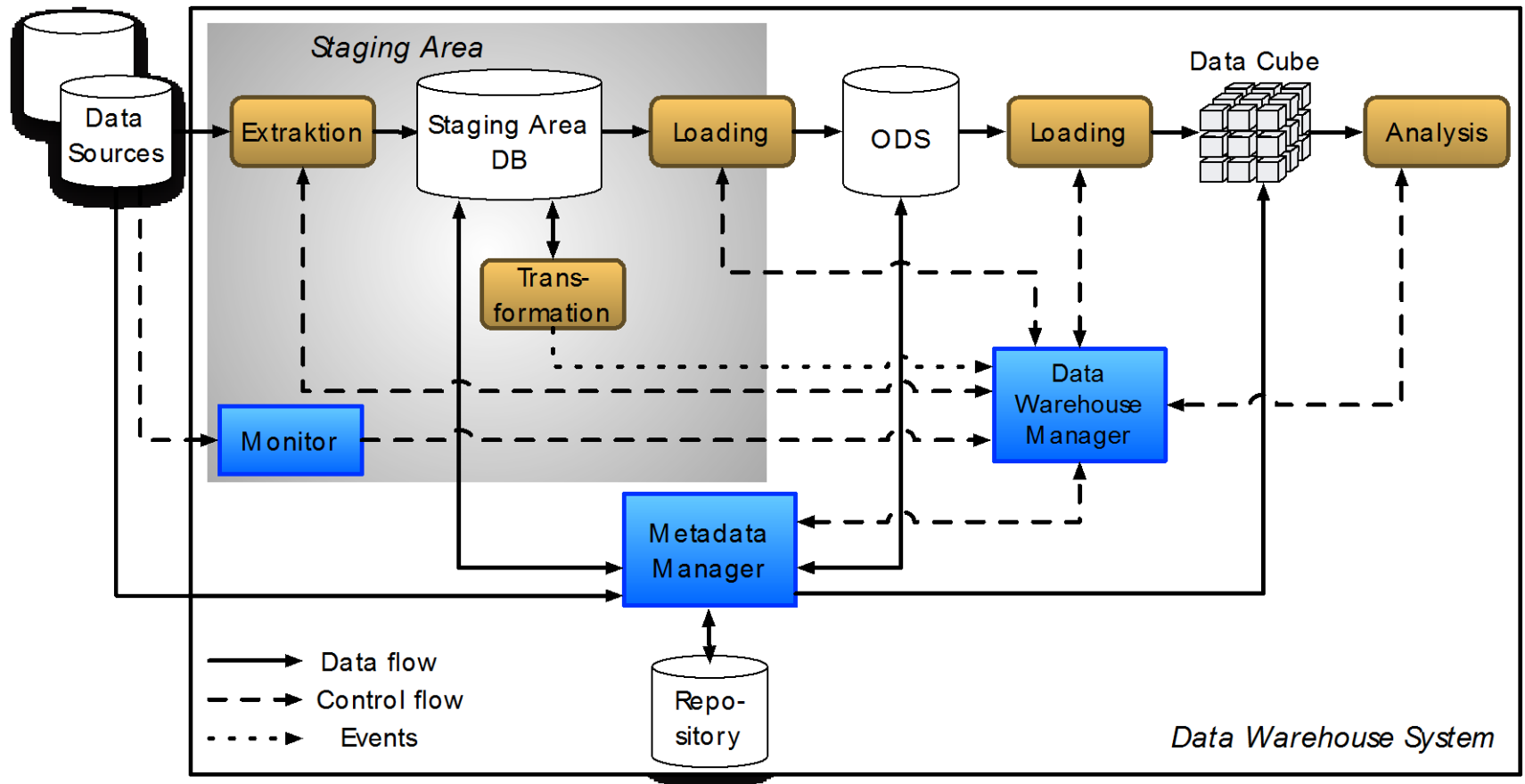
DW Market

- ▶ OLAP tools/servers
 - ▶ MS Analysis Services, Hyperion, Cognos, Pentaho Mondrian
- ▶ DW extensions for RDBMS
 - ▶ Oracle I I g, IBM DB2, MS SQL Server:
 - ▶ SQL extensions, index structures
 - ▶ Materialized views, bulk loading
- ▶ ETL (*E*xtract, *T*ransform, *L*oad) tools
 - ▶ MS Integration Services, Oracle Warehouse Builder, Kettle

Data Warehousing: Requirements

- ▶ Physical separation of data sources and analytics system (due to availability, load, updates and changes)
- ▶ Providing integrated, consistent, derived data in a persistent way
- ▶ Multiple uses of provided data
- ▶ Allowing in principle arbitrary analysis tasks
- ▶ Supporting individual views (e.g. time horizon, structure, ...)
- ▶ Extensibility (e.g. integrating new sources)
- ▶ Automation of processes (ETL, reporting, ...)
- ▶ Clear definition of data structures, roles, access authorities, processes
- ▶ Subject orientation: analysis of data

Reference Architecture



Data Warehousing: Steps

Steps:

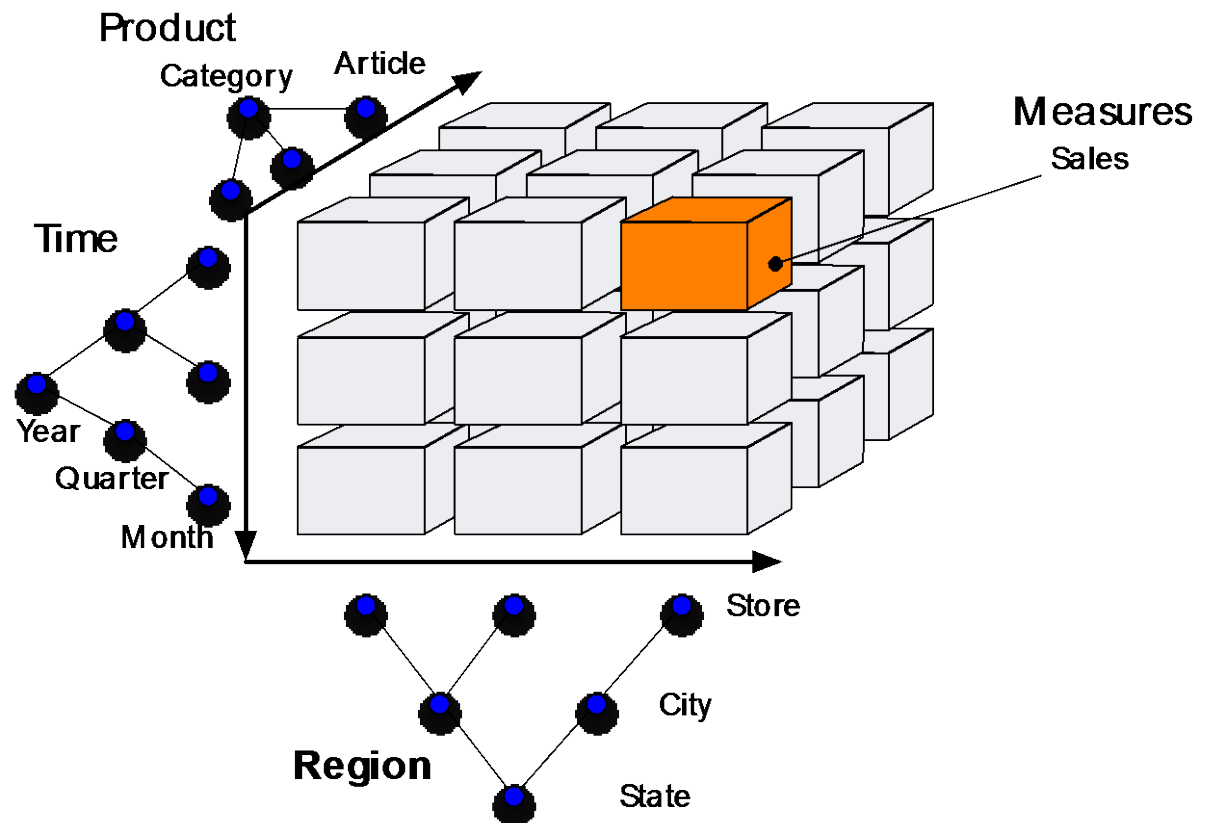
- ▶ Monitoring data sources for detecting data changes
- ▶ Extracting relevant data into a temporary working area (staging area)
- ▶ Transforming data in staging area (cleaning, integration)
- ▶ Loading data into an integrated data store (operational data store) as foundation for different analysis tasks
- ▶ Loading data into data warehouse (database for analysis)
- ▶ Analysis: Queries and operations of data in DW

Multidimensional Data Model

- ▶ Data model supporting multi-dimensional analytics
- ▶ data analysis in decision-support processes
- ▶ **Facts**
 - ▶ (= numerical measures related to the organization's business processes)
 - ▶ such as sales, revenue, loss are at the center of consideration
- ▶ **Dimensions**
 - ▶ Different views on facts:
 - ▶ e.g. related to time, geographic region, products
- ▶ **Hierarchies or consolidation paths**
 - ▶ Allows to have different levels of analysis in dimensions (e.g. year, quarter, month)

Basic Concepts

- ▶ Dimensions
- ▶ Facts and Measures

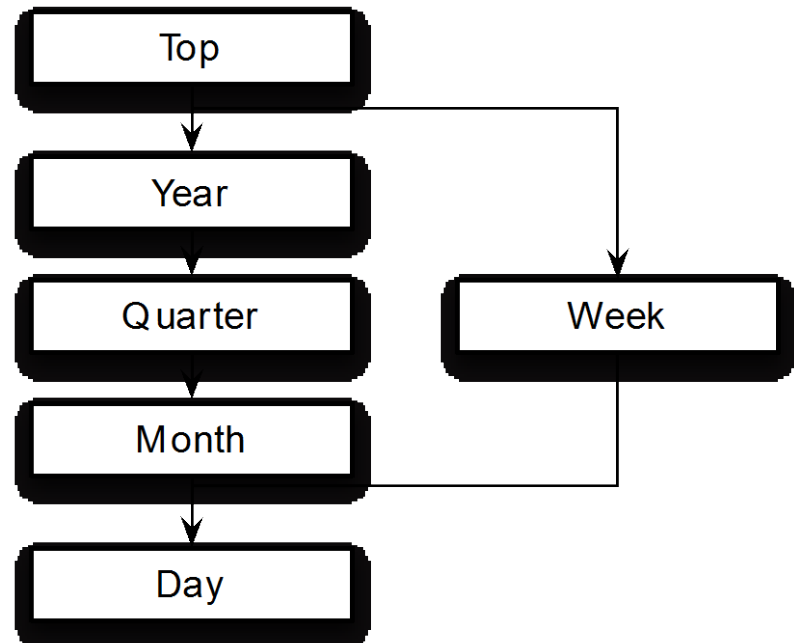
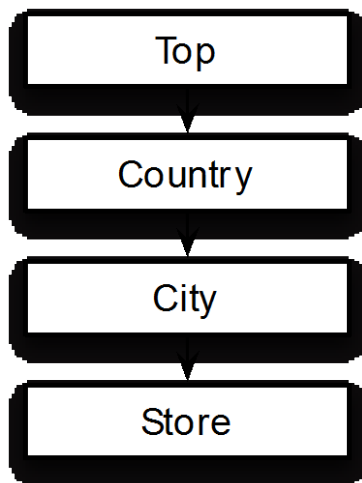


Dimensions

- ▶ Categorize measures, i.e. provide a certain view
- ▶ Used for orthogonal structuring of the data space
- ▶ Dimension: finite set of $n \geq 2$ dimension elements with a semantic relationship
- ▶ **Example:**
 - ▶ product categorization, geographical region, time
- ▶ Dimensions can be organized hierarchically
 - ▶ A level contains aggregated values of the following level
 - ▶ Highest level Top_D = single aggregated value of the dimension
- ▶ Dimension element:
 - ▶ Node in a classification hierarchy
 - ▶ Level of classification represents level of detail or aggregation
- ▶ Representing a dimension by a classification schema

Hierarchies in Dimensions

- ▶ Simple hierarchies vs. Parallel hierarchies
- ▶ Parallel hierarchy
 - ▶ Multiple independent paths
 - ▶ No relationships between parallel paths



Dimension Schema

- ▶ Partially ordered set of categorical attributes

$$(\{D_1, \dots, D_n, Top_D\}; \rightarrow)$$

- ▶ Generic element Top_D
- ▶ Functional dependency \rightarrow

- ▶ Top_D depends on all other attributes

$$\forall i, 1 \leq i \leq n : D_i \rightarrow Top_D$$

- ▶ There is exactly one D_i which determines all other categorical attributes

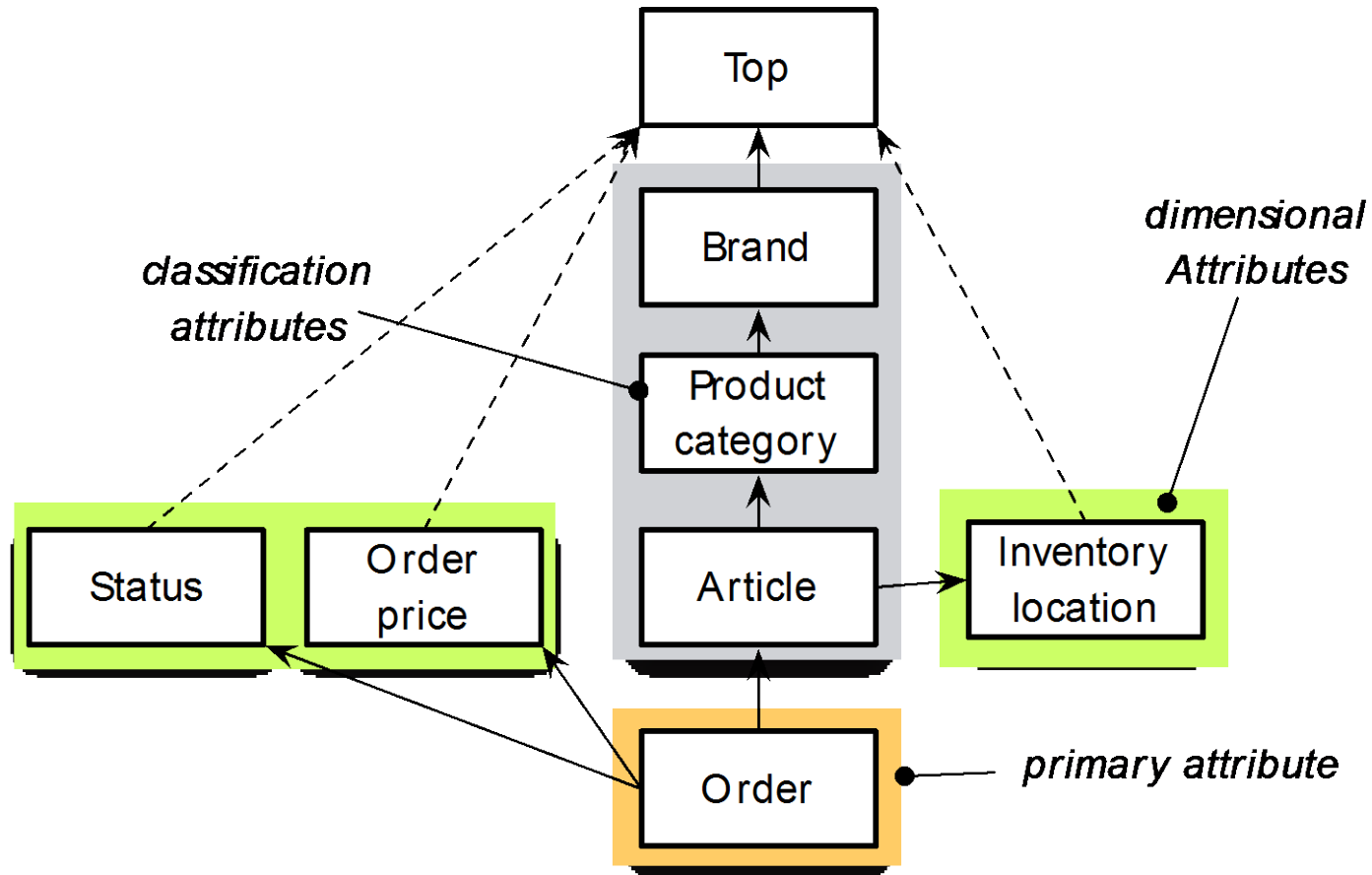
Defines the highest level of detail (smallest granularity) of a dimension

$$\exists i, 1 \leq i \leq n, \forall j, 1 \leq j \leq n, i \neq j : D_i \rightarrow D_j$$

Categorical Attributes

- ▶ Different roles of attributes
- ▶ **Primary attribute**
 - ▶ Attribute which determines all other attributes of the dimension
 - ▶ Defines highest level of detail
 - ▶ **Example:** Order, Day, Product
- ▶ **Classification attributes**
 - ▶ Set of attributes forming a multi-level hierarchy
 - ▶ **Example:** Customer, Nation, Region, product category
- ▶ **Dimensional Attributes**
 - ▶ Set of attributes which depend on the primary attribute or on a classification attribute and only determine Top_D
 - ▶ **Example:** Address, Phone Number

Structure of a Dimension



Facts and Measures

- ▶ **Facts / measures**
 - ▶ Aggregated numerical values
 - ▶ Representing facts about a managed entity or system
- ▶ **Facts:**
 - ▶ Explicitly stored in the data warehouse
- ▶ **Measures**
 - ▶ Calculated from facts (or other measures)
 - ▶ by applying arithmetic functions
 - ▶ **Examples:** Sales, Revenue, Loss

Measures: Schema

- ▶ Schema comprises several components

- ▶ granularity $G = \{G_1, \dots, G_k\}$

- ▶ G is a subset of categorical attributes of all in the schema existing dimension schemas DS_1, \dots, DS_n

$$\forall i, 1 \leq i \leq k, \exists j, 1 \leq j \leq n : G_i \in DS_j$$

- ▶ No functional dependencies between categorical attributes of a given granularity

$$\forall i, 1 \leq i \leq k, \forall j, 1 \leq j \leq k, i \neq j : G_i \not\rightarrow G_j$$

- ▶ Calculation (level of detail of facts)

- ▶ Measure type

Measures: Calculation

- ▶ **Scalar functions**

- ▶ +, -, *, /, mod
- ▶ **Example:** sales tax = quantity * price * tax rate

- ▶ **Aggregate functions**

- ▶ Function $H()$ for consolidating a data set by aggregating n values into a single value

$$H : 2^{dom(X_1) \times \dots \times dom(X_n)} \rightarrow dom(Y)$$

- ▶ SUM(), AVG(), MIN(), MAX(), COUNT()

- ▶ **Order-based functions**

- ▶ Definition of measures based on a specified order
- ▶ **Examples:** cumulative sum, TOP(n)

Measure Type

- ▶ Measure type characterizes aggregation operations applicable to the fact
- ▶ FLOW
 - ▶ Can be aggregated in any dimension
 - ▶ **Example:** order quantity of a certain article per day
- ▶ STOCK
 - ▶ Can be aggregated in any dimension except temporal dimension
 - ▶ **Example:** stock of inventory, population per city
- ▶ VALUE-PER-UNIT (VPU)
 - ▶ Measures representing state, which cannot be added across any dimension
 - ▶ Usable only: MIN(), MAX(), AVG()
 - ▶ **Example:** currency rate, tax rate

Cube

- ▶ fundamental concept of multi-dimensional analysis
- ▶ multiple dimensions
- ▶ Cell = one or more facts/measures (function of dimensions)
- ▶ Dimensionality = number of dimensions
- ▶ Visualization
 - ▶ 2 dimensions = table
 - ▶ 3 dimensions = cube
 - ▶ >3 dimensions = hypercube = multi-dimensional domain structure
- ▶ Schema C of a cube
 - ▶ Set of dimension (schemas) DS
 - ▶ Set of measures M
$$C = (DS, M) = (\{D^1, \dots, D^n\}, \{M^1, \dots, M^m\})$$
- ▶ Orthogonality
 - ▶ There are no functional dependencies between attributes from different dimensions

ME/R: A Conceptual Model for DW

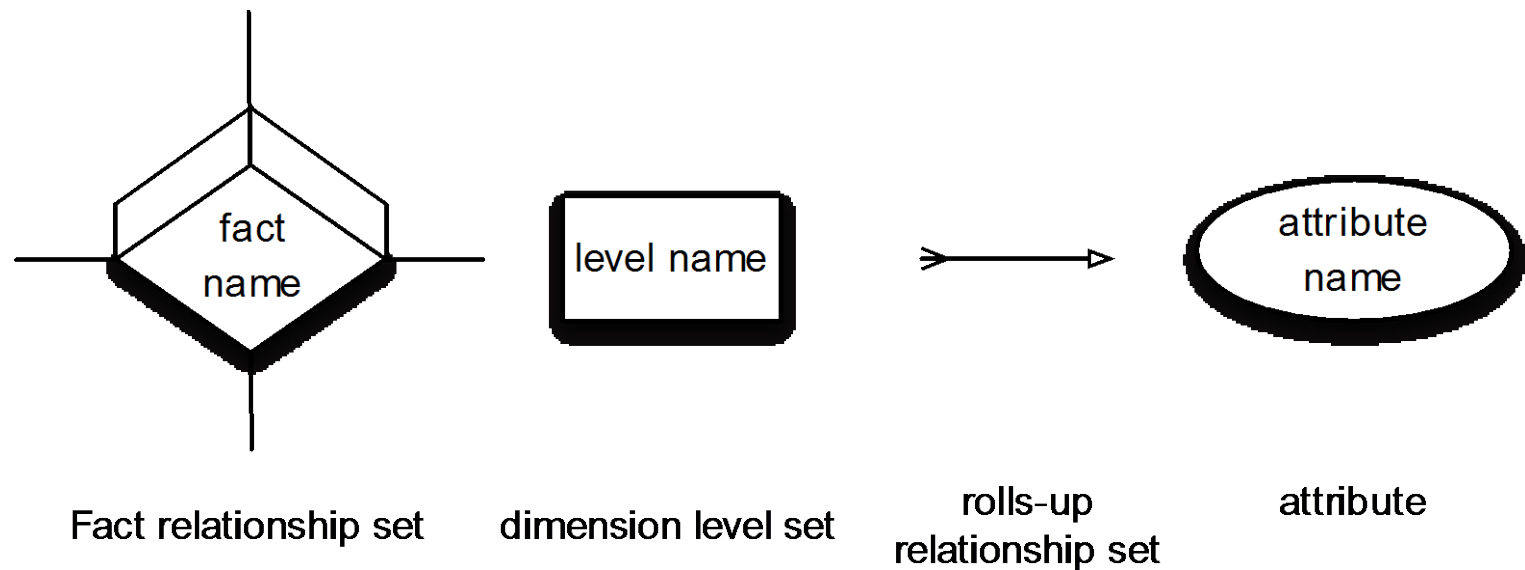
▶ Multidimensional Entity/Relationship Model

[Sapia et. al. , Lecture Notes in Computer Science, Springer (LNCS 1552)]

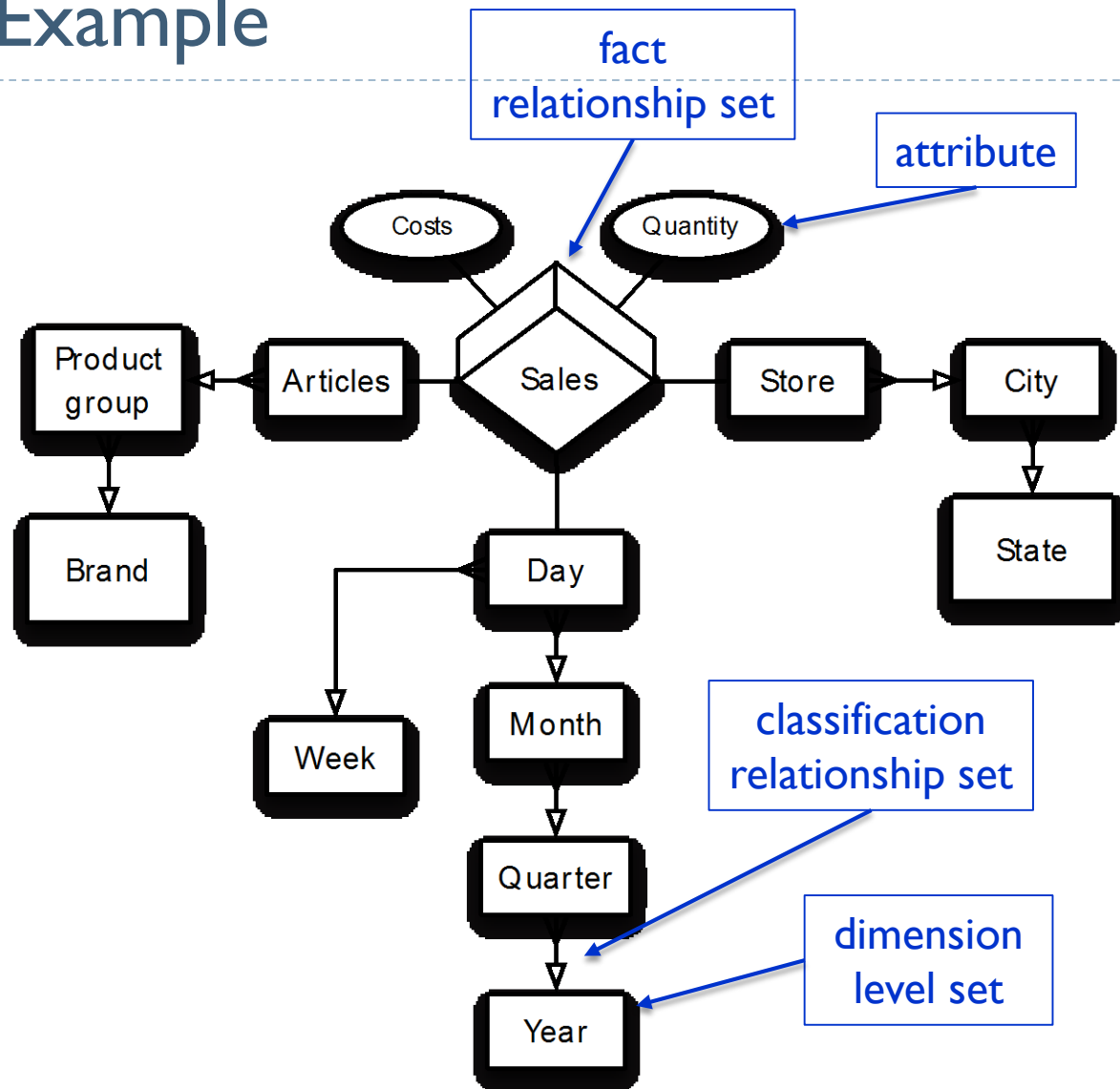
▶ Extends the classic ER model by

- ▶ Entity set „dimension level“
 - ▶ But no explicit modeling of dimensions
- ▶ N-ary relationship set „fact“
 - ▶ Measures as attributes of this relationship
- ▶ Binary relationship set „classification“ or „roll-up“ (connects dimension levels)
 - ▶ Defines directed, acyclic graph

ME/R: Notation



ME/R: Example



Mapping the Multidimensional Data Model

- ▶ **Multidimensional view**
 - ▶ Modelling of data
 - ▶ Query formulation
- ▶ **Internal representation of data requires mapping to**
 - ▶ Relational structures (tables) → **ROLAP** (relational OLAP)
 - ▶ Availability, matured systems
 - ▶ Multidimensional structures (direct storage) → MOLAP (multidimensional OLAP)
 - ▶ Omission of transformation
- ▶ **Issues**
 - ▶ Storage
 - ▶ Query formulation and execution

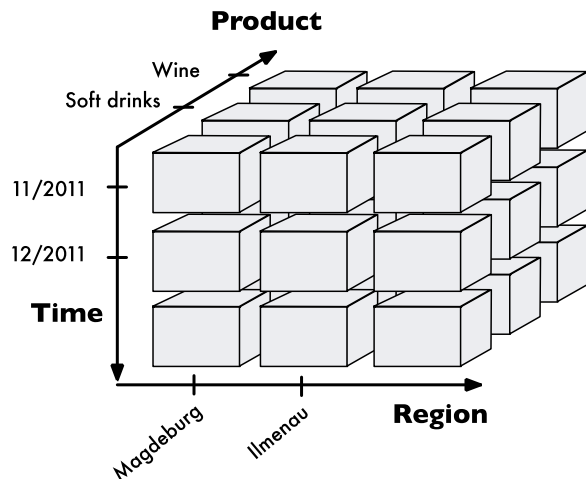
Relational Mapping

Consider:

- ▶ Avoid losing application-related semantics from the multidimensional model (e.g. classification hierarchies)
- ▶ Efficient rewriting of multidimensional queries
- ▶ Efficient processing of queries
- ▶ Easy maintenance of tables (e.g. loading new data)
- ▶ Taking characteristics as well as volume of data of analytical applications into account

Relational Mapping: Fact Table

- ▶ **first** step: mapping the cube without considering classification hierarchies
 - ▶ Dimensions, facts → columns of the relation
 - ▶ Cell → row

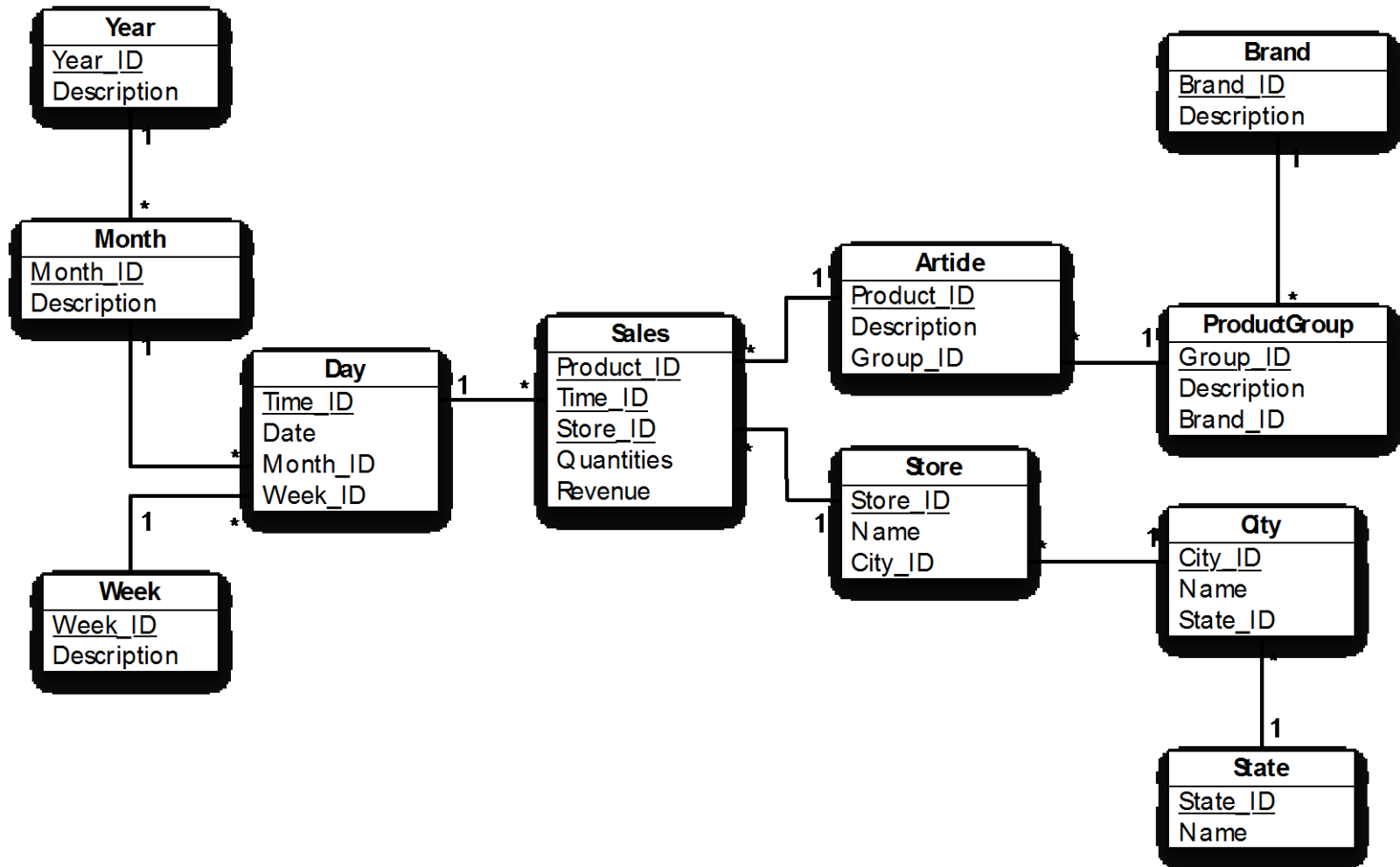


Product	Region	Month	Sales
Soft drink	Ilmenau	11/2011	145
Wine	Ilmenau	11/2011	98
Soft drink	Magdeburg	11/2011	245
...			

Snowflake Schema

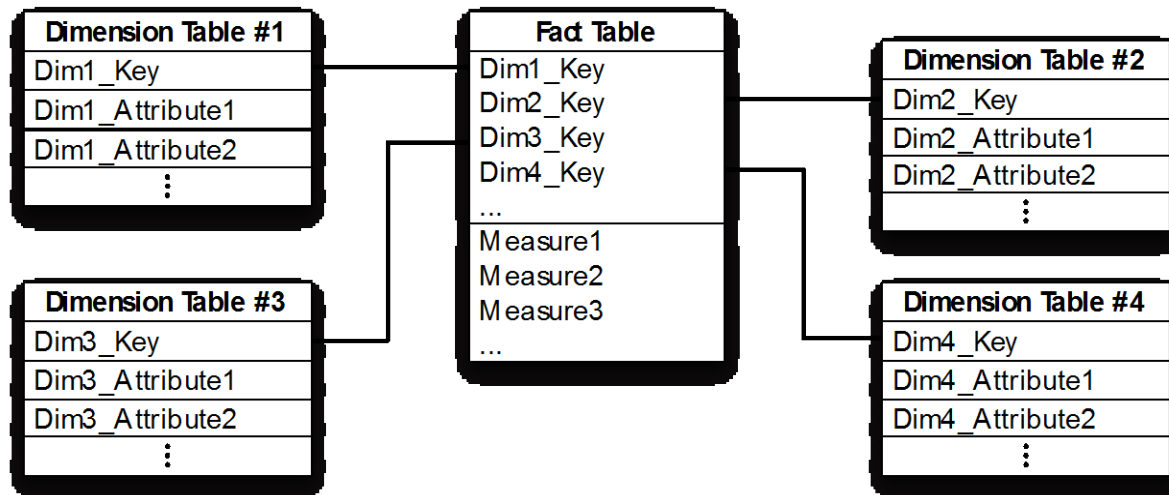
- ▶ Mapping of classification hierarchies: a separate table for each classification level (e.g. article, product group etc.)
- ▶ Dimension tables contain
 - ▶ ID column for classification level
 - ▶ Dimensional attributes (e.g. brand, supplier address, description)
 - ▶ Foreign key of the parent level
- ▶ Fact table contains (in addition to measures):
 - ▶ Foreign key of the lowest level of each dimension
 - ▶ Combination of all foreign keys represent the composite primary key of the fact table

Snowflake Schema: Example

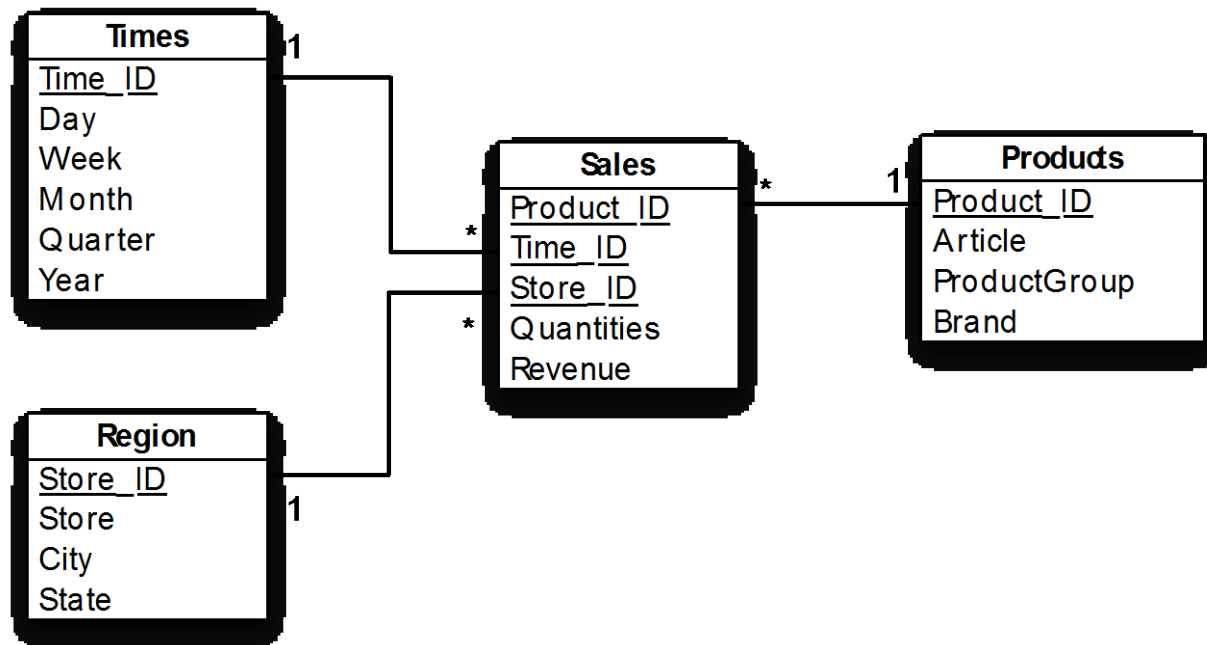


Star Schema

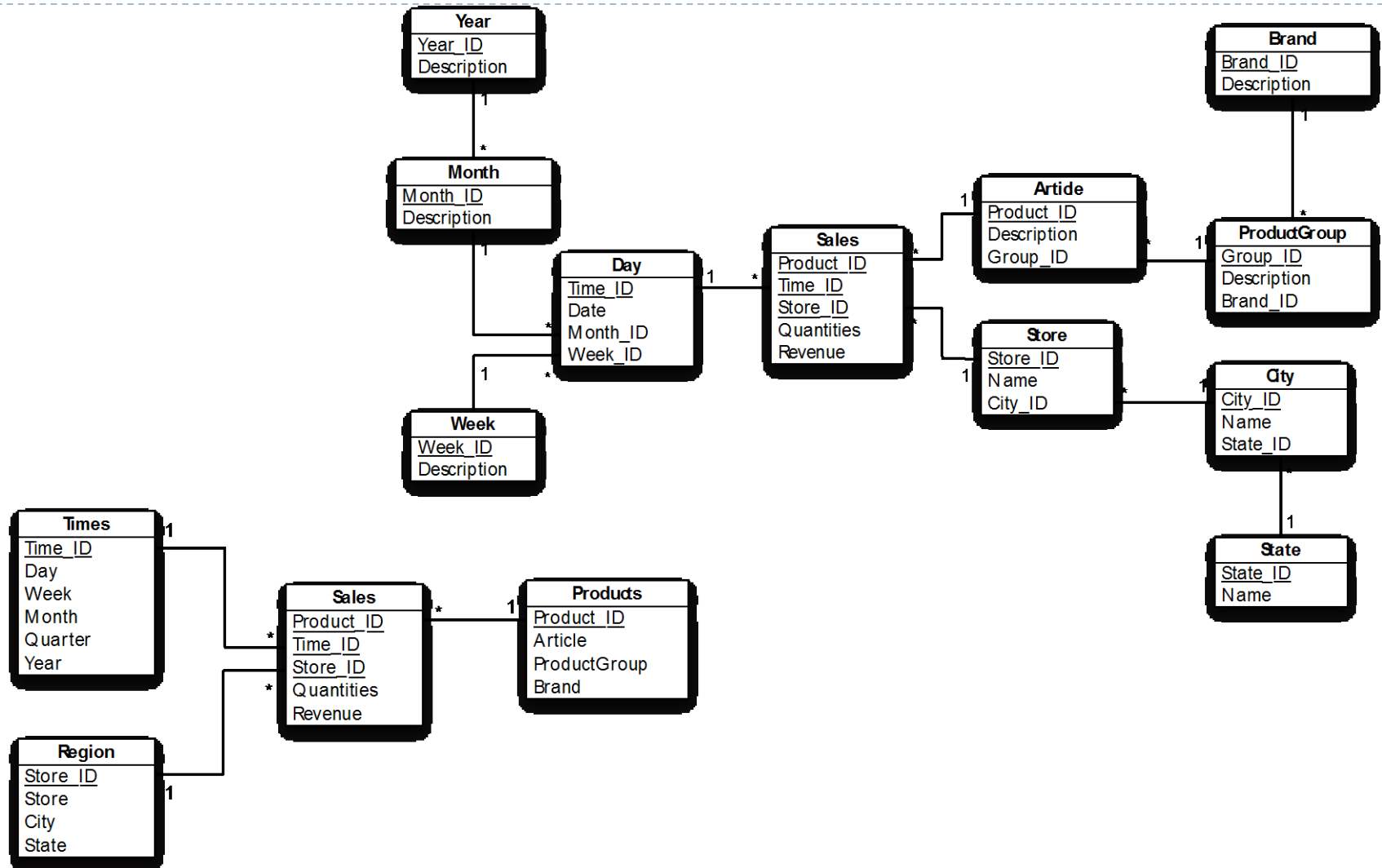
- ▶ Snowflake schema is a normalized schema: avoids update anomalies
 - ▶ But: requires joins between several tables
- ▶ Star Schema:
 - ▶ Denormalization of tables representing a dimension
 - ▶ each dimension is represented by a single dimension table
 - ▶ Introduces redundancies for better query performance



Star Schema: Example



Star vs. Snowflake Schema



Queries Star vs. Snowflake Schema

- ▶ Sales of soft drinks per store and year
- ▶ For **snowflake** schema

```
SELECT store.name, year.description, SUM(quantities)
FROM sales, store, article, productgroup, day, month, year
WHERE sales.product_id = article.product_id           // Product-Dimension
      AND article.group_id = productgroup.group_id
      AND productgroup.description = 'soft drink'
      AND sales.time_id = day.time_id                 // Time-Dimension
      AND day.month_id = month.month_id
      AND month.year_id = year.year_id
      AND sales.store_id = store.store_id             // Region-Dimension
GROUP BY store.name, year.description
```

- ▶ Number of joins: **6** (grows linearly with the number of aggregation paths)

Queries Star vs. Snowflake Schema

► For **star** schema

```
SELECT region.store, times.year, SUM(quantities)
FROM sales, region, products, time
WHERE sales.product_id = products.product_id      // Product-Dimension
      AND products.productgroup = 'soft drink'
      AND sales.time_id = times.time_id           // Time-Dimension
      AND sales.store_id = region.store_id        // Region-Dimension
GROUP BY region.store, times.year
```

- Number of joins: **3** (independently from the number of aggregation paths)

Star vs. Snowflake Schema

- ▶ **Characteristics of DW applications**
 - ▶ Typically, queries with restrictions at higher dimension levels (joins)
 - ▶ Small volumes of data in dimension tables compared to the fact table
 - ▶ Only rarely updates on classification (potential update anomalies)
- ▶ **Advantages of star schema**
 - ▶ Simple structure (simplified query formulation)
 - ▶ Simple and flexible representation of classification hierarchies (columns in dimension tables)
 - ▶ Efficient processing of queries inside a dimension (no join required)

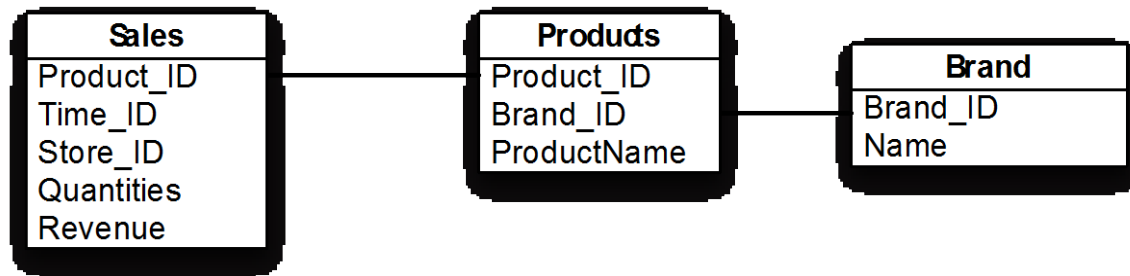
CREATE DIMENSION in Oracle

- ▶ Foreign key constraints are expressible in SQL
- ▶ But, it's not possible to specify functional dependencies between attributes of the same dimension
- ▶ Extension in Oracle: **CREATE DIMENSION**
 - ▶ „Non-compulsory“ constraints
 - ▶ Correctness is not ensured by the DBMS
 - ▶ Used only for query rewriting
- ▶ **Clauses**
 - ▶ **LEVEL**: defines classification levels
 - ▶ **HIERARCHY**: defines dependencies between classification levels
 - ▶ **ATTRIBUTE ... DETERMINES**: defines dependency between classification attribute and dimensional attributes

CREATE DIMENSION: Example

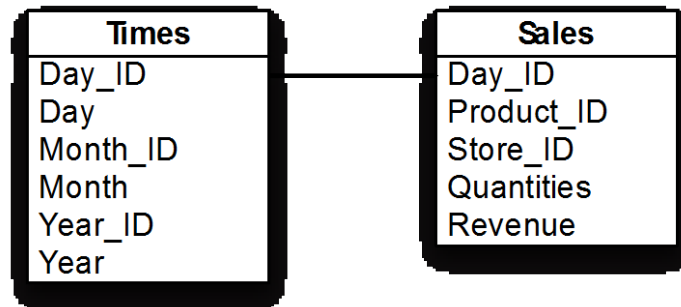
```
CREATE DIMENSION order_dim
  LEVEL order IS (orders.orderkey)
  LEVEL customer IS (orders.customerkey)
  LEVEL nation IS (orders.nationkey)
  LEVEL region IS (orders.regionkey)
  HIERARCHY order_rollup (
    order CHILD OF
    customer CHILD OF
    nation CHILD OF
    region)
  ATTRIBUTE order DETERMINES (o_status, o_date, ...)
  ATTRIBUTE customer DETERMINES (c_name, c_address, ...);
```

CREATE DIMENSION: Snowflake Schema



```
CREATE DIMENSION product_dim
  LEVEL product IS (products.product_id)
  LEVEL brand IS (brand.brand_id)
  HIERARCHY product_rollup (
    product CHILD OF brand)
  JOIN KEY (products.brand_id) REFERENCES brand.brand_id
```

CREATE DIMENSION: Star Schema



```
CREATE DIMENSION time_dim
  LEVEL day IS (times.day_id)
  LEVEL month IS (times.month_id)
  LEVEL year IS (times.year_id)
  HIERARCHY time_rollup (
    day CHILD OF month CHILD OF year)
  ATTRIBUTE day_id DETERMINES (day)
  ATTRIBUTE month_id DETERMINES (month)
  ATTRIBUTE year_id DETERMINES (year)
```

Conclusions

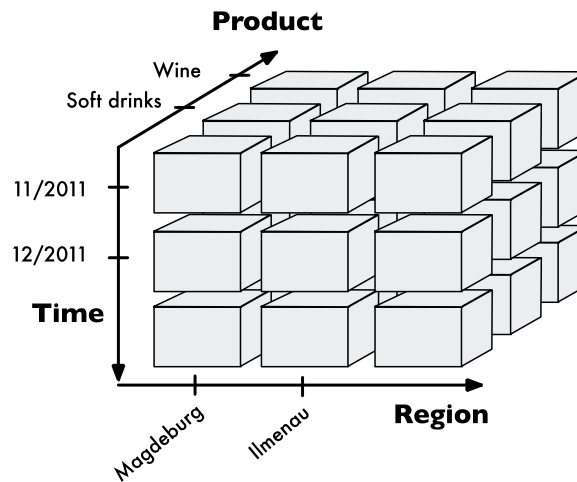
- ▶ Problems of relational mapping
 - ▶ Transformation of multidimensional queries into relational queries → complex queries
 - ▶ Complex query tools (OLAP tools)
 - ▶ Loss of semantics
- ▶ Loss of semantics caused by relational mapping:
 - ▶ Distinction between measures and dimensions (attributes of the fact table)
 - ▶ Attributes of dimension tables (dimensional attributes or classification attributes)
 - ▶ Structure of the dimension (consolidation paths)
- ▶ Solution:
 - ▶ Extending the system catalog with metadata describing multidimensional data
 - ▶ **Example:** CREATE DIMENSION, HIERARCHY in Oracle
 - ▶ direct multidimensional representation (MOLAP)

Querying the DW: OLAP Operations

- ▶ OLAP operations on multidimensional data structures (cube)
- ▶ Standard operations
 - ▶ Pivot
 - ▶ Pivoting or rotating of the cube
 - ▶ Roll-Up, Drill-Down
 - ▶ Navigation along the hierarchies
 - ▶ Drill-Across
 - ▶ Exchange between data cubes
 - ▶ Slice, Dice
 - ▶ Building subsets

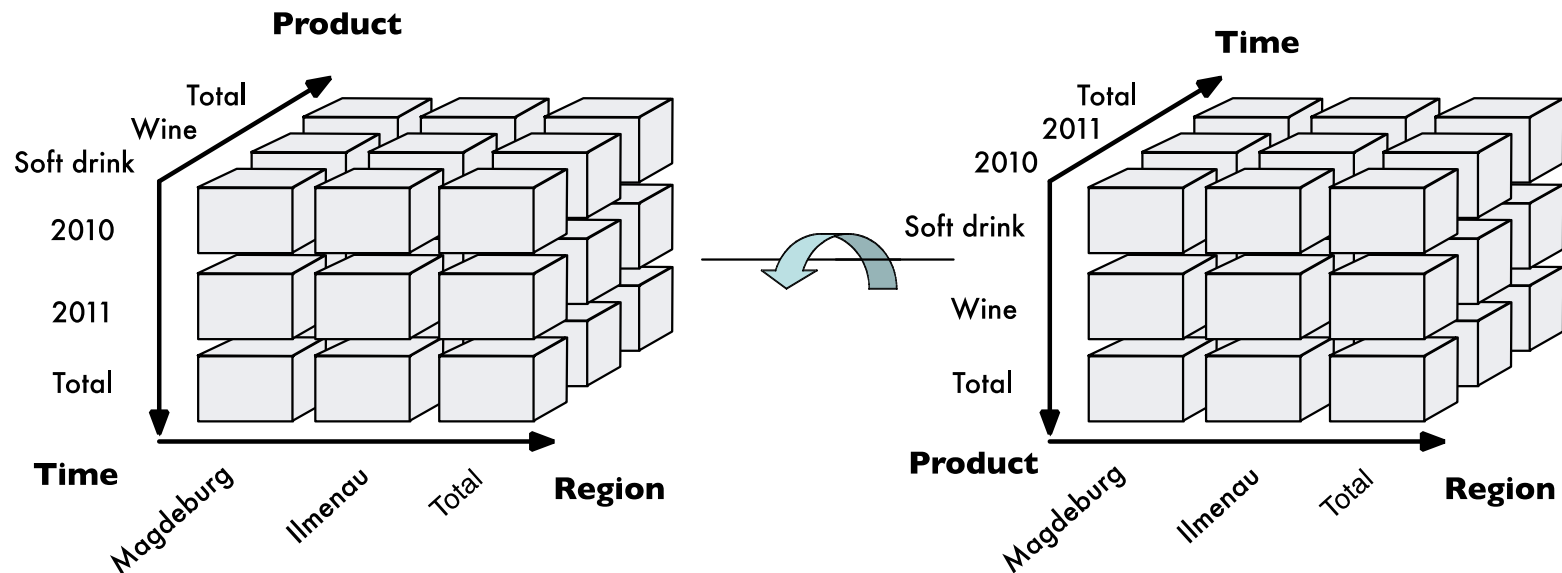
Querying the DW: Example

- ▶ How many units were sold in the product groups of soft drinks and wine in the states of Thuringia and Hesse per month and place in the years 2010 and 2011 and which turnover (total revenue) has been achieved?



Pivot / Rotate

- ▶ Rotate the cube by exchanging dimensions
- ▶ Analyse data from different perspectives



Roll Up and Drill Down

▶ Roll-Up:

- ▶ Derives new information by aggregating data along the classification path
- ▶ Dimensionality remains constant
- ▶ **Example:** day → month → quarter → year

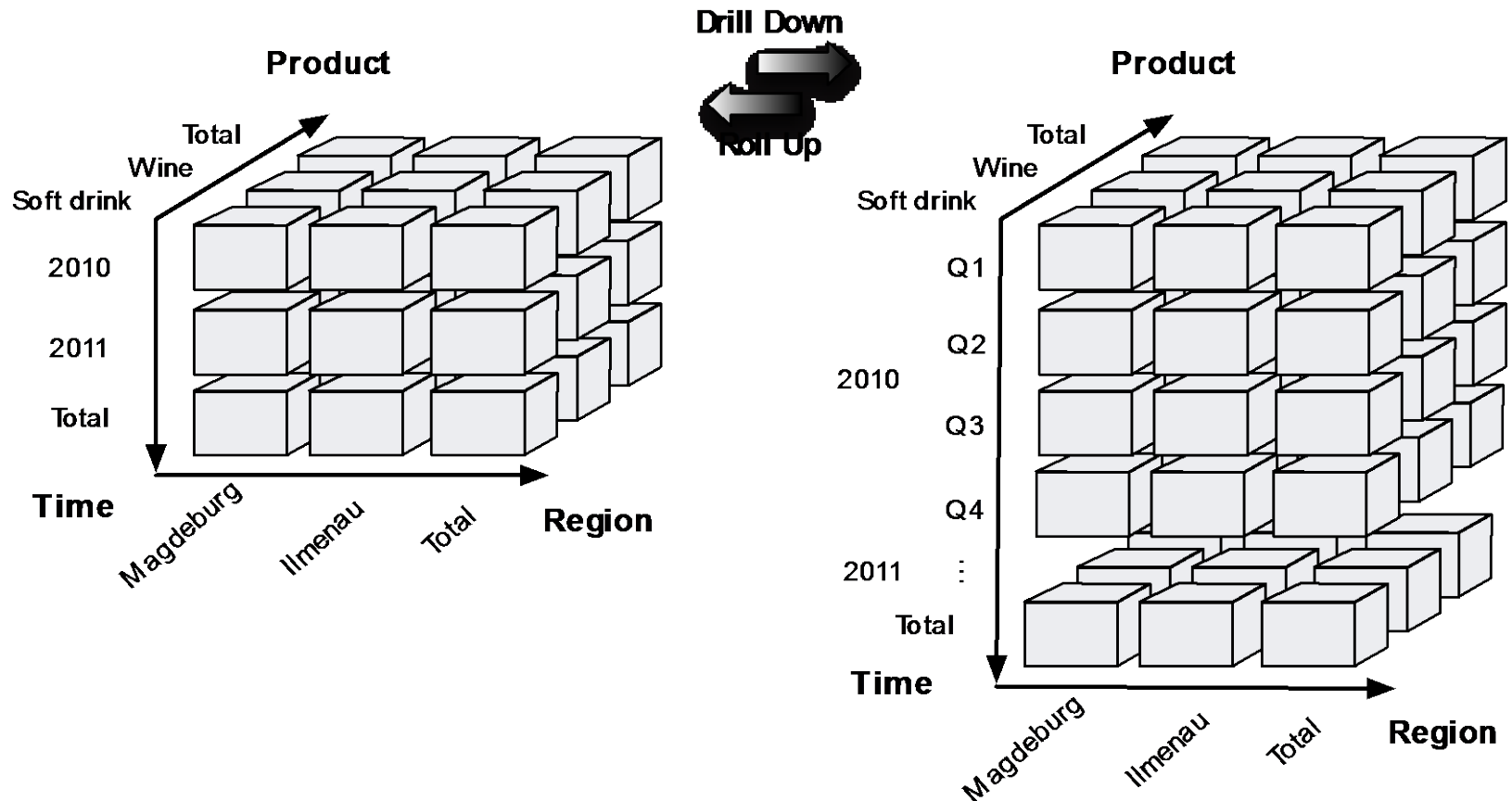
▶ Drill-Down:

- ▶ Complementary to Roll-Up
- ▶ Navigating along the classification path from aggregated data to detail data

▶ Drill-Across:

- ▶ Changing from one cube to another one (with other measures)

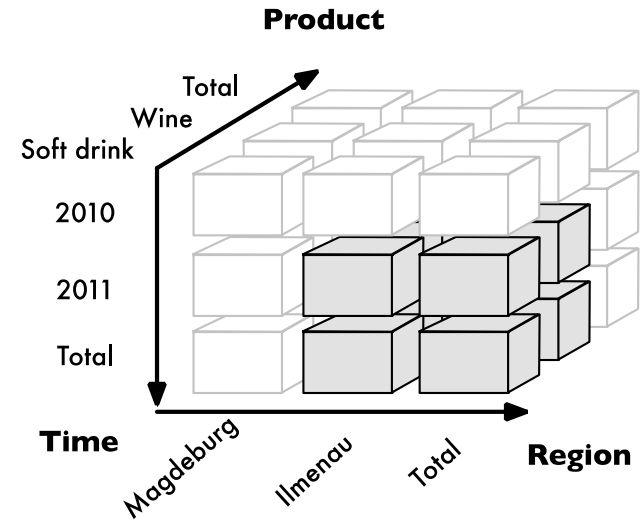
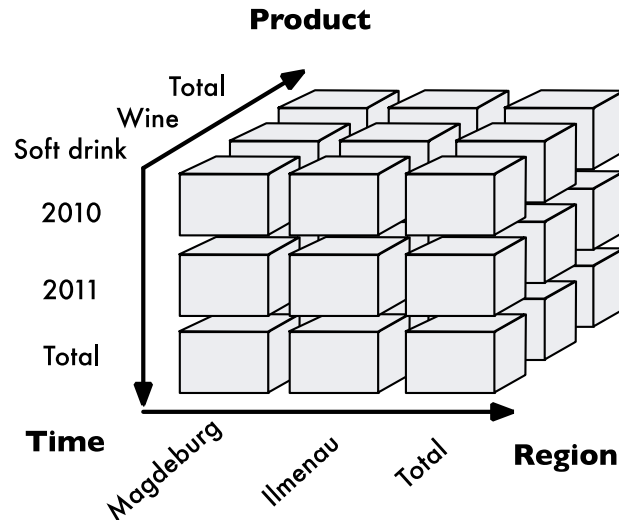
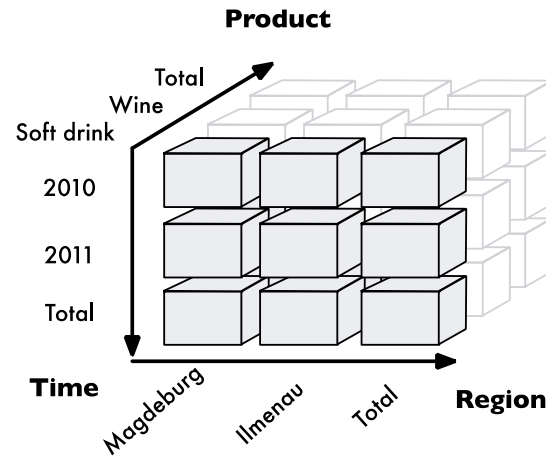
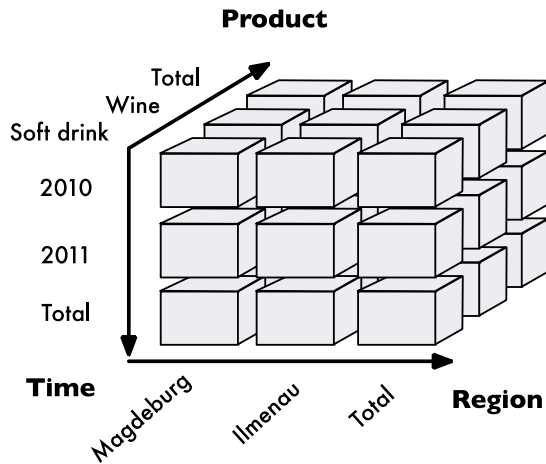
Roll Up and Drill Down



Slice and Dice

- ▶ Creates individual views
- ▶ Slice:
 - ▶ Obtains a slice from the cube
 - ▶ Reduces dimensionality
 - ▶ **Example:** all sales of the current year
 - ▶ Corresponds to the relational projection
- ▶ Dice:
 - ▶ Obtains a sub-cube
 - ▶ Keeps dimensionality, changes hierarchy levels
 - ▶ **Example:** Sales of a certain brand in a given city
 - ▶ Corresponds to the relational selection

Slice and Dice



Rewriting Multidimensional Queries to SQL

- ▶ Basically depends on mapping of multidimensional schema
 - ▶ Star- vs. snowflake schema
 - ▶ Classification hierarchies
- ▶ Frequently used query pattern
 - ▶ $(n+1)$ -multiway join between n dimension tables and the fact table,
 - ▶ restrictions (selections) on some dimension tables,
 - ▶ aggregations on the measures in conjunction with grouping

Star Join Query: Example

```
SELECT region.name, time.year, SUM(quantities)
FROM sales, region, products, time
WHERE sales.product_id = products.product_id
      AND products.group = 'soft drink'
      AND sales.time_id = time.time_id
      AND time.year = 2011
      AND sales.region_id = region.region_id
      AND region.state = 'Thuringia'
GROUP BY region.name, time.year
```

Star Join Query: Structure

- ▶ **SELECT clause**
 - ▶ Measures (possibly aggregated)
 - ▶ Result granularity (e.g. month, city)
- ▶ **FROM clause**
 - ▶ Fact table and dimension tables
- ▶ **WHERE clause**
 - ▶ Join conditions
 - ▶ Dimensional restrictions, e.g.
 products.group = 'soft drink' AND
 time.year = 2011 AND
 region.state = 'Thuringia'

Calculating Sub-Total and Total Sums

Product	Year	Region	Quantities-PYR	Quantities-PY	Quantities-P
Soft drink	2010	Hesse	135		
		Thuringia	120		
				255	
	2011	Hesse	140		
Thuringia		135			
			275		
					530

Product	Year	Region	Quantities
Soft drink	2010	Hesse	135
Soft drink	2010	Thuringia	120
Soft drink	2010	⊥	255
Soft drink	2011	Hesse	140
Soft drink	2011	Thuringia	135
Soft drink	2011	⊥	275
Soft drink	⊥	⊥	530

Calculating Sub-Total and Total Sums

```
SELECT product, CAST(NULL AS INT), CAST(NULL AS VARCHAR(20)),  
        SUM(quantities)  
FROM sales  
WHERE product = 'soft drink'  
GROUP BY product  
UNION ALL  
SELECT product, year, CAST(NULL AS VARCHAR(20)), SUM(quantities)  
FROM sales  
WHERE product = 'soft drink'  
GROUP BY product, year  
UNION ALL  
SELECT product, year, region, SUM(quantities)  
FROM sales  
WHERE product = 'soft drink'  
GROUP BY product, year, region
```

Disadvantages of the UNION Query

- ▶ High effort
 - ▶ Calculating **all** sub-totals for n grouping attributes requires 2^n sub-queries
 - ▶ If join operations are required, they have to be performed multiple times
- ▶ Complex query formulation:
 - ▶ But may be supported by OLAP tools generating queries

Pivot Tables

► Symmetric aggregation

Sales	2010	2011	Total
Thuringia	135	140	275
Hesse	120	135	255
Total	255	275	530

Pivot Tables in MS SQL Server

```
SELECT Year, [THUR] AS Thuringia, [HESS] AS Hesse
FROM Sales
PIVOT ( SUM(sales) FOR State IN ( [THUR], [HESS] ) ) AS Pvt
```

Year	Region	Sales
2010	THUR	135
2011	THUR	140
2010	HESS	120
2011	HESS	135

Year	Thuringia	Hesse
2010	135	120
2011	140	135

CUBE Operator

- ▶ Shortform for query pattern to calculate sub-total and total sums
- ▶ Generates all possible grouping combinations from a given set of grouping attributes
 - ▶ $()$, (A_1) , (A_2) , (A_3) , (A_1, A_2) , (A_1, A_3) , (A_2, A_3) , (A_1, A_2, A_3)
 - ▶ for a given list of attributes A_1, A_2, A_3
- ▶ Result: table with aggregated values
- ▶ Total aggregate:
 $\text{NULL}, \text{NULL}, \dots, \text{NULL}, f(*)$
- ▶ Higher dimensional levels contain less NULL values

Cube Operator: Example

PGroup	State	Year	Quantity
Soft drink	Hesse	2010	45
Soft drink	Thuringia	2010	43
Soft drink	Hesse	2011	47
Soft drink	Thuringia	2011	42



PGroup	State	Year	Quantity
Soft drink	Hesse	2010	45
Soft drink	Thuringia	2010	43
...			
Soft drink	Hesse	⊥	92
Soft drink	Thuringia	⊥	85
Soft drink	⊥	2010	88
Soft drink	⊥	2011	89
Soft drink	⊥	⊥	177
⊥	Hesse	2010	45
...			
⊥	⊥	2010	88
⊥	⊥	2011	89
⊥	⊥	⊥	177

CUBE Operator: Details

► Cardinality

- n attributes A_1, A_2, \dots, A_n with cardinalities (numbers of distinct values) C_1, C_2, \dots, C_n

- Total cardinality of the relation as result of

$$CUBE (A_1, A_2, \dots, A_n) : \prod_{i=1}^n (C_i + 1)$$

► Number of super aggregates in the result

- n attributes in the SELECT clause
- Super aggregates: $2^n - 1$

► **Example:** 10 products, 15 states, 5 years

- $11 * 16 * 6 = 1056$ groups
- $2^3 - 1 = 7$ super aggregates

CUBE Operator in SQL

- ▶ Supported by SQL Server, DB2, Oracle
- ▶ Syntax:

```
SELECT pgroup, state, year, SUM(quantities)
FROM sales
GROUP BY CUBE( pgroup, state, year )
```

- ▶ Scalar function **GROUPING** (*attribute*) in the **HAVING** clause
 - ▶ Returns value = 1, if aggregating on this attribute
 - ▶ Returns value = 0, if grouping on this attribute
- ▶ Usage: suppressing sub-totals or the total sum

```
...
HAVING NOT ( GROUPING( pgroup ) = 1 AND
              GROUPING( state ) = 1 AND
              GROUPING( year ) = 1 )           // suppress the total sum
```

ROLLUP Operator

- ▶ CUBE operator: **interdimensional**
 - ▶ Applicable to attributes from different dimensions
 - ▶ For roll-up or drill-down operation often too expensive
- ▶ ROLLUP operator: **intradimensional**
 - ▶ Generates attribute combinations
$$(A_1, \dots, A_n), (A_1, \dots, A_{n-1}), (A_1, A_2), (A_1), ()$$
for a given list of attributes A_1, \dots, A_n

CUBE	ROLLUP
$(),$ $(A_1), (A_2), (A_3),$ $(A_1, A_2), (A_1, A_3), (A_2, A_3),$ (A_1, A_2, A_3)	$(),$ $(A_1),$ $(A_1, A_2),$ (A_1, A_2, A_3)

ROLLUP Operator in SQL

```
SELECT pgroup, brand, city, state, SUM(quantities) AS Sales  
FROM ...  
GROUP BY ROLLUP ( pgroup, brand ),  
          ROLLUP ( city, state )
```

► Evaluation

- 1. Rollup: (pgroup, brand), (pgroup),()
- 2. Rollup: (city, state), (city), ()
- Cartesian product of both combinations

ROLLUP Operator: Example

Brand	PGroup	State	City	Sales
Waldbrunnen	Soft drink	Thuringia	Ilmenau	102
Waldbrunnen	Wine	Thuringia	Ilmenau	98
Waldbrunnen	⊥	Thuringia	Ilmenau	200
...				
Waldbrunnen	Soft drink	Thuringia	⊥	541
Waldbrunnen	Wine	Thuringia	⊥	326
Waldbrunnen	⊥	Thuringia	⊥	867
...				
Waldbrunnen	⊥	⊥	⊥	1232
...				
⊥	⊥	Thuringia	⊥	1432
...				
⊥	⊥	⊥	⊥	3456

CUBE vs. ROLLUP

▶ CUBE operator:

- ▶ Creates all 2^n combinations:
 - ▶ E.g. for 4 grouping attributes = 16 combinations

▶ ROLLUP operator:

- ▶ Creates only combinations with super aggregates
 - ▶ (f_1, f_2, \dots, f_n) ,
 - ▶ ...
 - ▶ $(f_1, \text{NULL}, \dots, \text{NULL})$,
 - ▶ $(\text{NULL}, \text{NULL}, \dots, \text{NULL})$
- ▶ $n + 1$ combinations

CUBE	ROLLUP
(), (A ₁), (A ₂), (A ₃), (A ₁ ,A ₂), (A ₁ ,A ₃), (A ₂ ,A ₃), (A ₁ ,A ₂ ,A ₃)	(), (A ₁), (A ₁ ,A ₂), (A ₁ ,A ₂ ,A ₃)

GROUPING SETS

- ▶ SQL:2003 grouping sets

GROUP BY ... GROUPING SETS (*grouping specification*)

- ▶ Grouping specification:
 - ▶ Basic grouping combination, e.g. (city, state)
 - ▶ Complex groupings using CUBE or ROLLUP

GROUPING SETS: Example

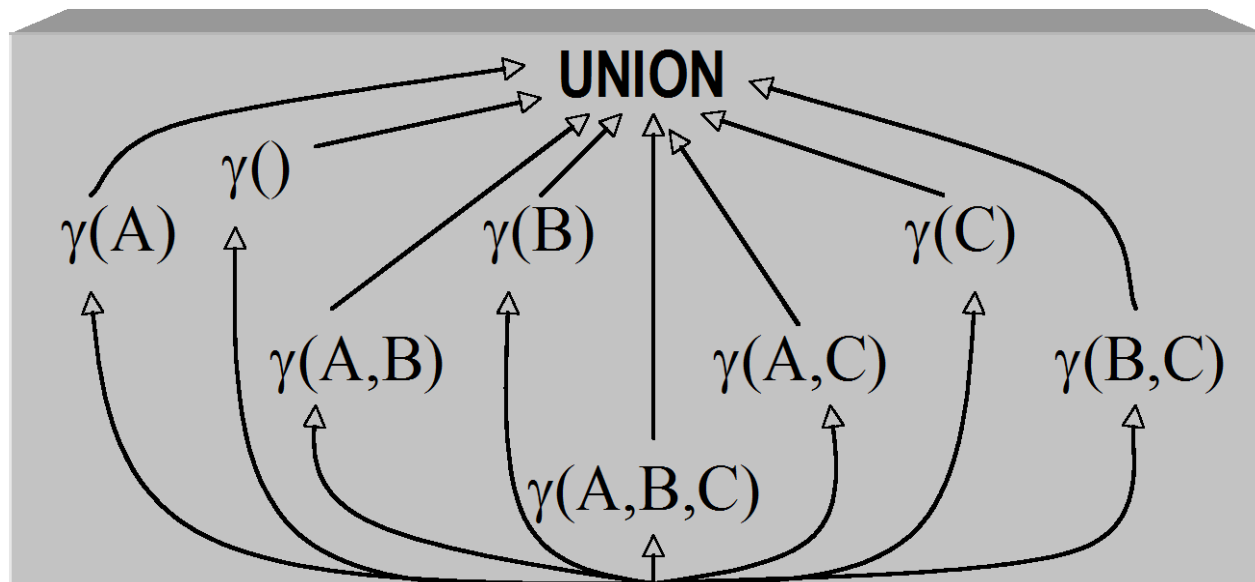
```
...  
GROUP BY  
    ROLLUP ( pgroup, brand ),                -- (1)  
    GROUPING SETS ( (city), (state) ),      -- (2)  
    GROUPING SETS ( ROLLUP ( year, quarter, month ), ( week ) )  -- (3)
```

► Explanation:

- (1) along the classification hierarchy
- (2) only for cities and states
- (3) using parallel hierarchy (year → quarter → month) and (week)

How to compute CUBE?

- ▶ Naive approach:
 - ▶ compute each grouping combination separately
 - ▶ compute the union of all results



CUBE and Aggregation Functions

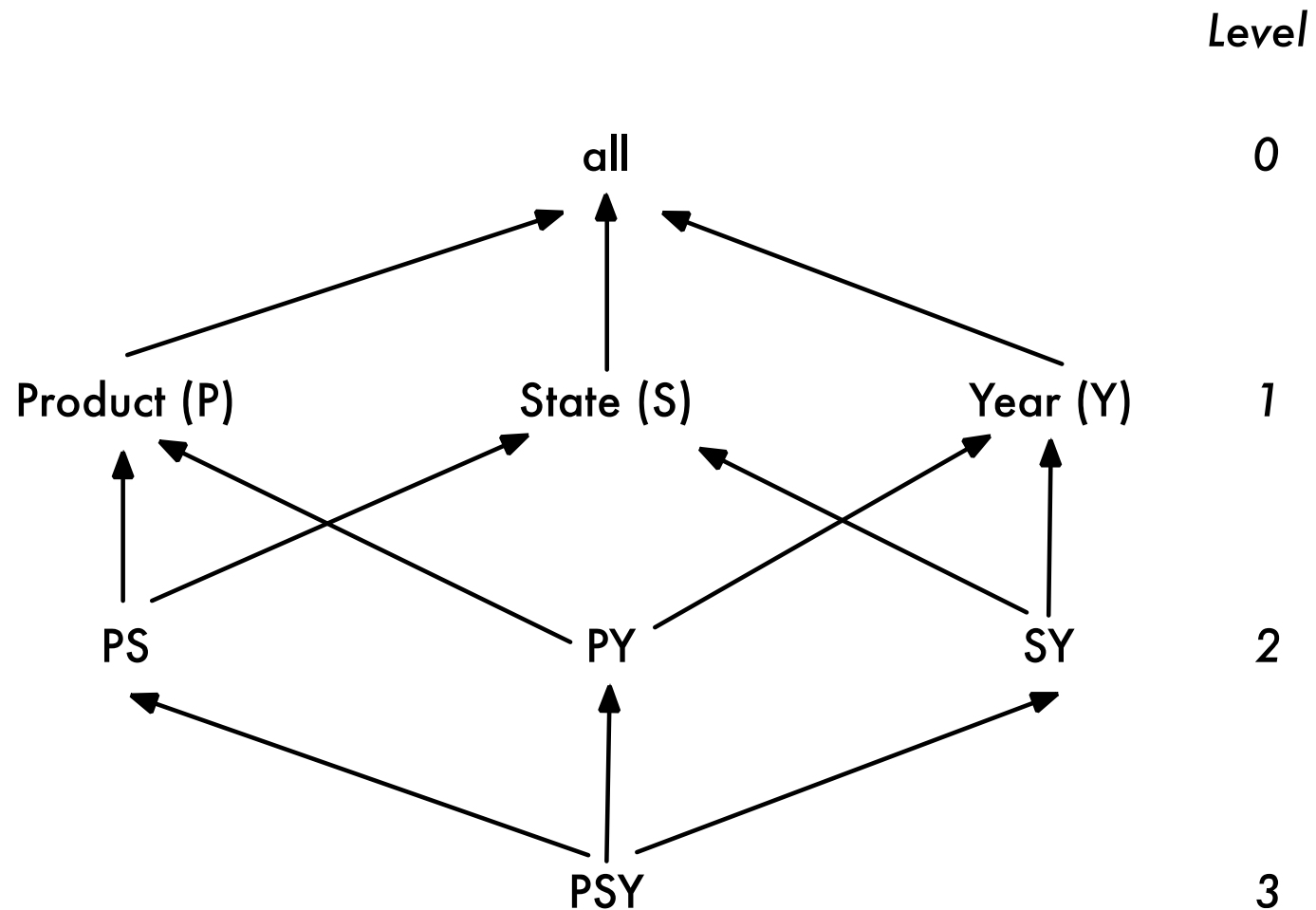
For a better implementation can exploit the fact that:

- ▶ algebraic aggregate functions (which can be expressed in terms of distributive aggregate functions which can process partitioned input sets) allow
 - ▶ to compute less detailed aggregates from more detailed aggregates
 - ▶ `SELECT ... , SUM(quantities)`
 `... GROUP BY (pgroup, state, year)`
 - ▶ `SELECT ... , SUM(quantities)`
 `... GROUP BY (pgroup, state)`
 - ▶ partial order of group-by operations in the cube

Derivability

- ▶ Derivability relationship between group-by combinations G_i
- ▶ directly derivable:
 - G_2 is directly derivable from G_1 iff (if and only if)
 - ▶ G_2 has one attribute less than G_1
 - ▶ or in G_1 one attribute A is replaced by B and $A \rightarrow B$
- ▶ forms a aggregation lattice (or search lattice)
- ▶ derivable:
 - G_2 is derivable from G_1 within an aggregation lattice if there is a path from G_1 to G_2

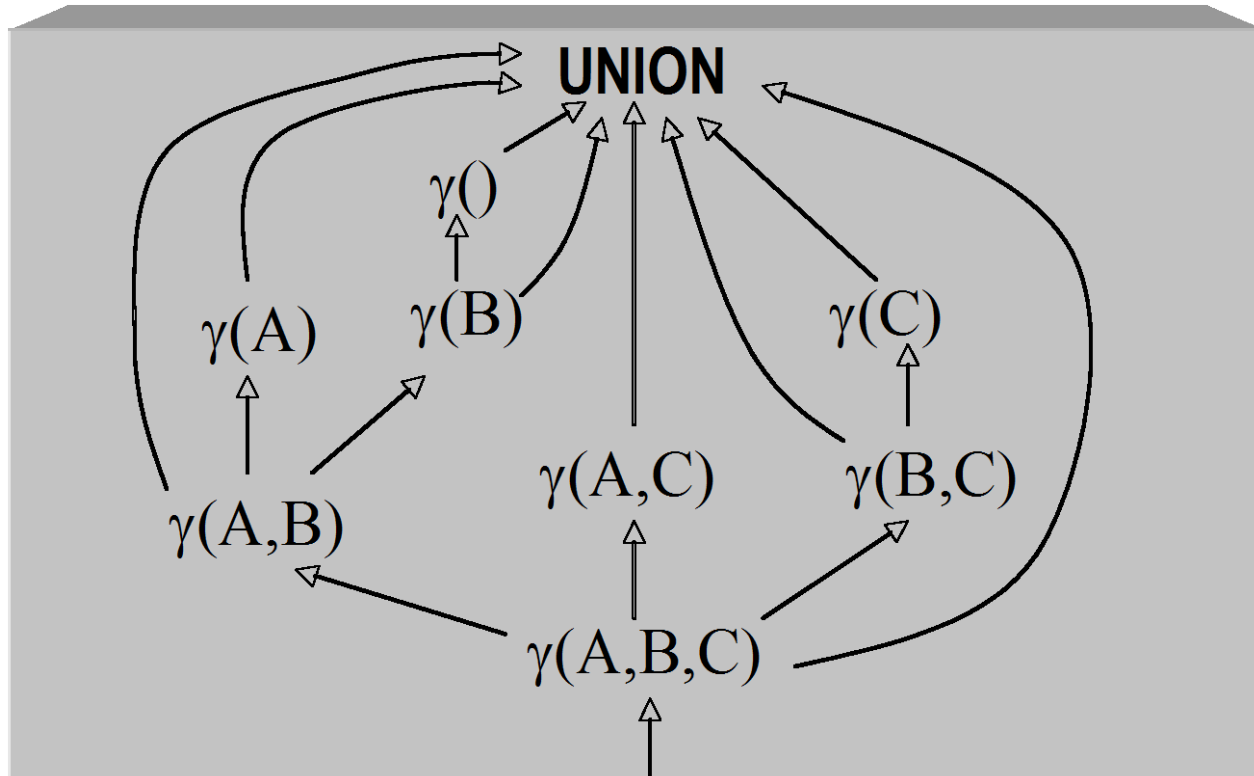
Search Lattice



Computing the CUBE

- ▶ **Idea:**
 - ▶ exploit the derivability relationship using the search lattice
 - ▶ group-by combinations with shared attributes can share partitions, sorts, ...
- ▶ **Algorithms (Sarawagi, Agrawal, Gupta: VLDB 96)**
 - ▶ based on sorting
 - ▶ based on hashing

Optimized Computing



Possible Optimizations

- ▶ **Smallest-parent**
 - ▶ compute group-by based on the minimal parent group-by
- ▶ **Cache-results**
 - ▶ cache results of a group-by for subsequent group-bys (e.g. $ABC \rightarrow AB$)
- ▶ **Amortize-scans**
 - ▶ compute multiple group-by combinations in a single scan (e.g. ABC, ACD, ABD, BCD from ABCD)
- ▶ **Share-sorts**
 - ▶ cache and reuse sorts
- ▶ **Share-partitions**
 - ▶ cache and reuse partitions for hashing

Effects of Sorting

- ▶ Assumptions: group-by implemented using sorting
- ▶ example:
 - ▶ group-by (product, year, region): sum(quantities)
 - ▶ tuples sorted on product, year, region
- ▶ use the result for
 - ▶ group by (product, region): sum(quantities)
 - ▶ requires to re-sort

Cost Model

- ▶ Types of costs
 - ▶ S -costs = Still-to-sort: compute a group-by G_j from G_i where G_i is not sorted yet
 - ▶ A -costs = Already-sorted: compute a group-by G_j from G_i where G_i is sorted
- ▶ Cost estimations based on data dictionary, ... (see query processing chapter)

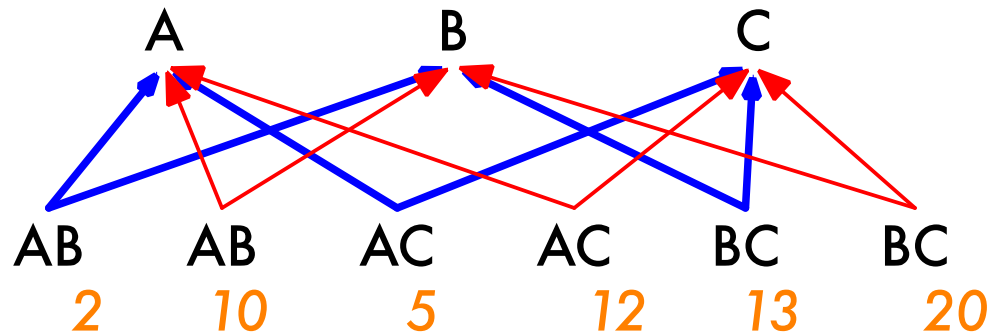
PipeSort

- ▶ **Input: search lattice**
 - ▶ edges e_{ij} connecting two group-by G_i and G_j are annotated with A - and S -costs
 - ▶ level k comprises all group-by's with k attributes
- ▶ **Output: subgraph of search lattice**
 - ▶ each node is connected with only one parent node
 - ▶ determines sorting order while allowing pipelining
- ▶ **Goal: find subgraph with minimal total costs**

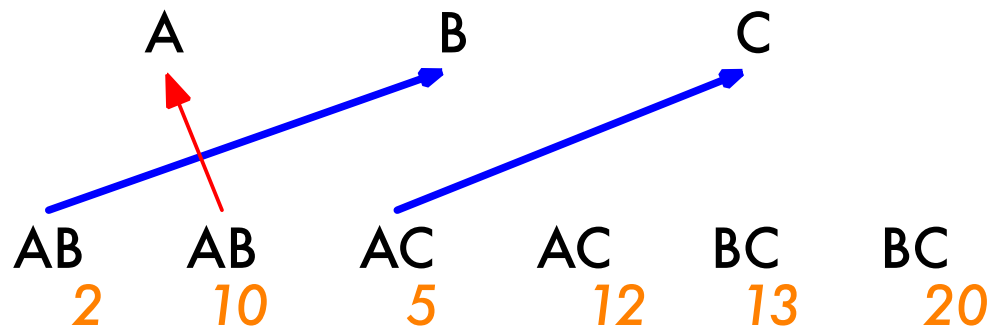
PipeSort Algorithm

- ▶ graph algorithm
- ▶ process level-wise: $k = 0..N-1$ (N = number of attributes)
- ▶ build level $k+1$ by making k copies of each node
- ▶ each copy of a node is connected to the same nodes as the original node
- ▶ edge costs for original node $A(e_{ij})$, otherwise $S(e_{ij})$
- ▶ find graph with minimal costs by constructing pairs and minimize total costs (weighted bipartite matching problem)

PipeSort: Example



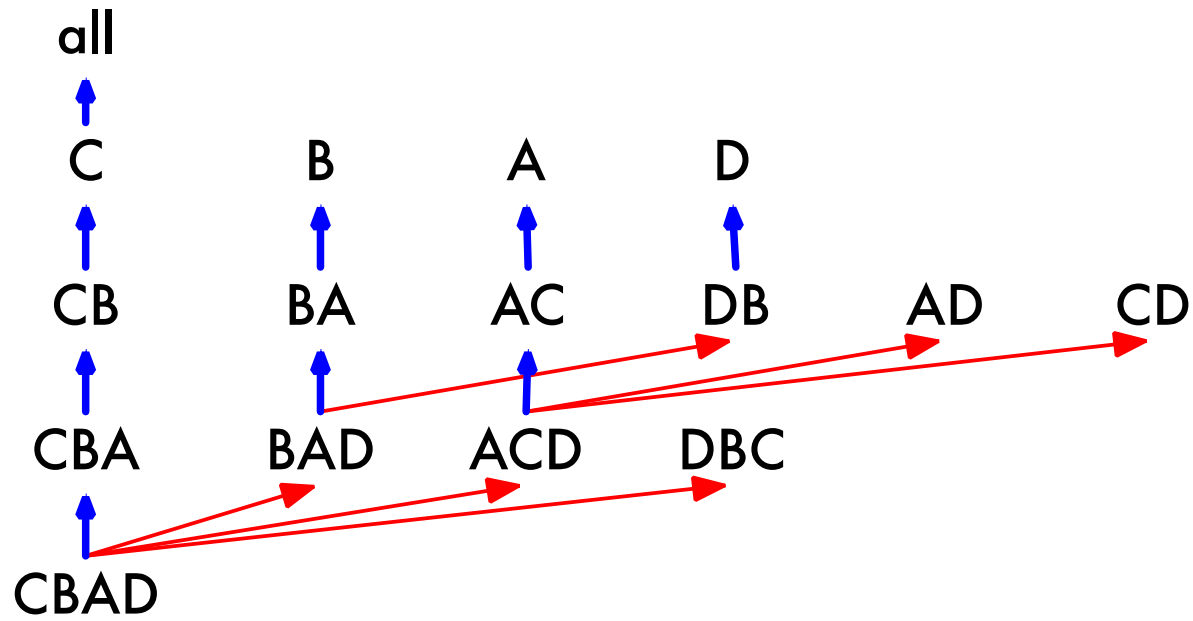
→ Pipeline → Sort



PipeSort: Sort Plan

- ▶ each node h at level k is connected to a node g at level $k+1$
- ▶ $h \rightarrow g$ with A -edge: h determines sort order for g
- ▶ $h \rightarrow g$ with S -edge: for computing g sorting is required

PipeSort: Sort Plan



Relation

—→ Pipeline —→ Sort

Summary

- ▶ Challenges of query formulation and processing in Data Warehouses
- ▶ CUBE and ROLLUP operators for simplifying formulation of complex grouping combinations (e.g. for OLAP, sub-total calculation, ...)
- ▶ efficient computation of cube

Auf Wiedersehen in Ilmenau



