Data Warehousing

Data Warehouse Design & Multi-dimensional Models

FS 2020

Dr. Andreas Geppert
geppert@acm.org

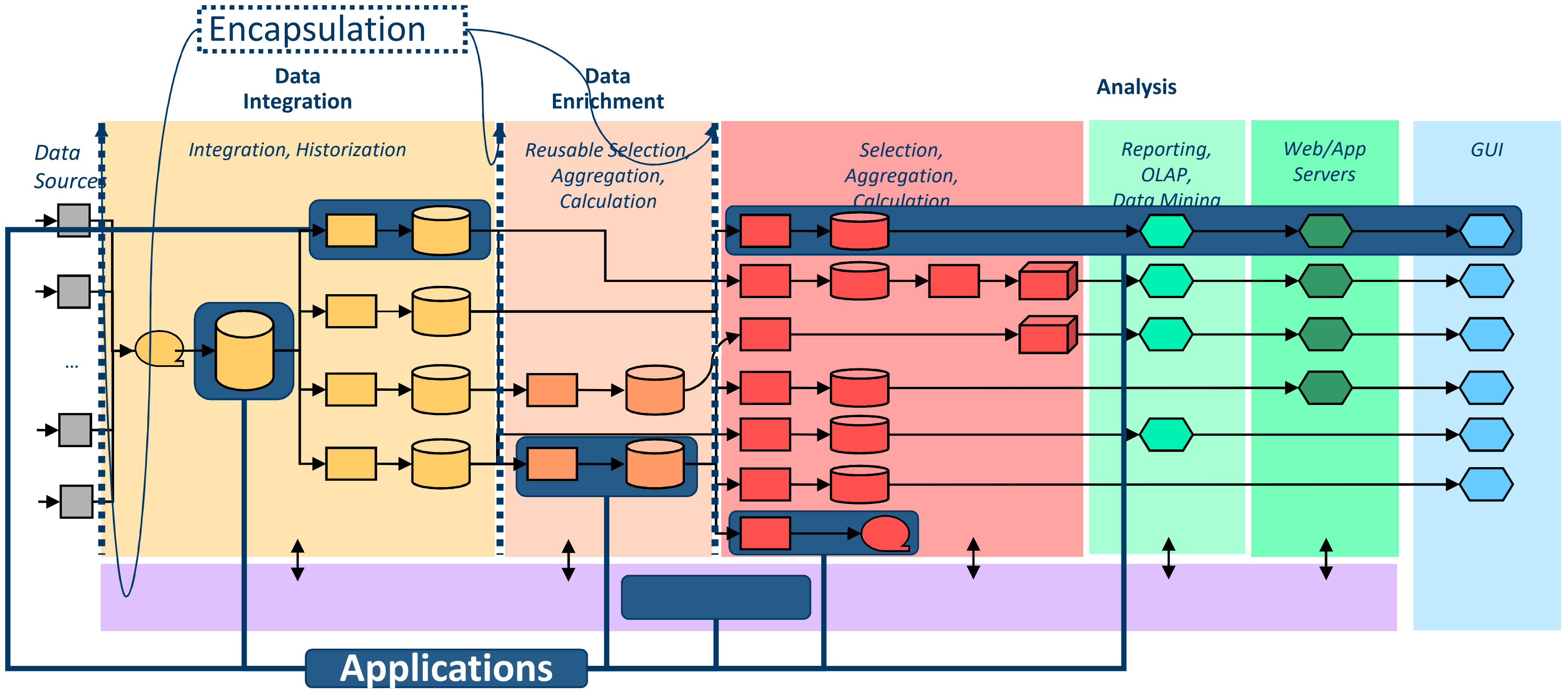
Outline of the Course

Content

Motivation

- complex facts are represented in the DWH
- DWH supports integration and analysis
  - databases must be modeled accordingly
  - without complete and adequate data models, DWH will not be a success
- analytic databases (data marts) use multidimensional concepts
  - how do you systematically design these databases?
  - what is the analogon to the Entity Relationship Model and to the mapping ER → RM ?
- notations and conceptual (meta) data models
- modeling approaches

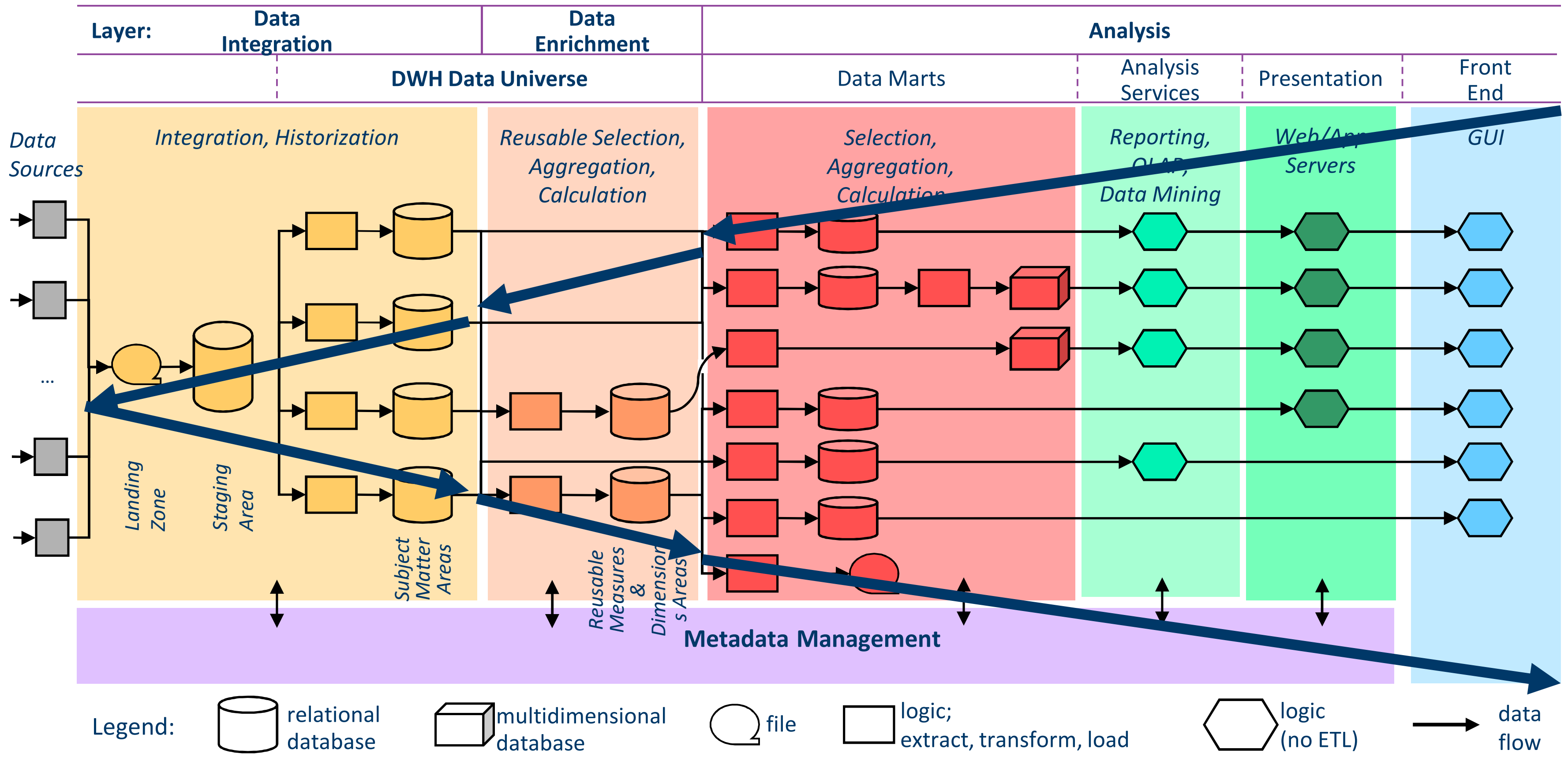
Reference Architecture V4 Applications


Encapsulation
Data Integration
Data Enrichment
Analysis
Data Sources
Integration, Historization
Reusable Selection, Aggregation, Calculation
Selection, Aggregation, Calculation
Reporting, OLAP, Data Mining
Web/App Servers
GUI
...
Applications

- build "seed" IL containing most-needed data and integrated most important sources

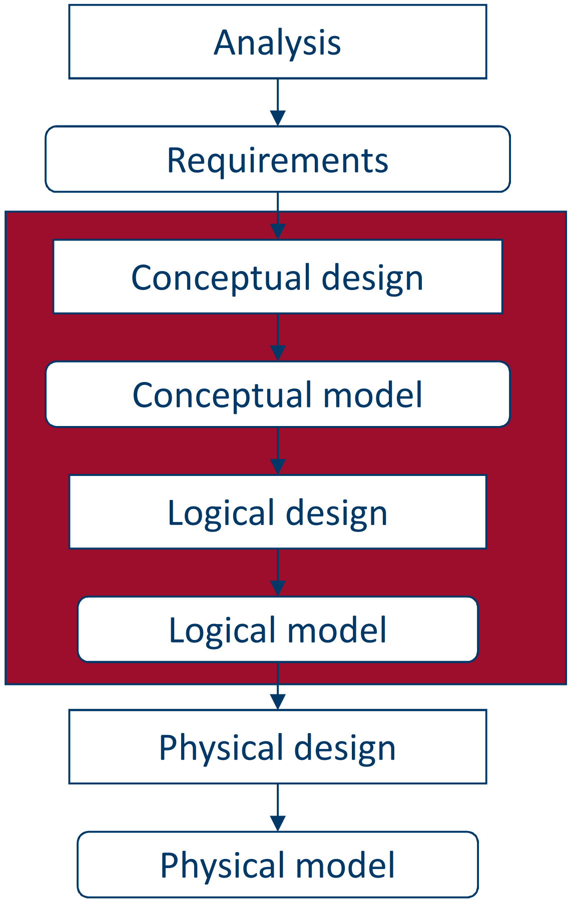
DWH Application Development

- start with requirements of analysis applications



General Approach & Data Model Hierarchy

- modeling-driven approach
- model/specify on abstract level and derive (or even generate) lower-level constructs
- distinguish conceptual, logical, and physical model
- the same hierarchy is applied to ETL processes / mappings



Data Models: Conceptual

- The Conceptual Data Model serves the following purposes:
  - Unambiguously represent business information structures and rules, enabling communication of this understanding to the entire development team
  - Provide an implementation–independent set of requirements as input to the logical data model, and to the physical data model
  - Clearly and uniquely identify all business entities in the system
- Note that the conceptual data model should **not** be considered as an intermediate design document to be disregarded after logical and physical design; rather it should remain as a part of the database specifications, organized with a variety of documents that also describe in detail the requirement acquisition and design process
- Finally one of the possibly most important advantages of conceptual design shows up during the operation of the database when the conceptual model and its documentation ease the understanding of data schemas and of applications that use them and thus facilitate their transformation and maintenance.

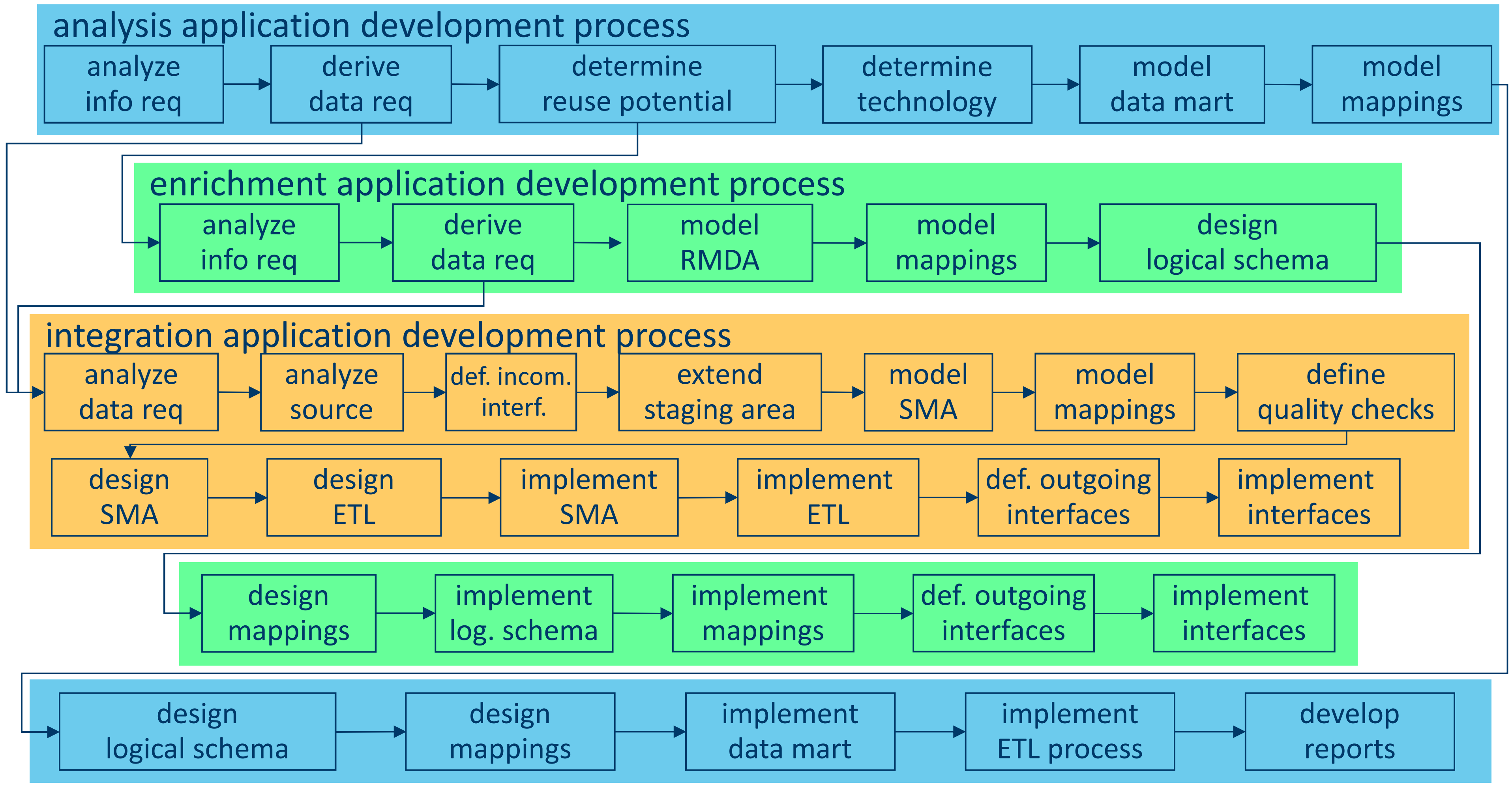
Data Models: Logical

- The Logical Data Model (LDM) is a database-near data model that hides details of data storage and DBMS-specific idiosyncrasies but can nevertheless be implemented straightforward on a computer system
- Its main purpose is to ensure a proper mapping from a high-level conceptual data model (i.e., an Entity Relationship Model) that focuses exclusively on business entities and their relationships to the (principal) schema constructs used by a class of DBMSs (e.g., relational DBMSs). In other words logical design is conducted in the same way for all relational DBMSs (e.g., Oracle, DB2 etc.) because they all implement the relational data model. As a consequence a specific relational logical model can be used "as is" to design the physical data model of DB2, Oracle, SQL Server etc. whereas it cannot be used to design the physical data model for an IMS System.

Data Models: Physical

- The purpose of the Physical Database Design is to ensure that database transactions and queries meet performance requirements while reflecting the semantics of the Logical Data Model
- While the Logical Data Model contains the information requirements of the system in a normalized form, a direct implementation of the model is unlikely to meet performance requirements
- Physical Database Design takes into account data and transaction volume as well as typical queries to produce a schema and environment that will meet necessary performance requirements.

Application Development Processes Big Picture



Content

Schema Design for Data Warehouses

- database design must support goals of the data warehouse
- integration
  - schema integration
  - target of ETL processes
- historization
- data quality
- granularity

- in most architectural styles (Hub-and-Spoke, etc.) the DWH is implemented relationally on the logical level (3NF)
- the ER Model can/should be used for conceptual modeling

Schema Design for Data Warehouses

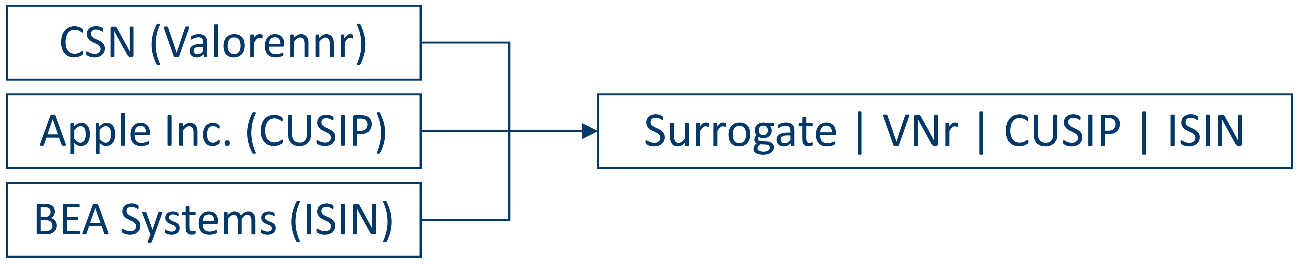
- Requirements to the data warehouse result from requirements analysis of the analysis/reporting application
- are the data that an application under construction needs already in the data warehouse (integration layer) and can be sourced from there?
  - if not, those data have to be sourced from one or more data sources / operational systems
- it is absolutely crucial to maintain accurate, timely, and complete information about the data in the data warehouse
  - data in the DWH must be modeled ($\rightarrow$ conceptual schemas !)
  - semantics of data must be understood
  - data ownership must be defined

Integration

- the data warehouse integrates data from different sources
  - different aspects of the same business entities are managed in different business processes with disjoint databases
  - different business processes over the same business entity operate on disjoint databases
  - the same business process is implemented by multiple applications (e.g., because of mergers and acquisitions); e.g., multiple CRM systems
- "vertical" integration: integrate attributes from different sources into the same entity
- "horizontal" integration: integrate entities from different sources into the same entity collection (logically: relation)

Key Generation and Integration

- different sources usually maintain different kinds of primary keys
  - different types of keys (e.g., securities)
  - overlapping sets of key values (no globally unique identifiers)
- business keys vs. technical keys
- use artificial/technical keys in the data warehouse ("surrogate keys")
  - map business keys / source keys → surrogate
  - maintain business keys as attributes
  - surrogates are not visible to users and applications should not rely on the mapping of business keys onto surrogates



Historization

- DWH must represent historical evolution of objects
- differing states of objects on the timeline
- bi-temporal time notion (temporal databases)
- Validity time: Interval, during which an object has been in a specific state (e.g., during which an attribute has had a certain value)
- Transaction time: point in time when the state of an object changed (e.g., an attribute has been modified). Could also be an interval.

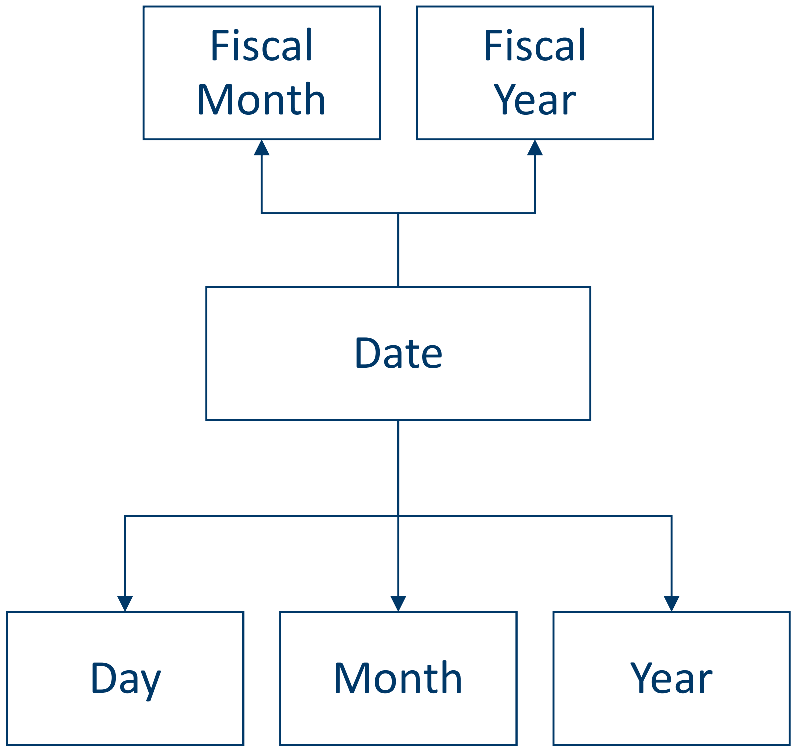
Historization (2)

- Customer Calvin has been living in Basel for a long time
- On April 1, he moved to Bern. He announced his move one week in advance
  - this change is reflected by an update (closing the validity interval of the old address) and an insert (containing the new address with an infinite valid_until_date)
- On November 11, we learned that Calvin was deleted from the customer database as per November 1.
  - This change is reflected by an update (update of the transaction time and closing of the validity interval)

Customer	Name	Address	TX_Time	Valid_From	Valid_Until
12345	Calvin	Basel	2004-04-04	2004-04-01	~~9999-12-31~~ 2019-03-31
12345	Calvin	Bern	2019-03-24 2019-11-11	2019-04-01	9999-12-31 ~~9999-12-31~~ 2019-11-01

Modeling of "Time"

- date and time are important properties in many applications
- in addition to the data and time, further properties are important, depending on the application (e.g., holidays)
- different calendars exists (we normally use the Gregorian Calendar)
- in addition to the calendar year, other notions of "year" are common (e.g., fiscal year)
- in many cases, the explicit modeling of date and time is recommended over simply using the database system's calendar



Data Vault Modeling

- The Data Vault technique has been introduced in the 1990s
- Today it is used in many DWH projects
- Previous techniques (3NF–based data models) have issues with changing sources. Data Vault modeling has been designed to better cope with such changes
- The Data Vault main components:
- Hubs
- Link Tables
- Satellites

Data Vault Main Elements: Hub Entities

- Represent an entity in the subject area of interest
- Carries at a minimum a unique list of business keys
  - invoice number, customer number, employee PID, …
- In case the business uses multiple business keys, the hub contains multiple rows (one for each business key)
- Surrogate key: optional (see surrogates above)
- Load date timestamp: records when the key was first loaded into the data warehouse
- Record source: reference to the source system where the business key came from, for traceability reasons

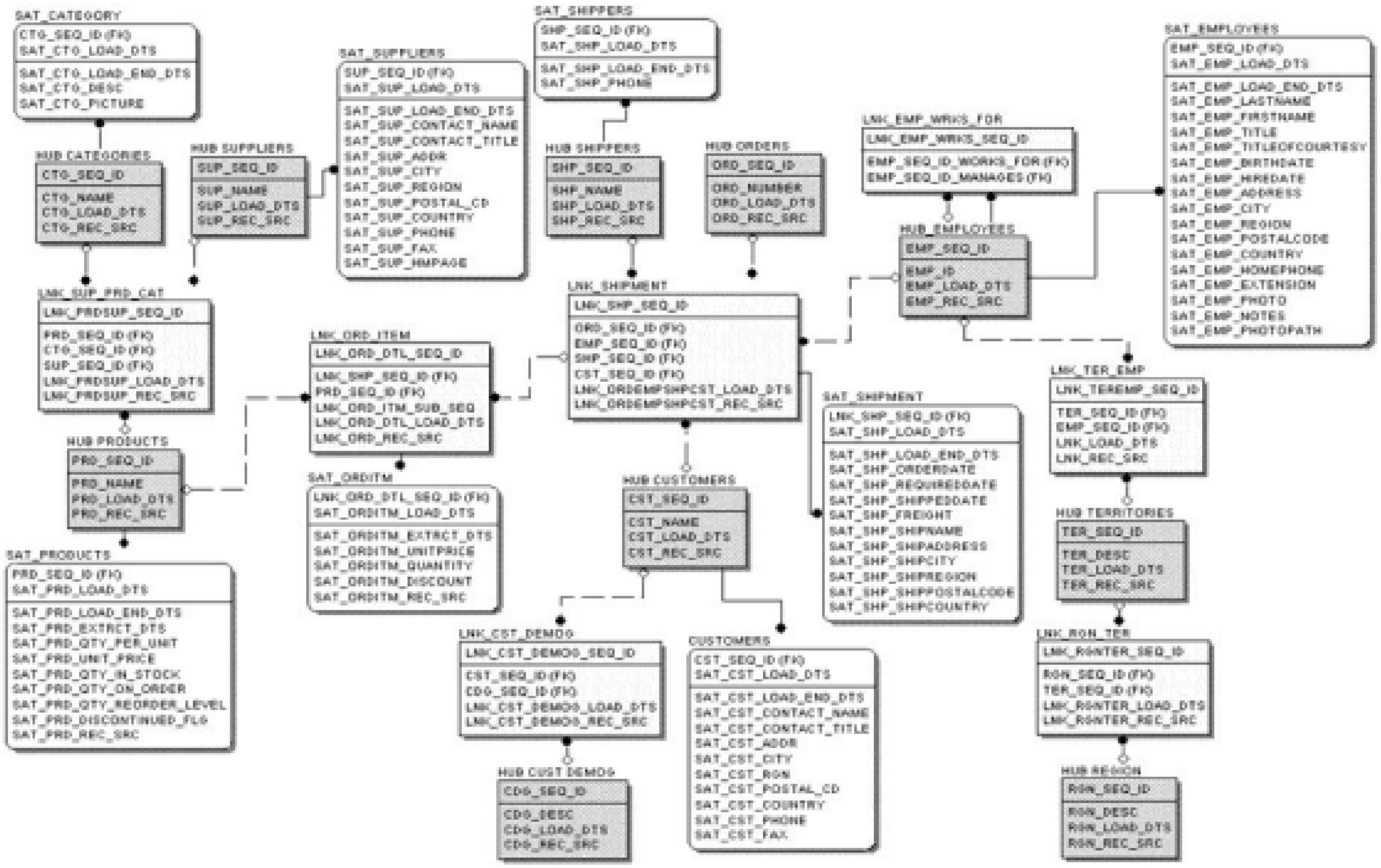
Data Vault Main Elements: Link Entities

- Represent the relationship between two or more business components
- Hub keys represent the relationship between the hubs
- Surrogate key: optional component
- Load date timestamp: indicates when the relationship was first created in the warehouse
- Record source: indicates from which data source the relationship was loaded; for traceability reasons

Data Vault Main Elements: Satellite Entities

- hub key context (descriptive) information
- satellite data are subject to change over time; therefore the structure must be capable of storing new or altered data
- Satellite primary key: Hub or link primary key
- Load date timestamp: indicates when the context information was first created in the warehouse
- Sequence surrogate number: optional. Useful for Satellites that have multiple values
- Record source: indicates from which data source the satellite was loaded; for traceability reasons

Data Vault Example



source: https://danlinstedt.com/solutions-2/data-vault-basics/

Inhalt

Multi-dimensional Data Models

- "Classical" relations:
  - One-dimensional (not in the mathematical sense)
  - Relation maps key onto attributes
- However, in many cases in data warehousing one is interested in multiple perspectives („dimensions")
  - Example: Sales based on product, time, region, customer, store, manager/employee
- ➢ Cannot be represented with normal relations
- ➢ Multi-dimensional data models
- ➢ Multi-dimensional database systems

Comparison: relational vs. Multi-dimensional DM

relational model

- Simple, little semantics
- Application–neutral
- Less ostensive modeling ($\Rightarrow$ ERM)
- standardized

Multi–dimensional data models

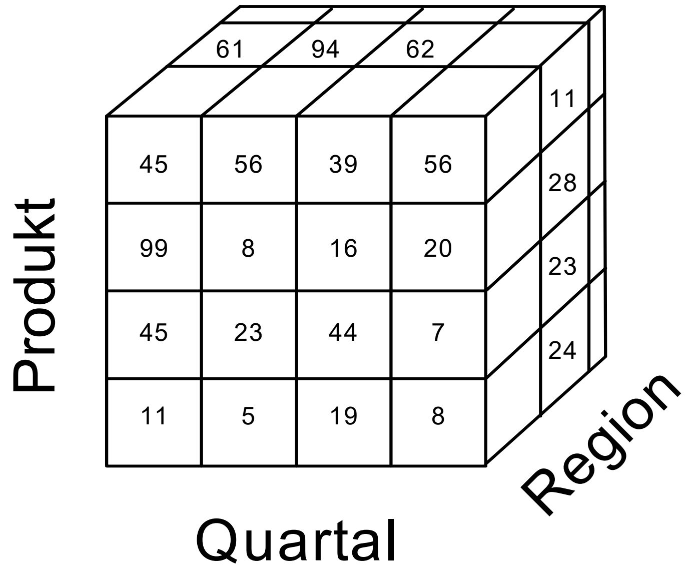
- More complex, more semantics
- Well–suited only for specific applications
- Ostensive modeling $\Rightarrow$ user–friendly
- Caution: the multi–dimensional data model does not exist
  - No uniform query language
  - No standards
  - No uniform formalization

Content

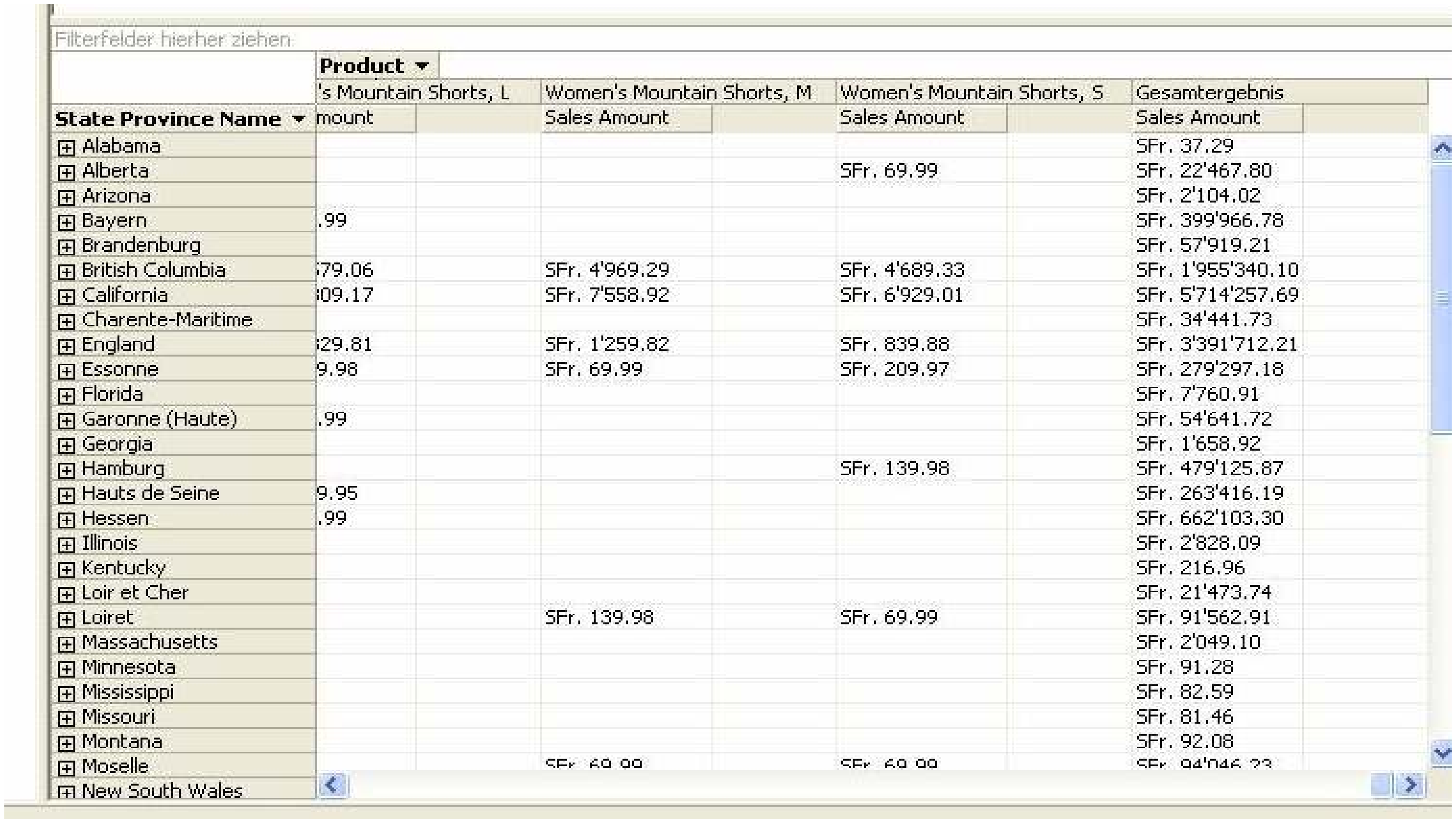
Multi-dimensional Data: Cubes

- Multi–dimensional data are seen and represented as data cubes
- More precisely: *Hypercubes*

- Distinction into
  - Qualifying and
  - Quantifying information
- Qualifying information identifies cell or sub-cube
- Quantifying information contains numbers



Multi-dimensional Data: Representation



Filterfelder hierher ziehen

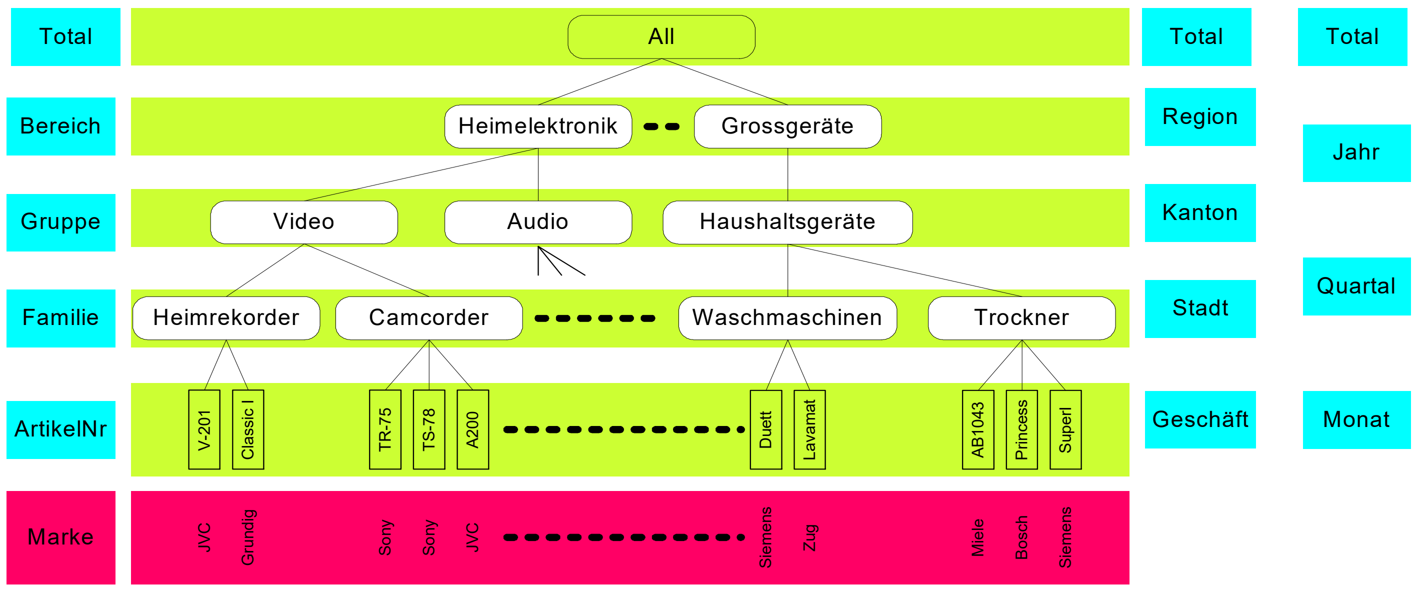
	Product			
	's Mountain Shorts, L	Women's Mountain Shorts, M	Women's Mountain Shorts, S	Gesamtergebnis
State Province Name	mount	Sales Amount	Sales Amount	Sales Amount
Alabama				SFr. 37.29
Alberta			SFr. 69.99	SFr. 22'467.80
Arizona				SFr. 2'104.02
Bayern	.99			SFr. 399'966.78
Brandenburg				SFr. 57'919.21
British Columbia	79.06	SFr. 4'969.29	SFr. 4'689.33	SFr. 1'955'340.10
California	09.17	SFr. 7'558.92	SFr. 6'929.01	SFr. 5'714'257.69
Charente-Maritime				SFr. 34'441.73
England	29.81	SFr. 1'259.82	SFr. 839.88	SFr. 3'391'712.21
Essonne	9.98	SFr. 69.99	SFr. 209.97	SFr. 279'297.18
Florida				SFr. 7'760.91
Garonne (Haute)	.99			SFr. 54'641.72
Georgia				SFr. 1'658.92
Hamburg			SFr. 139.98	SFr. 479'125.87
Hauts de Seine	9.95			SFr. 263'416.19
Hessen	.99			SFr. 662'103.30
Illinois				SFr. 2'828.09
Kentucky				SFr. 216.96
Loir et Cher				SFr. 21'473.74
Loiret		SFr. 139.98	SFr. 69.99	SFr. 91'562.91
Massachusetts				SFr. 2'049.10
Minnesota				SFr. 91.28
Mississippi				SFr. 82.59
Missouri				SFr. 81.46
Montana				SFr. 92.08
Moselle		SFr. 69.99	SFr. 69.99	SFr. 94'046.23
New South Wales				

Multi-dimensional Data: Dimensions

- Dimension:
  - Set of (at least two) dimension elements
    - All articles, customers, …
  - Analysis perspective of an application area
  - Qualifying information
  - "cube axes"
- Examples:
  - Customer, Product, Time, Store, …

Multi-dimensional Data: Dimensions (2)

- Dimensions are rarely flat
- Attributes form a classification hierarchy
- Examples:
  - Month → Quarter → Year
  - City → Canton → Region
  - Product → Product family → Product group → Area



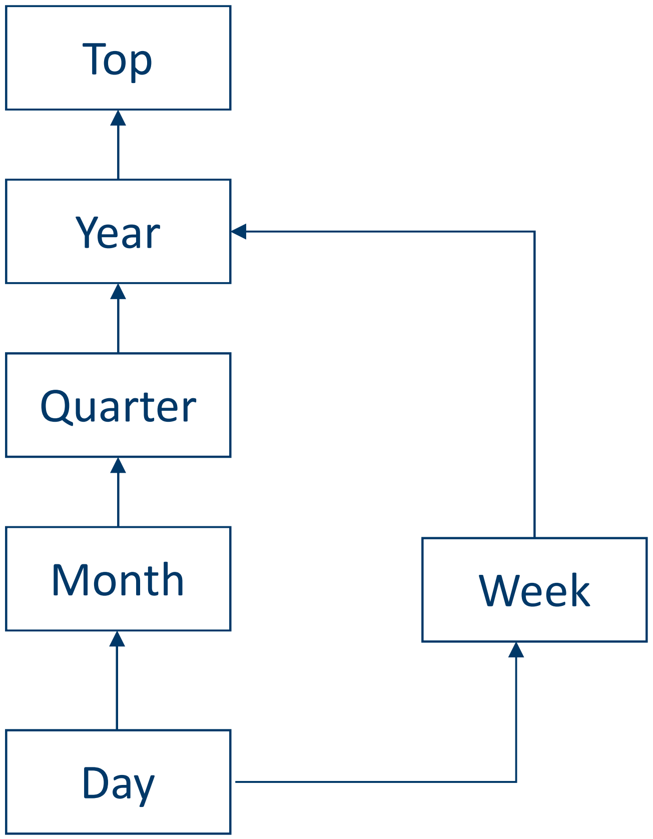
Dimension Schemas

- Schema of a dimension hierarchy
- Partially ordered set D of dimensional attributes
  - $(\{D_1, \cdots, D_n, \text{Top}_{\mathcal{D}}\}; \rightarrow)$
  - $\rightarrow$ functional dependency

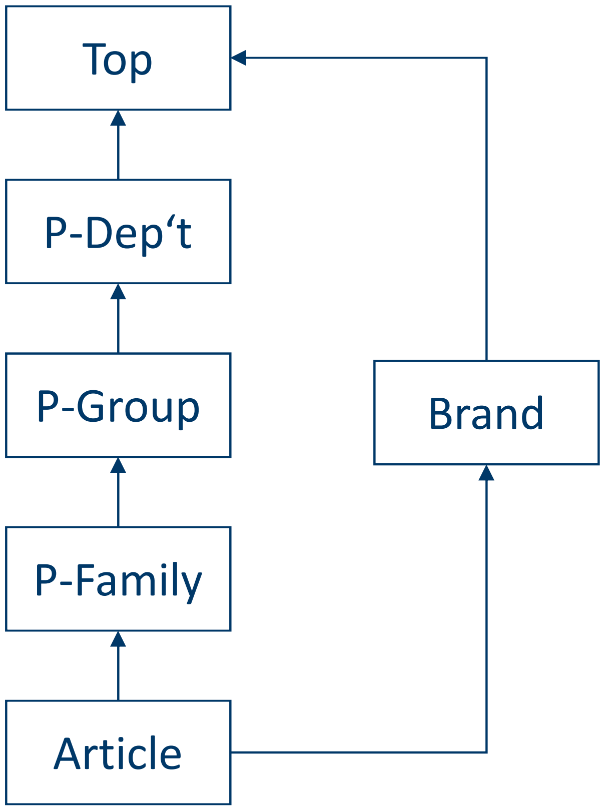
    attribute A determines B ($A \rightarrow B$), if the value of B is uniquely determined by the value of A
  - $\text{Top}_{\mathcal{D}}$ is the maximum element regarding $\rightarrow$
    - $\forall D_i: D_i \rightarrow \text{Top}_{\mathcal{D}}$
  - There is a unique, smallest element D_i, which determines all others
    - $\exists D_i: \forall D_{k,\, k \neq i}: D_i \rightarrow D_k$
- Example:
  - {Day, Month, Quarter, Year, Top}, $\rightarrow$
  - Day $\rightarrow$ Month $\rightarrow$ Quarter $\rightarrow$ Year $\rightarrow$ Top

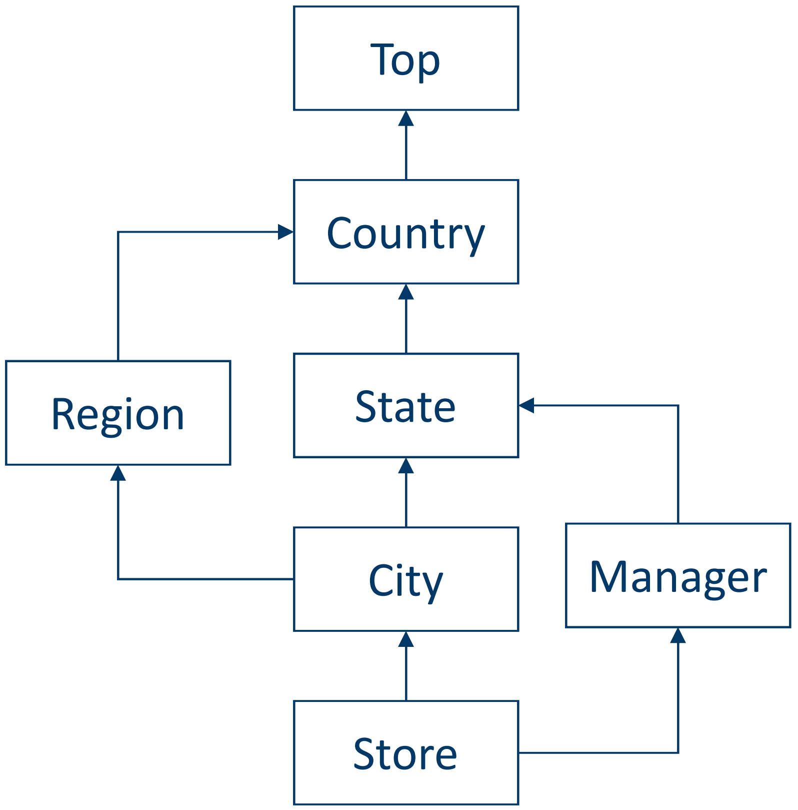
Dimension Schemas

- Partial ordering allows for parallel hierarchies
- Example:
  - {Day, Month, Quarter, Year, Calendar Week}, $\rightarrow$
  - Day $\rightarrow$ Month $\rightarrow$ Quarter $\rightarrow$ Year
  - Day $\rightarrow$ Calendar Week $\rightarrow$ Year
- Orthogonality
  - $\neg \exists\, D.\mathrm{D}_i \rightarrow D'.\mathrm{D}_j$



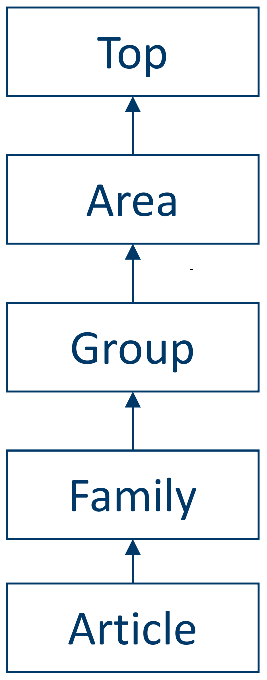
Dimension schemas: Examples Product and Store

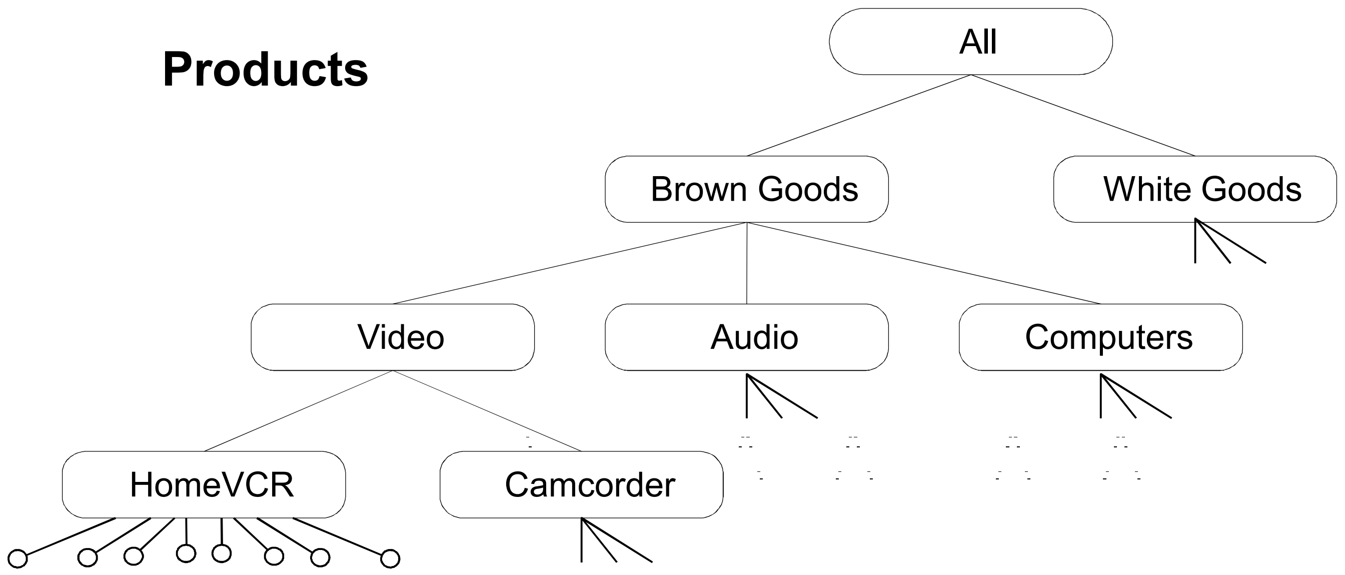




Dimension schemas and Instances

- Functional dependencies determine instances structures
  - 1:n relationships
  - Path to the root: consolidation path
    - Camcorder → Video → Brown Goods → All





Dimension Schemas: Special Cases

- Relationships between hierarchy levels not always 1:n
- Element node has more than one predecessor
  - No tree structure anymore, but acyclic graph
  - Example: Product ⇢ Promotion
- Gaps in the ancestor relation
  - Node on level n+1 does not have predecessor on level n
  - Tree is no longer balanced
  - Example: Shop → City → State

Multi-dimensional Data: Measures

- Quantifying information
  - Usually numeric
- Key figures, measures
- Examples:
  - sales figures
  - turnover
  - Measurements (temperature, rainfall, …)

Characteristics of Measures

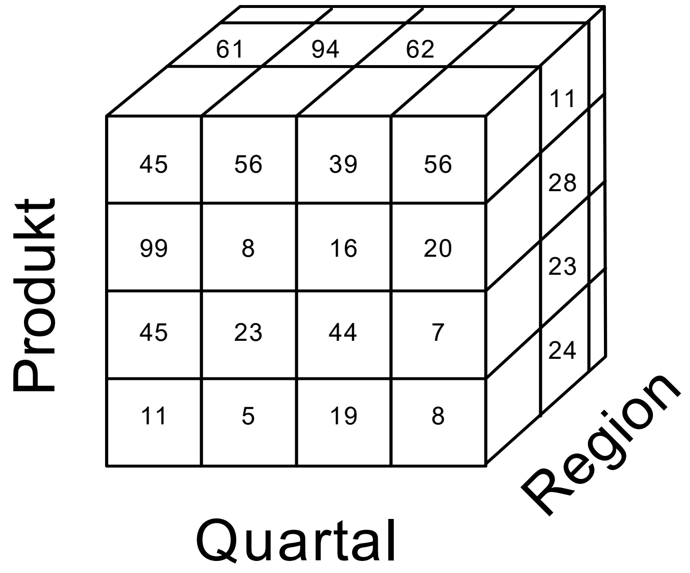
- Name, data type, range
- Aggregation type
  - Defines which aggregation operation are meaningful (hence, allowed)
  - FLOW: can be aggregated in arbitrary ways (sales, turnover)
  - Stock: cannot be summed up over time (inventory, …)
  - VPU (Value per Unit): cannot be summed up at all (price, tax, …)
- Average, minimum, maximum are always possible
  - When semantically meaningful

Multi-dimensional Data: Facts

- A fact is an element of the multi–dimensional space
- Associates a set of dimension elements with measures
- „cube cell“
  - Granularity is given by dimension elements
  - Fact is uniquely identified through a combination of dimension elements
- Qualifying and quantifying information
- Interesting occurrences
- Sometimes misleading use of terms
- Facts and measures are sometimes confused, but are not the same
- Fact $\cong$ Cell, Measure $\cong$ cell content

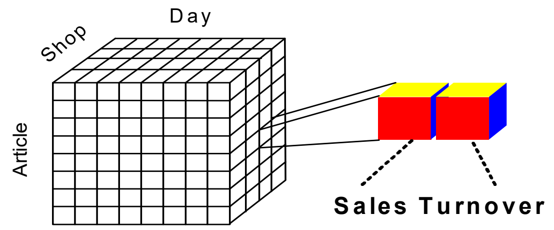
Multi-dimensional Data: Cubes

- A data cube is a multi–dimensional space of facts
- *Cubes, Hypercubes*



Data Cube Instances

- Cube domain:
  - $\mathrm{dom}(C) = ((\mathrm{dom}(G_1) \times \cdots \times \mathrm{dom}(G_n)) \times ((\mathrm{dom}(M_1) \times \cdots \times \mathrm{dom}(M_m))$
- Instance:
  - All cube cells from the cube domain
  - Not: subset of existing facts as in the relational model

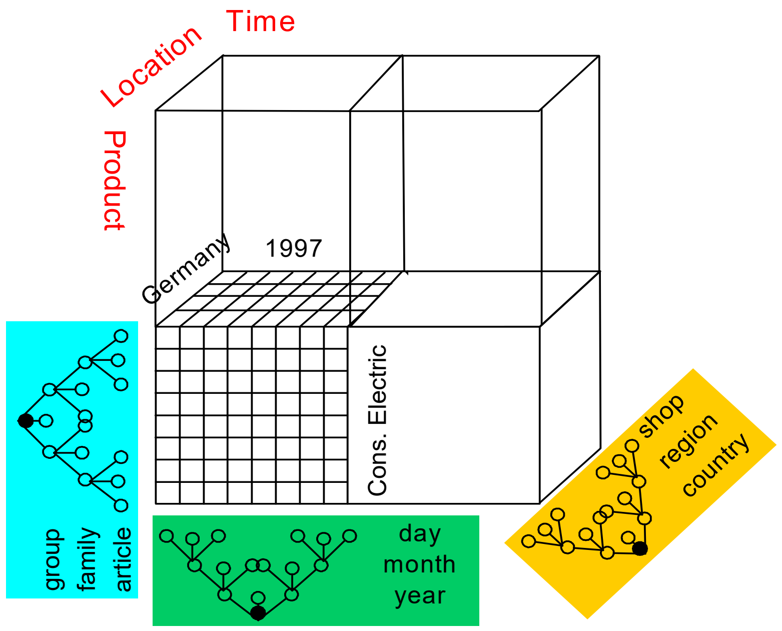


- „cube“ is more like a metaphor
  - Rarely all cells are really present (in the sense of facts that occurred in the real world)
  - Implementation as null or 0

Content

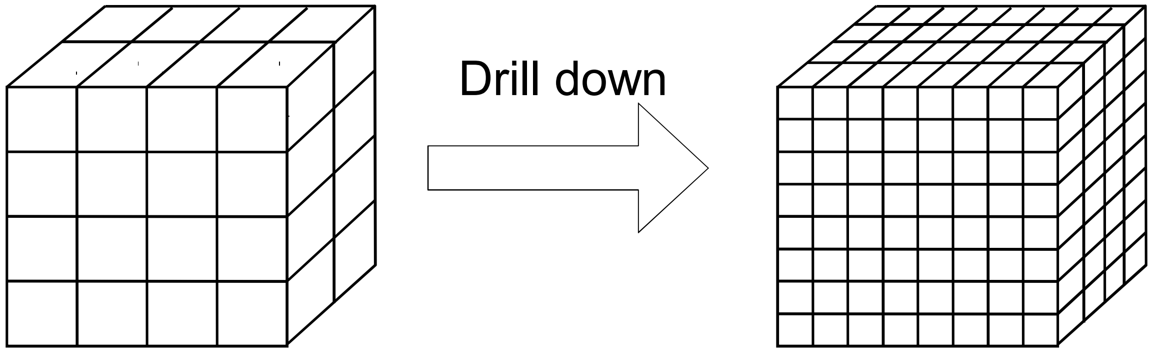
Multi-dimensional Operators: Slice and Dice

- Cuts a cube
  - Specification of a hierarchy node
  - Example: sales in Germany
- Selection of a sub–cube
  - Specification of nodes in the dimension hierarchies
  - Sales for Consumer Electric in Germany in 1997



Multi-dimensional Operators: Navigation

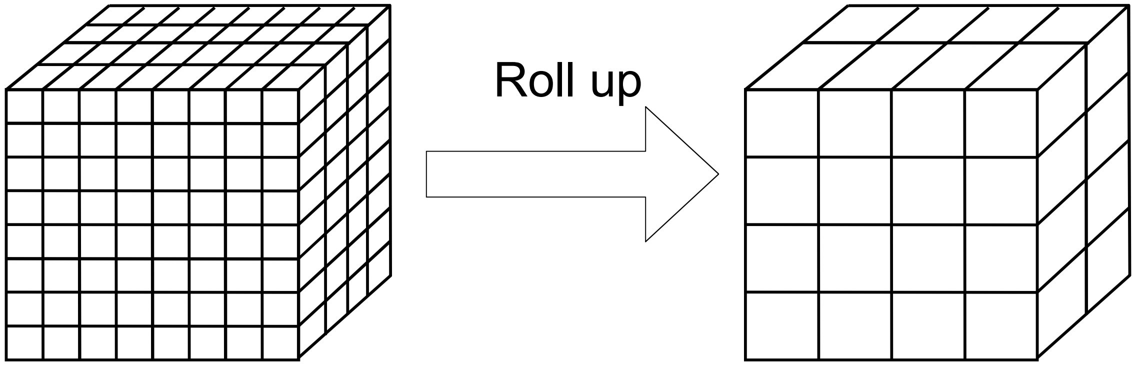
- Drill–Down
  - Start at coarser granularity
  - Navigate to a representation with finer granularity (i.e., more detailed)
  - Sums will be broken up in partial sums
  - Example: Sales [Year, Kanton] –> Sales [Quarter, Kanton]



Multi-dimensional Operators: Navigation (2)

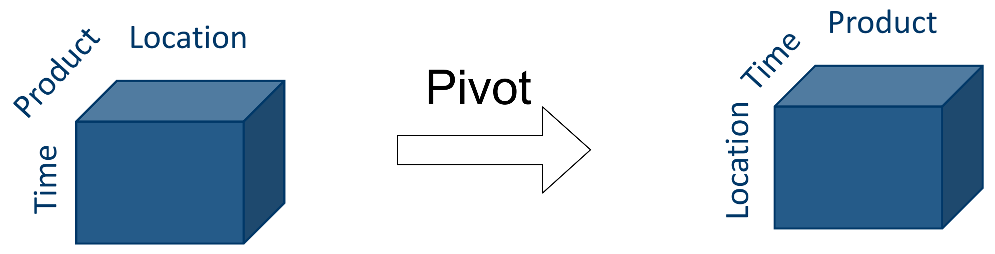
- Roll-up
  - Start at finer granularity
  - Navigate to a representation with coarser granularity (i.e., less detailed)
  - Possibly first expansion of the data
  - Example: Sales per year and state (2005, CA)

    → Sales per year and country (2005, USA)



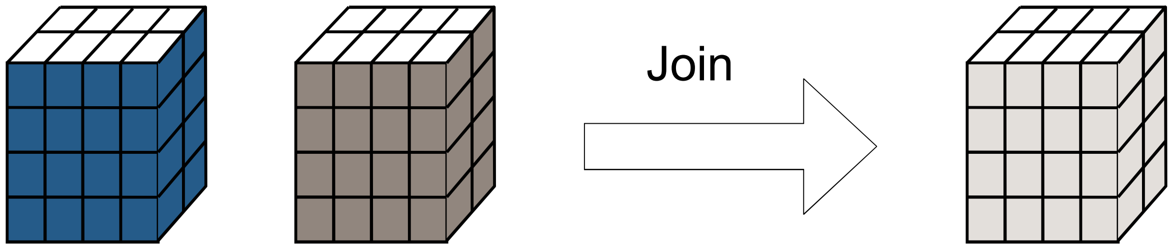
Multi-dimensional Operators: Pivot

- Changes orientation of the dimensions
- Rather an operation on presentation level, not of the data model



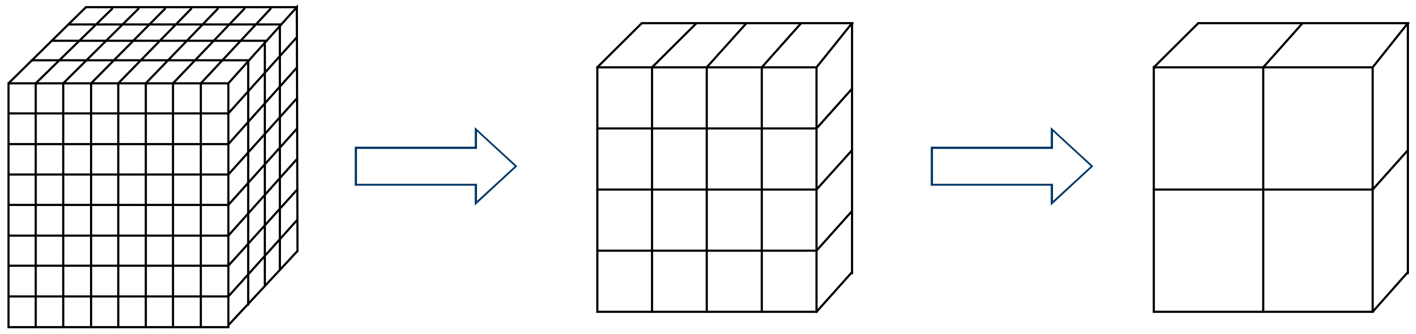
Multi-dimensional Operators: Join

- Combines two cubes into a new one
- $(G, M_1) \otimes (G, M_2) = (G, M_1 \cup M_2)$
- Example: Sales Cube $\otimes$ Price Cube
- Possibly granularities need to be adjusted first



Multi-dimensional Operators: Aggregation

- Aggregation happens (implicitly) whenever granularity changes (e.g., rollup)
  - By default, sums are computed
  - Other standard aggregation operators are possible as well: average, minimum, maximum, count
  - Further, application-specific calculations
  - A multi-dimensional DBS can be used as a calculation engine



Multi-dimensionale Data Models: Summary

- Several multi–dimensional data models
- Dimensions & hierarchies, measures, facts, cubes
- Cubes as the basis for ...
  - Analysis
  - Calculations

- Some (subtle) restrictions
- „Cube" is a concept, there are different instantiations on logical and physical level

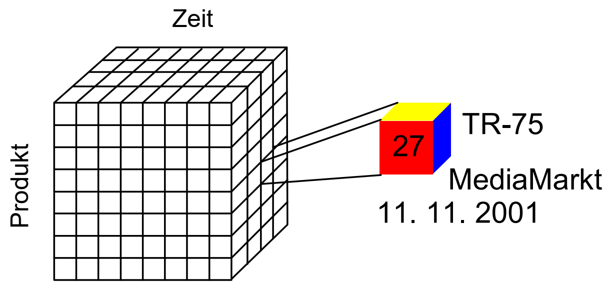
Content

Logical Design

- Derive logical schema from conceptual one
- express logical schema with the means of the logical (meta) data model (!)
  - logical data model can be relational or multidimensional
- Requirements (to logical (meta) data models and DBMSs
  - adequate representation of data (types)
    - dimensions and hierarchies
    - facts
  - operators support analysis adequately
  - DBMS aspects
    - large volumes of data
    - performance
    - multi user access
    - security, …

Relational Mapping of Multidimensional Structures

- ☞ Storage of multidimensional data
  - in relations
  - only existing cube cells
- ☞ Separation of structure and content (unlike as in cubes)
  - central fact table(s)
  - dimension tables
  - facts reference dimension tables
  - fact table contains measures

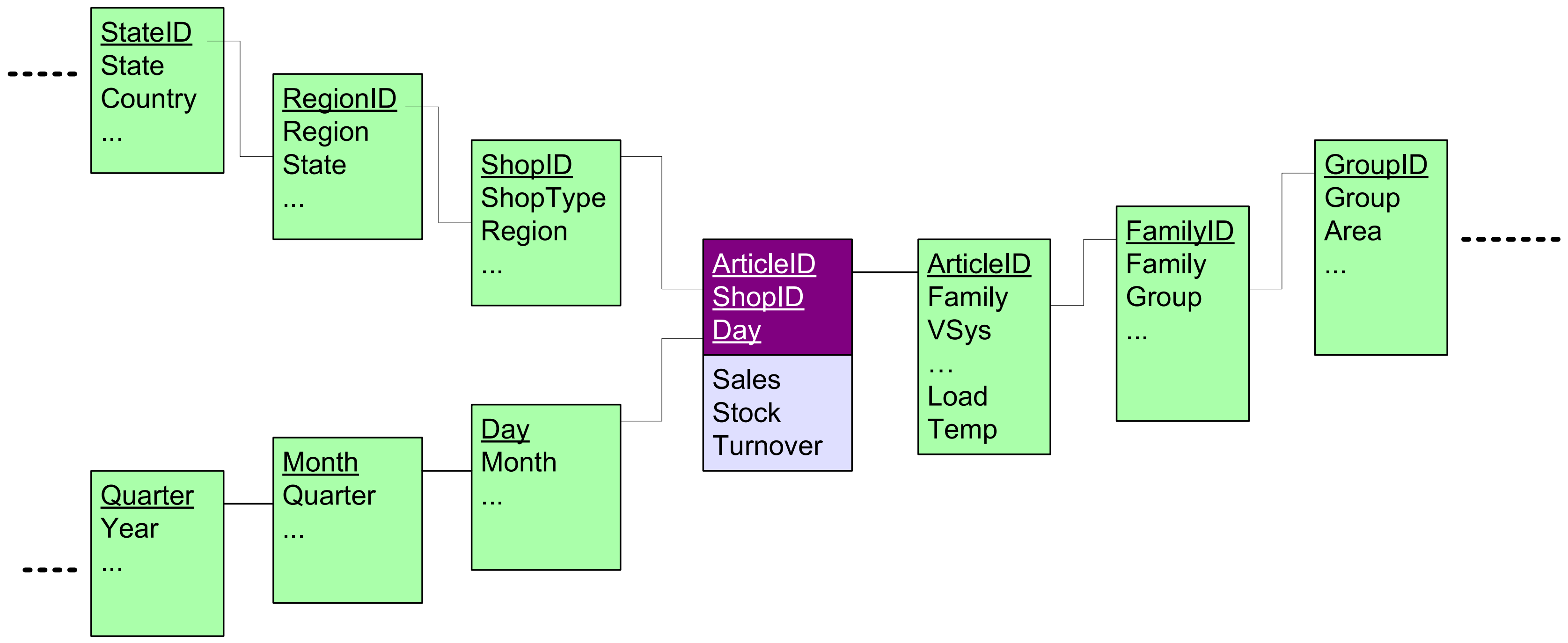


Produkt	Geschäft	Zeit	Verkäufe
TR-75	MediaMarkt	11. 11. 2001	27
AC300	ProMarkt	23. 12. 2001	39
...	...	...	...

Snowflake Schemas

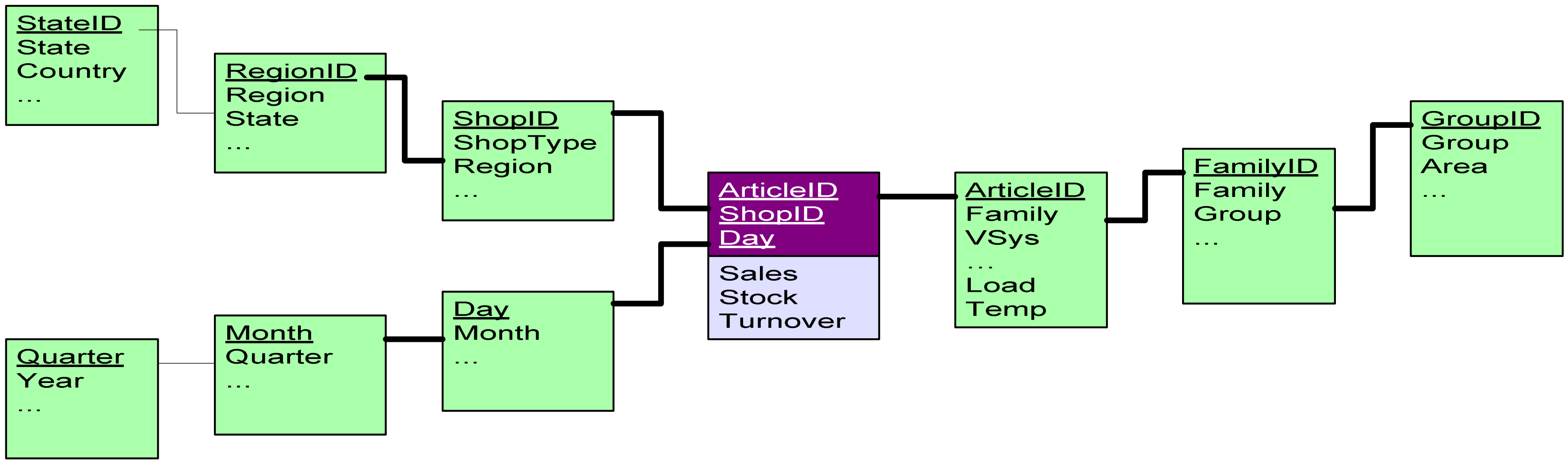
- (central) fact table, dimension tables
- dimension tables are normalized
- n dimension hierarchy levels (within a dimension) $\rightarrow$ n dimension relations for this dimension
- foreign key/primary key relationship between fact and dimension tables: fact table references dimension table representing most granular hierarchy level
- primary key of fact table is composed out of foreign keys (i.e., primary keys of dimensions)
- foreign key/primary key relationship between dimension tables (representing the hierarchy levels)

Snowflake Schema: Example



Snowflake Schemas: Implications for Queries

- Example: Sum of sales by region, by month, by product group
- 7 join operations:
  - 3 between fact table and (most granular) dimension tables
  - 4 for rollup to higher hierarchy levels



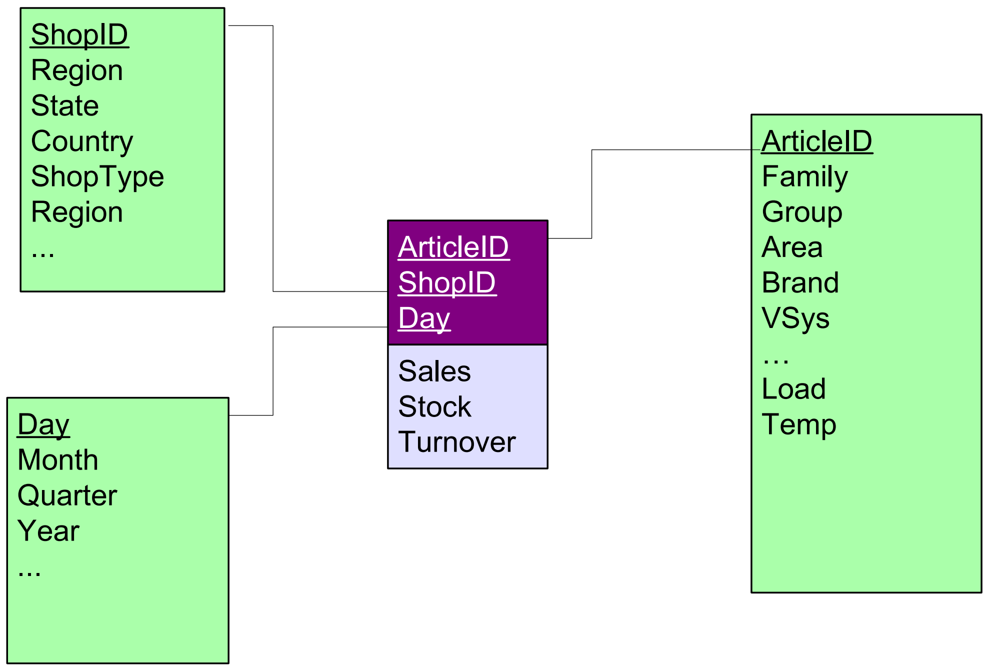
Snowflake Schemas: Properties

- correct update of dimension tables is easier
  - because of normalization
  - normalization avoids update anomalies
- schema design is easier
  - depending on conceptual design, snowflake design is obtained as a result of an automatic transformation
- performance is worse
  - many joins are required
  - one join per involved dimension + one join per ″rollup″
- no redundancy
  - again, because of normalization

Star Schemas

- central fact table
- one table per dimension, regardless of the number of hierarchy levels within the dimensions
- denormalized dimension tables
- foreign key/primary key relationship between fact and dimension tables: fact table references dimension tables
- primary key of fact table is composed out of foreign keys (i.e., primary keys of dimensions)

Star Schema: Example

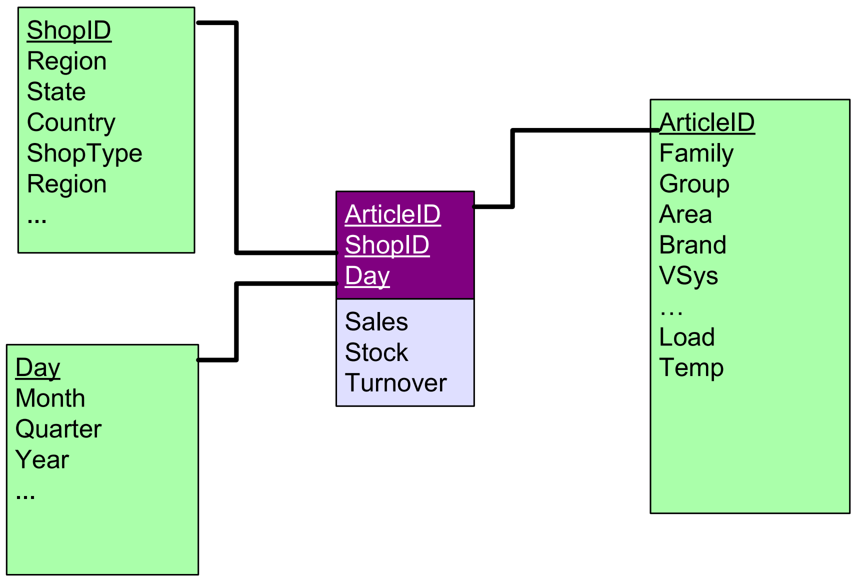


Star Schemas: Properties

- Redundancy
  - because of denormalization
  - update anomalies are possible
- Performance
  - smaller number of joins
  - because of denormalization
- intuitive

Star Schemas: Implications for Queries

- Example: Sum of sales by region, by month, by product group
- 3 join operations
  - 3 between fact table and dimension tables
  - no joins for rollup to higher hierarchy levels



Comparison: Star vs. Snowflake

- Advantages of Star schemas

☺ faster query evaluation

- no difference between star and snowflake when for each dimension only the most granular hierarchy level is needed
- but: analytic queries are more often than not on a high (coarse) aggregation level
- one join per rollup

☺ simpler structure

- important for the generation of OLAP-queries

☺ additional data volume is negligible

☺ updates to classifications are rare

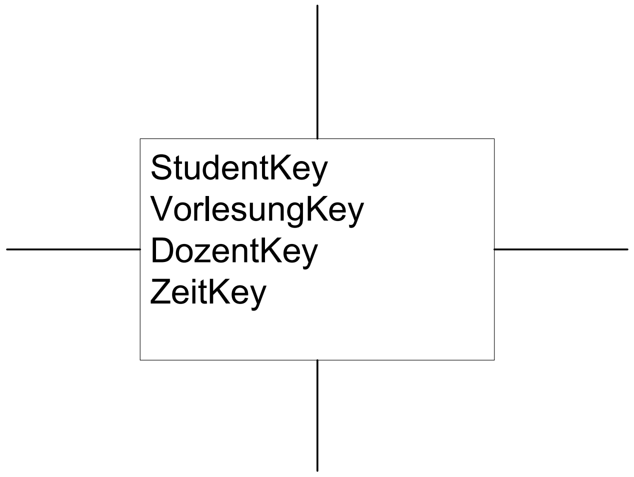
- update anomalies are less of an issue

Special Cases: Factless Fact Tables

- Fact table contains dimensions and measures
- Special case:
  - no measure

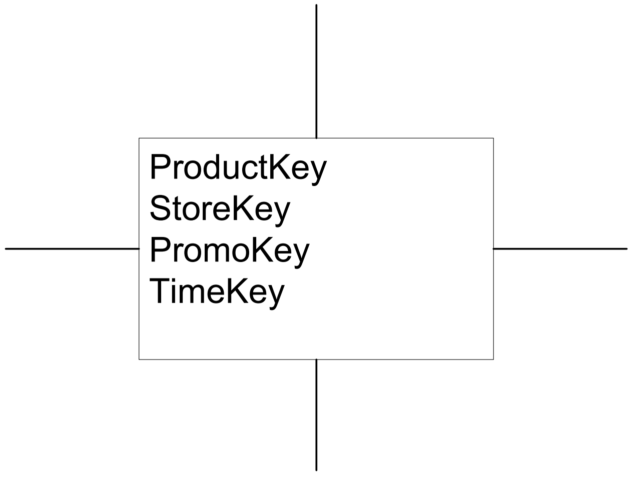
➔ existence statement without numeric attributes

- recording of events
- Example: student attends course held by a lecturer on a certain day



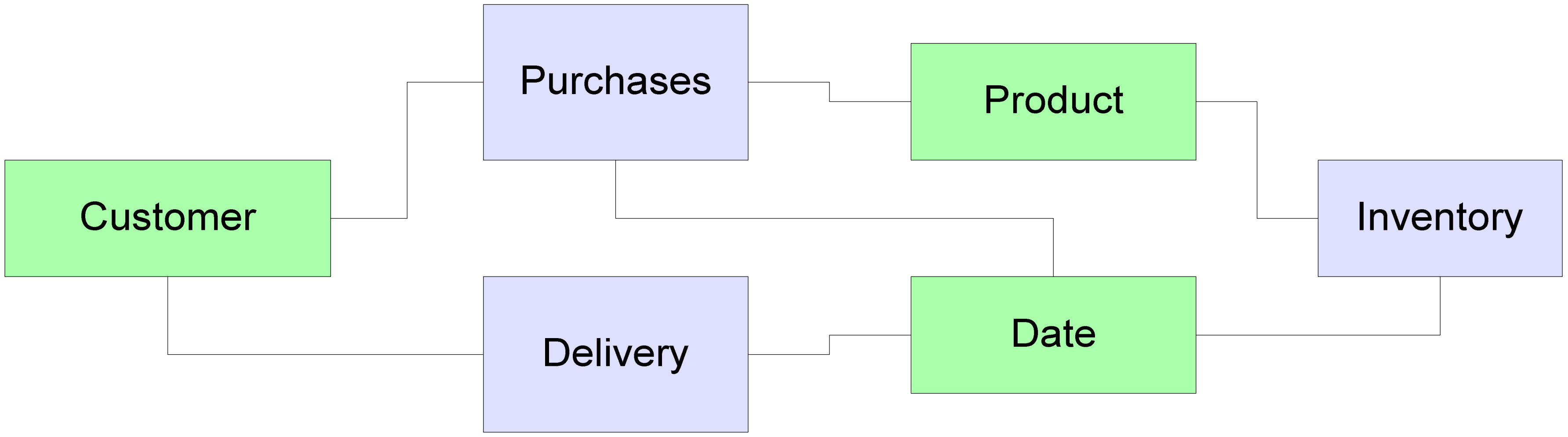
Factless Fact Tables (2)

- Non-occurrence of events
- "Coverage table" contains possible events
- fact table contains actual events
- non-occurred events are obtained by computing the table difference
- Example: which products have not been sold despite promotion campaigns?



Galaxy Schemas

- multiple independent fact tables
- fact tables share some (but not all) dimension tables

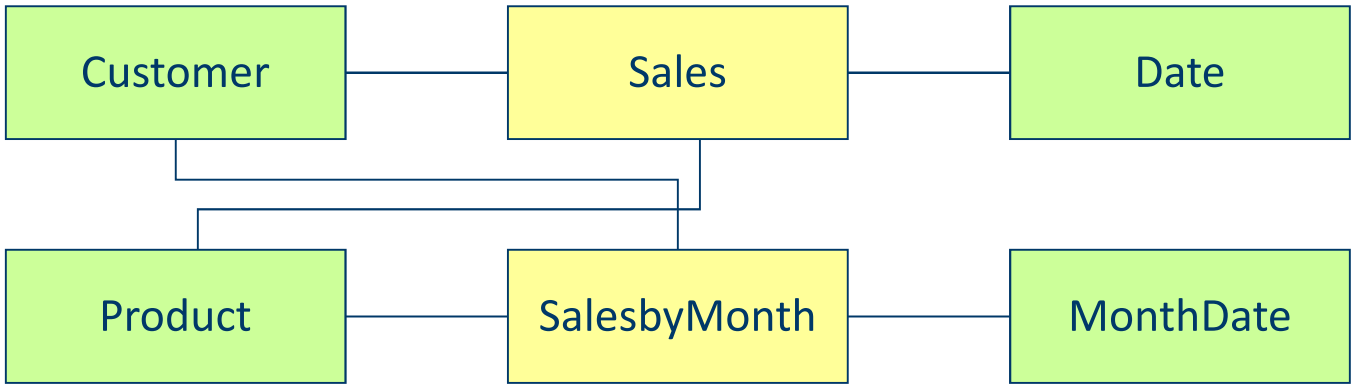


Fact Constellation

- Measures are needed on different aggregation levels within an analysis scenario
- Computation of aggregated measures always possible, but possibly expensive
- possible optimization: materialization of aggregate values
- option 1: addition of aggregate values to fact table
  - additional attribute for each aggregate value
  - additional discriminator attribute
  - correct linkage to dimension tables?

Fact Constellation (2)

- Option 2: separate table for aggregate values (**fact constellation**)
- „smaller“ fact table
- also smaller dimension tables possible
- more efficient queries against detailed and aggregated data

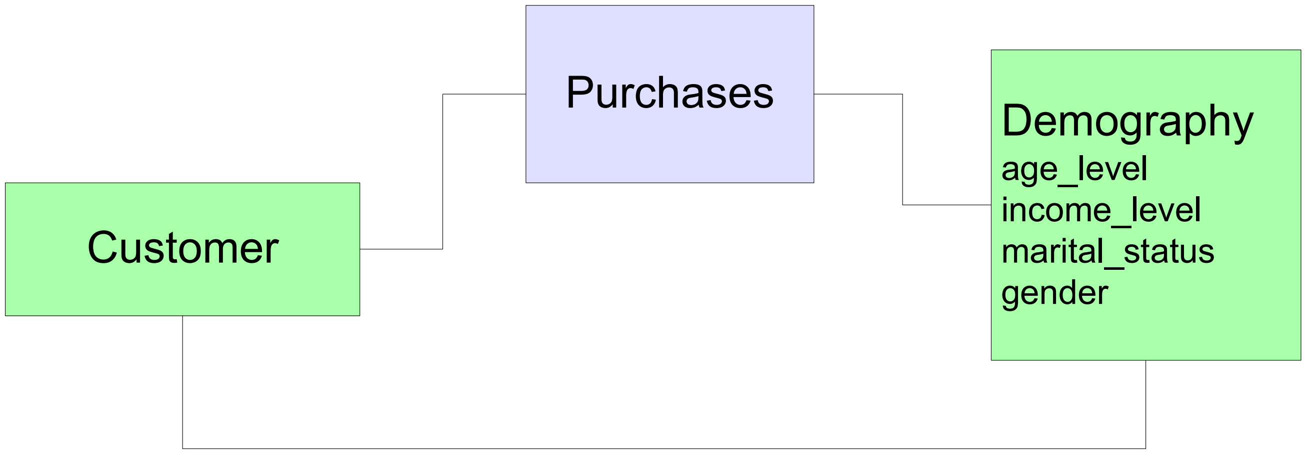


Monster Dimensions

- common problem: very big dimension tables
- many attributes, many rows
- example: product dimension
  - 10K–100K product articles, 20–50 attributes (warehouses, retail)
- example: customer dimension
  - up to 100M customers, many attributes
- demography: often queried, rather frequent modifications
  - example income, marital status

Monster Dimensions

- demographic mini–dimensions
- extract of the demographic information out of customer dimension
- better performance
- demographic information of a customer no longer in customer dimension



Versioning of Dimension Tables

- Problem: changes in dimension tables
- Examples:
  - Customers change address, name, marital status, ...
  - products are abandoned
  - product properties (classification) change
- Dimensions are yet "rather constant"
  - slowly changing dimensions (SCD)
- changes are permitted in OLTP applications, because there the current state is of (sole) interest
  - old values are overridden, update-in-place
- changes are problematic in a DWH, they can lead to misleading, erroneous analysis results

Versioning of Dimension Tables (2)

- Versioning of Dimension Tables: Solutions

1. Update-in-place
   - old values are lost
   - simple to implement
   - no history is available
   - misleading analysis results are possible
   - SCD Type 1

Customer_key	name	status
1234	Joe Cool	~~single~~ married

Versioning of Dimension Tables (3)

2. Addition of a ″status attribute″
   - current and previous value
   - rarely done
   - useful?
   - SCD Type 3

Customer_key	name	old_status	current	effective_date
1234	Joe Cool	single	married	04-10-1992

Versioning of Dimension Tables (4)

3. Addition of a version number
   - clean modeling
   - querying becomes more difficult
   - temporal history not obvious
   - version number must be added to fact table, too
   - composite foreign/primary key

Customer_key	version	name	status
1234	001	Joe Cool	single
1234	002	Joe Cool	married

Versioning of Dimension Tables (5)

4. extension of primary key with version number
   - questionable from a modeling point of view
   - unique identification and reference, for instance for aggregation?

Customer_key	name	status
1234001	Joe Cool	single
1234002	Joe Cool	married

Versioning of Dimension Tables (6)

5. explicit addition of validity intervals
   - eases historical and time–based queries
   - at least one of the timestamps must be added to primary key
   - or referential integrity cannot be maintained
   - SCD Type 2

Customer_key	name	status	valid_since	valid_till
1234	Joe Cool	single	01-01-1900	04-09-1992
1234	Joe Cool	married	04-10-1992	

Summary

- DWH application development
- "standard" approach for DWH itself (ER-based)
- conceptual data mart design
  - no established notation (a la ER)
  - patterns of logical design are used for conceptual one ("dimensional modeling")
- relational implementation
  - Star and Snowflake schemas
  - Galaxy schema & Fact Constellation Schema
  - Factless Fact Tables
  - Slowly Changing Dimensions

APPENDIX

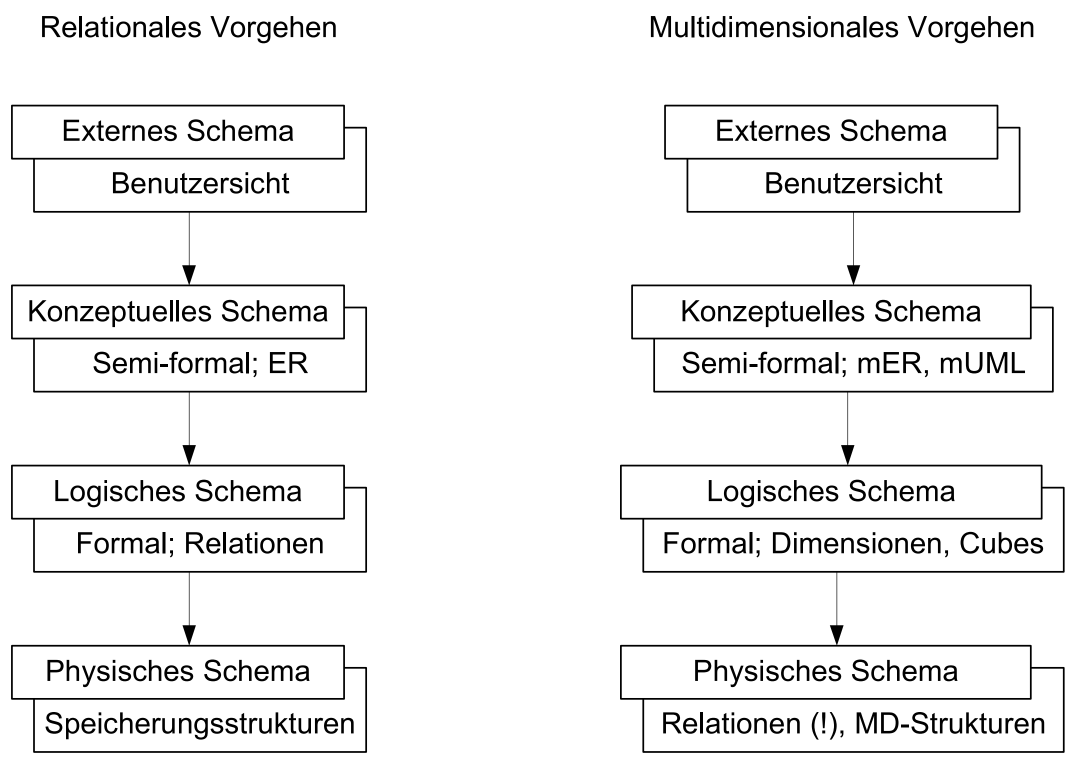
Content

Conceptual Design for Data Marts

Multidimensional concepts: basic support considered in DWH and OLAP systems

- ☹ no common, widely accepted model
- ☹ no standardized modeling language and notation
- ☹ no common formalization
- ☺ methods and guidelines based on experience

Relational vs. Multidimensional Schema Design



Conceptual Design: Variants

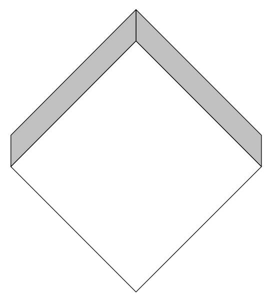
- Use of existing notations (and methods, tools)
  - possibly with adapted semantics
  - currently prevalent approach in practice
- Extensions of existing notations
  - ER
  - UML
  - typically academic/research proposals
- Development of new notations
  - typically academic/research proposals
  - ☺ can be designed to optimally meet requirements
  - ☹ additional notation

Notations: M-E/R

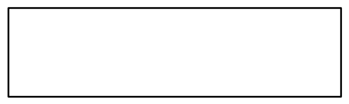
- Extension of the ER model
- additional entity type representing dimension hierarchy level
- additional relationship types:
  - relationship between dimension hierarchy levels
  - relationship between dimensions and hierarchies

Notations: M-E/R

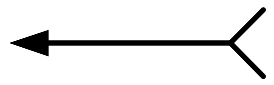
Fakten-Beziehungstyp



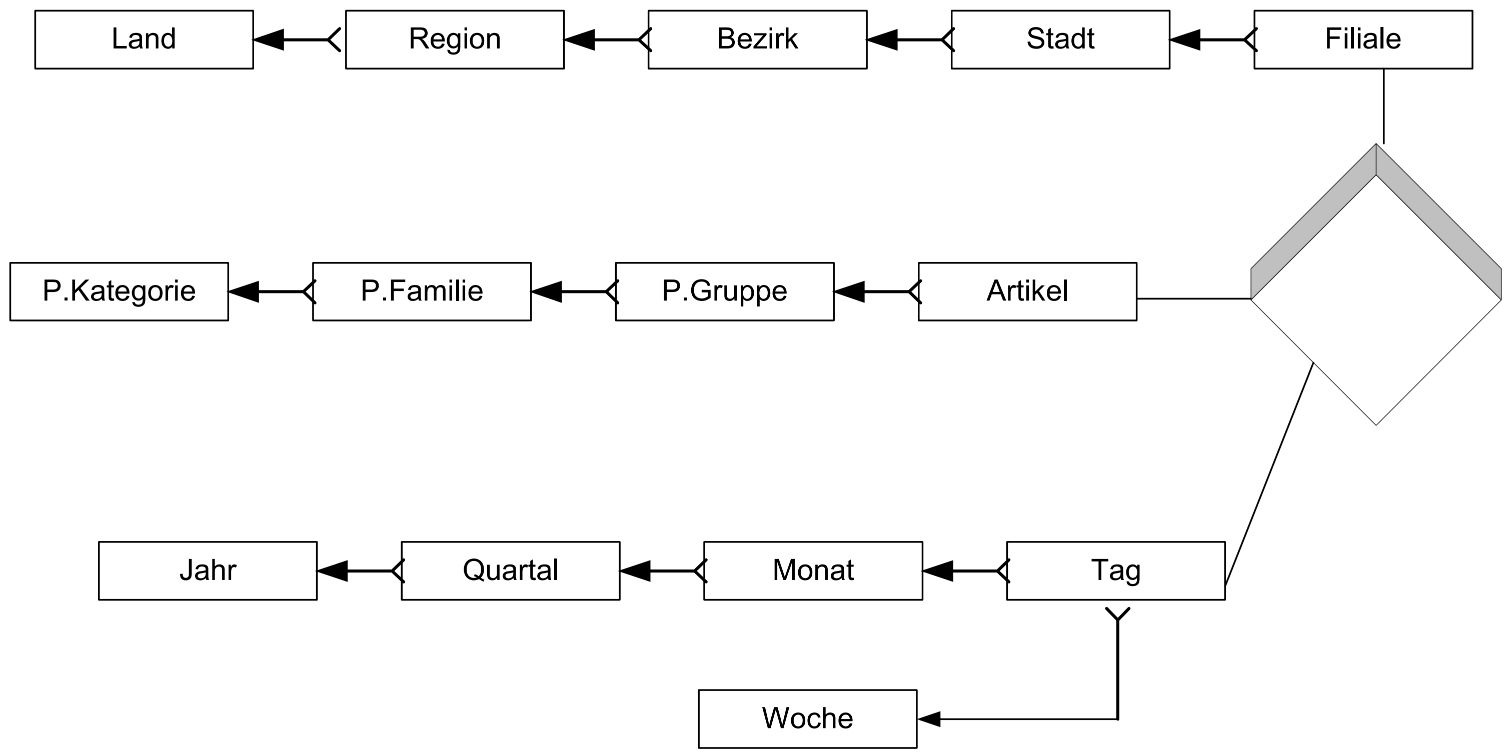
Dimensionsebenentyp



Rolls-Up-Beziehungstyp



M-E/R: Example


Land
Region
Bezirk
Stadt
Filiale
P.Kategorie
P.Familie
P.Gruppe
Artikel
Jahr
Quartal
Monat
Tag
Woche

Gold: Notation

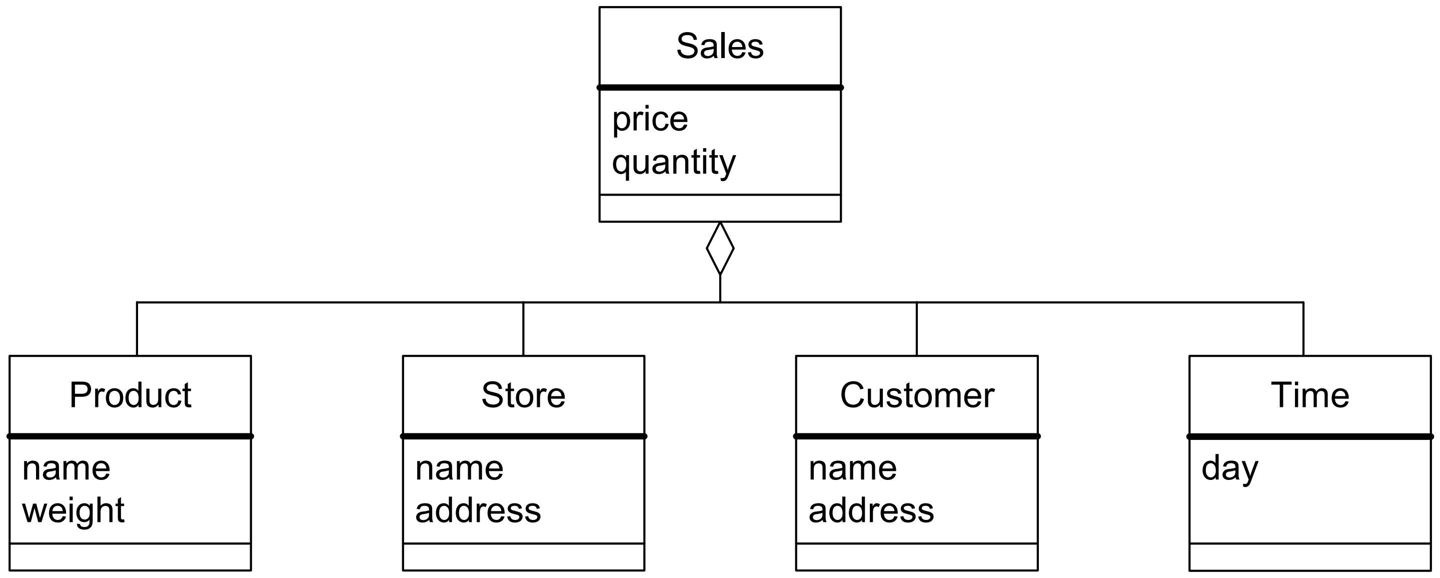
- developed on the basis of the Unified Modeling Language (UML)
- Approach by Trujillo et al., prototypically implemented in the ″Gold″ tool

Usage of UML constructs for the modeling of:

- fact classes
- dimensions
- relationships between
  - facts and dimensions
  - dimension hierarchy levels
- cardinalities

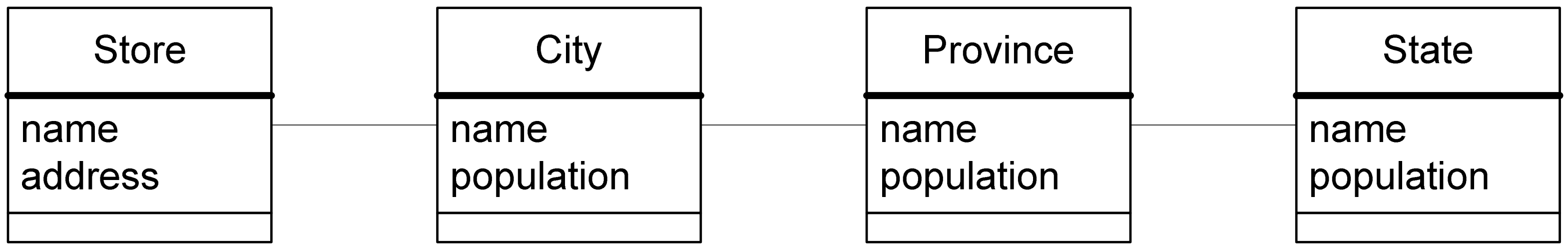
Gold: Notation (2)

- UML classes for the modeling of fact and dimension classes
- UML–aggregation for the modeling of the relationship between facts and dimensions



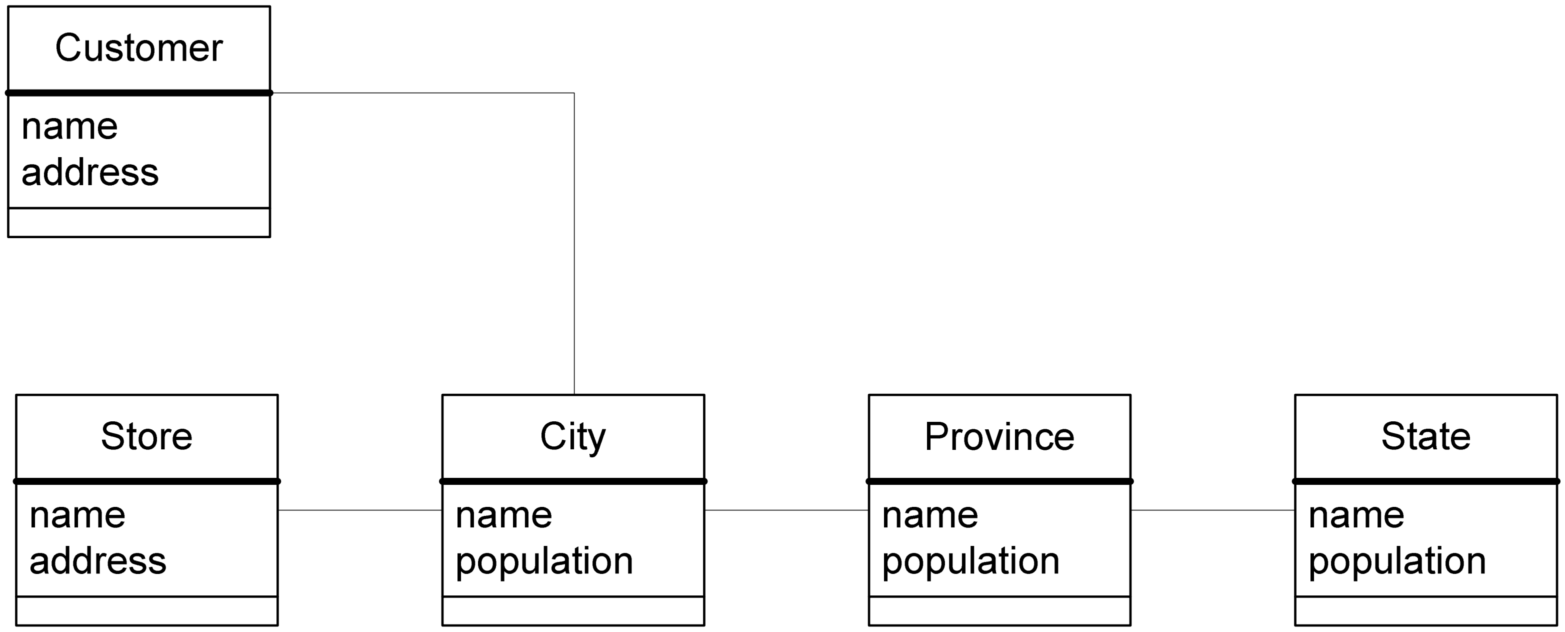
Gold: Notation (3)

- dimension hierarchies
- general associations used for the modeling of relationships between dimension classes (dimension hierarchy levels)



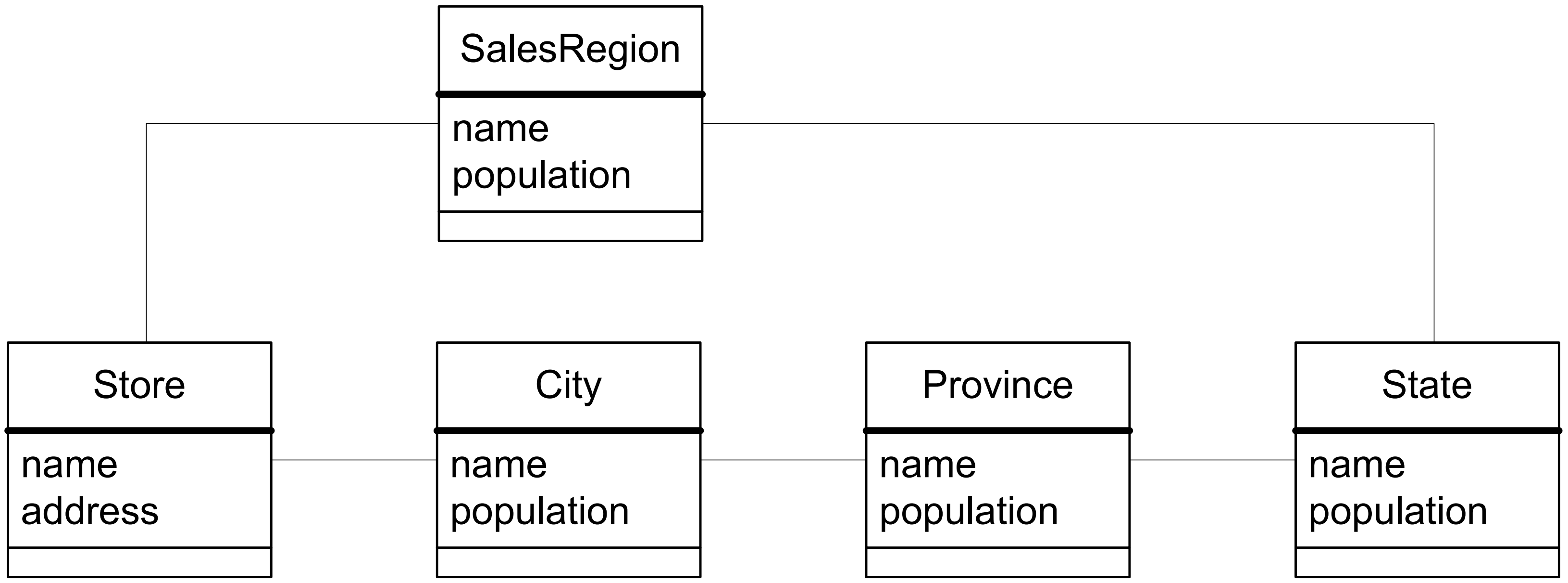
Gold: Notation (4)

- Gold supports shared hierarchies



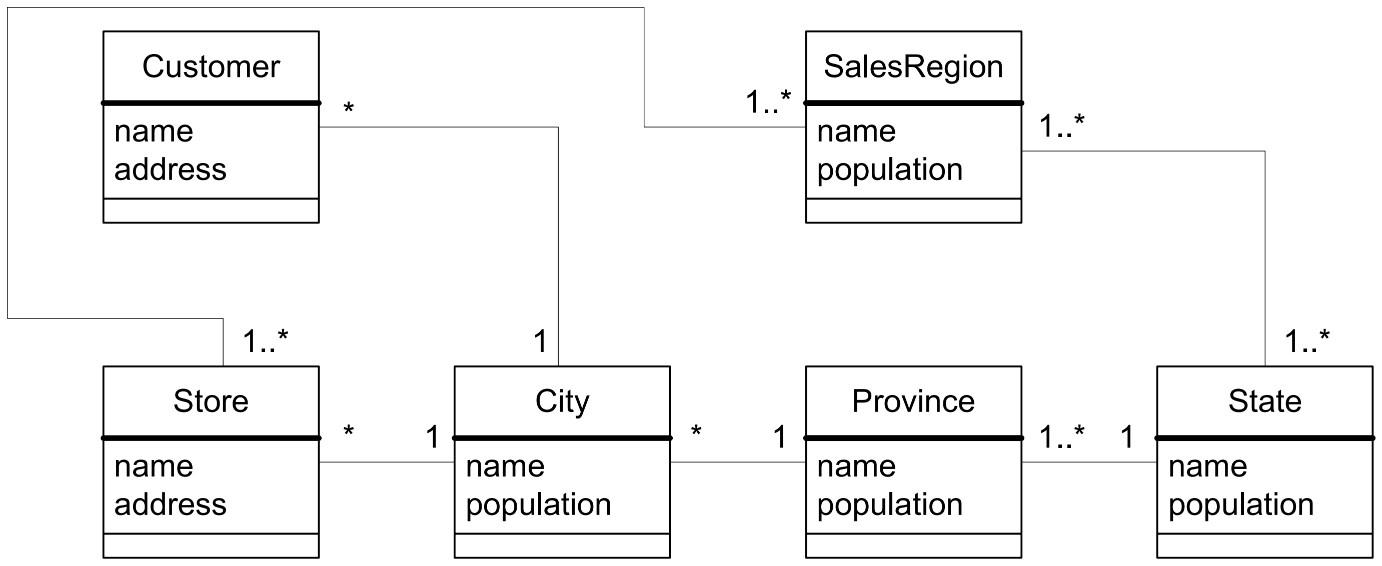
Gold: Notation (5)

- Gold supports parallel hierarchies



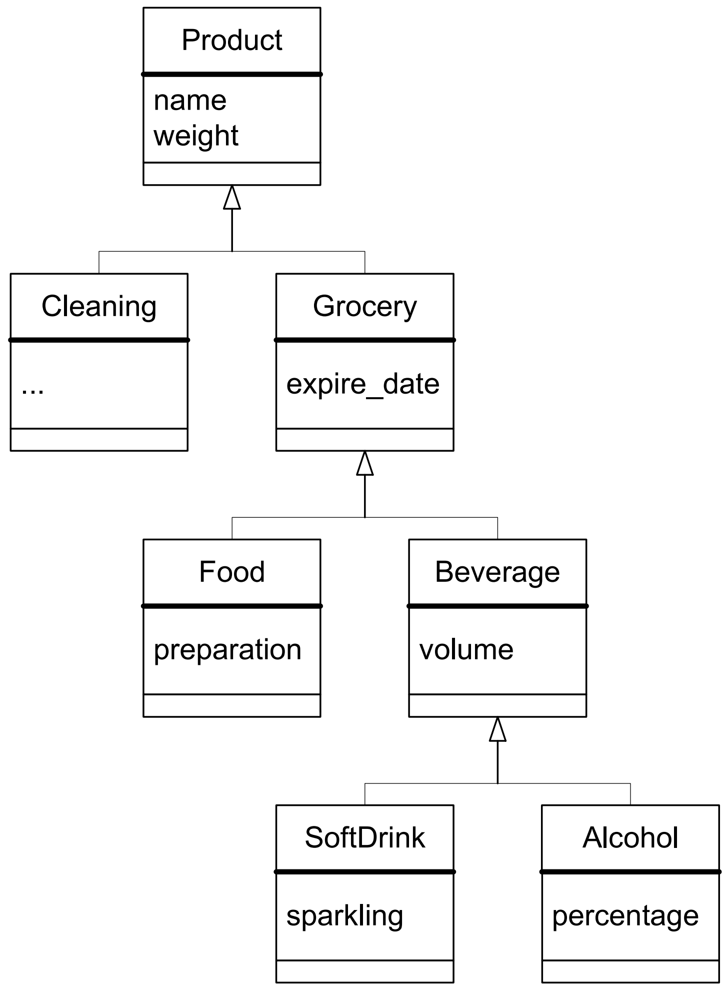
Gold: Notation (6)

- relationship properties: Cardinalities
- m:n relationship between facts and dimensions
- Completeness: annotation of relationships



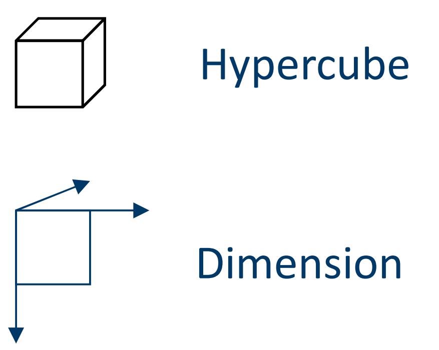
Gold: Notation (7)

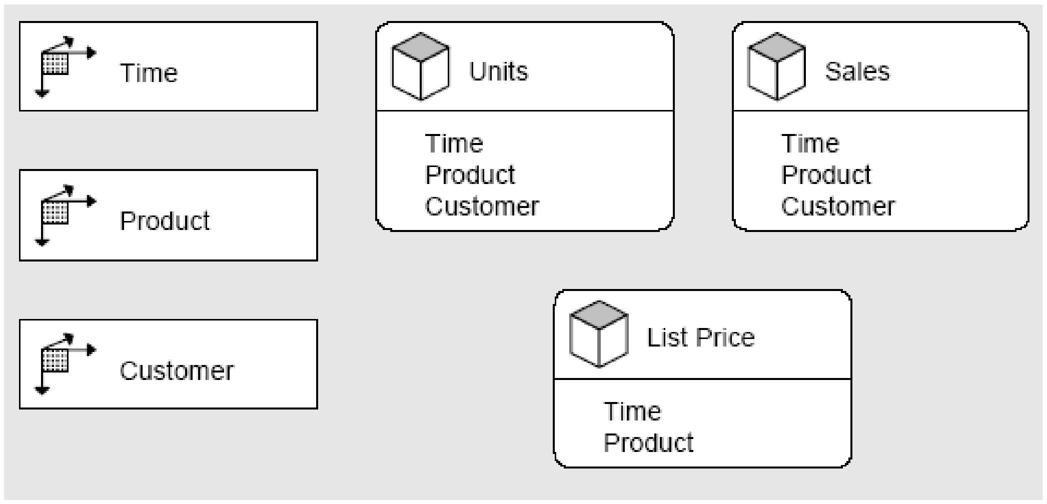
- Specialization of dimension classes using UML specialization
- problem: which circumstances should be modeled as a classification hierarchy, which as a specialization hierarchy



Adapt

- Application Design for Analytical Processing Technologies
- Trademark of Symmetry Corp.
- intended as a conceptual model for OLAP-Applications
  - Star schemas presume implementation
  - ER and "dimensional Modeling" are biased towards relational implementations

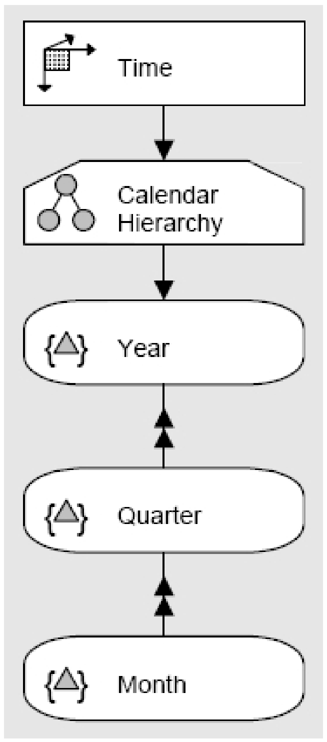


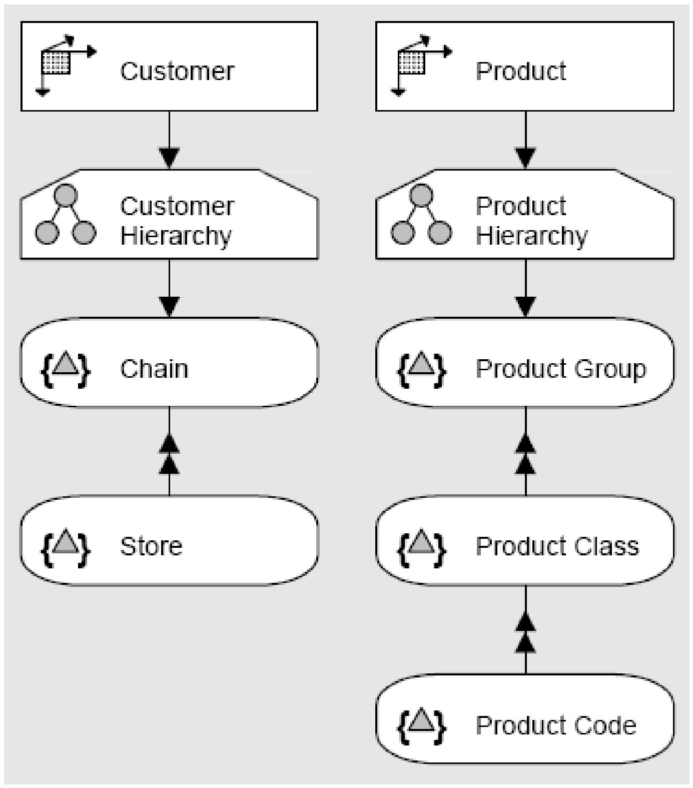


Adapt: Modeling of Hierarchies

Hierarchy (within a dimension), consisting of levels

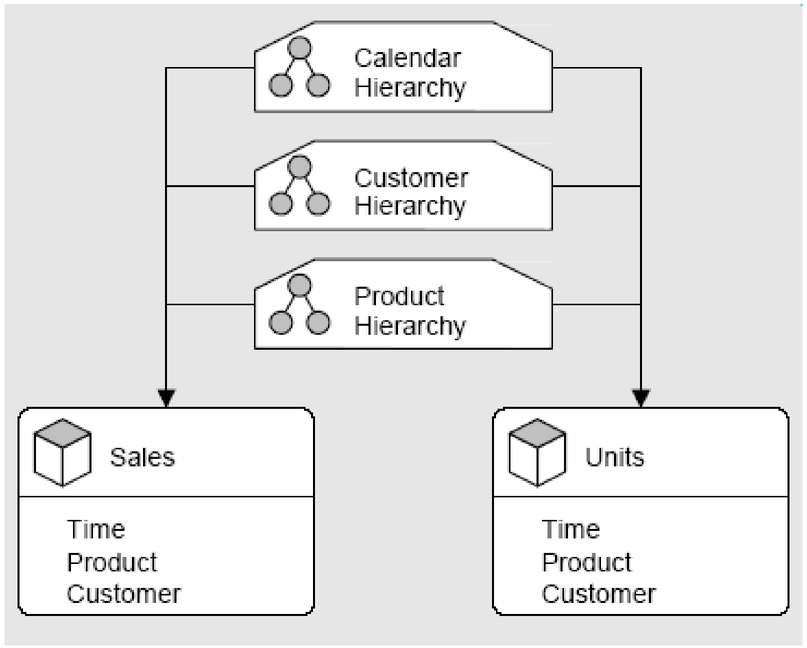
{△} Level





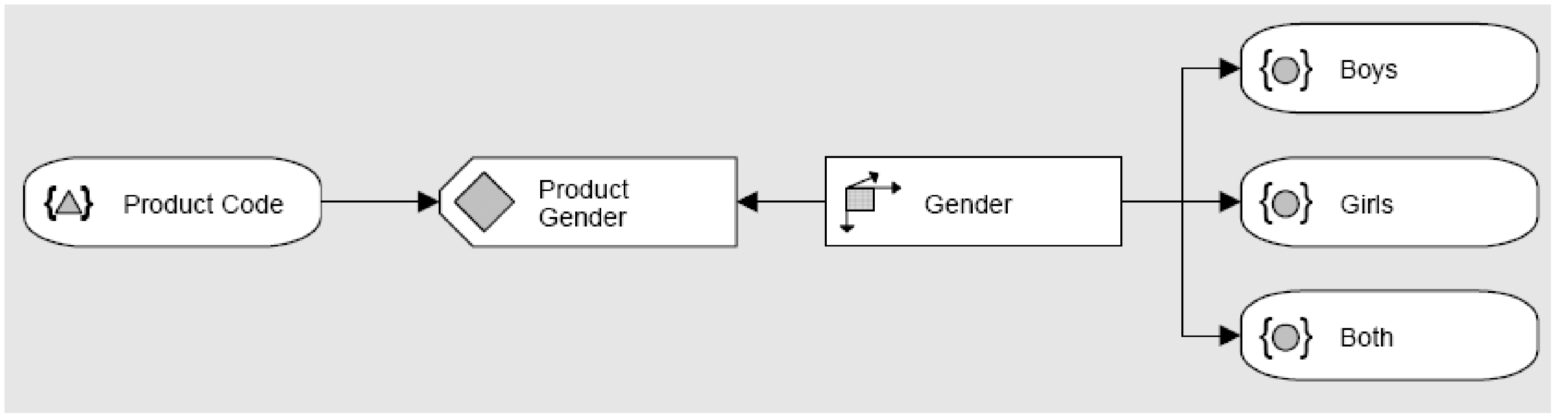
Adapt

- Combination of dimensions and hierarchies into hypercubes



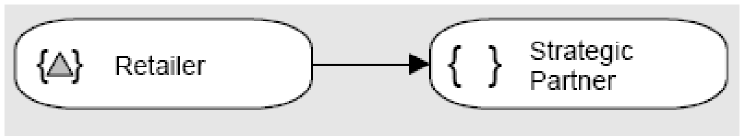
Adapt: Modeling of dimension details

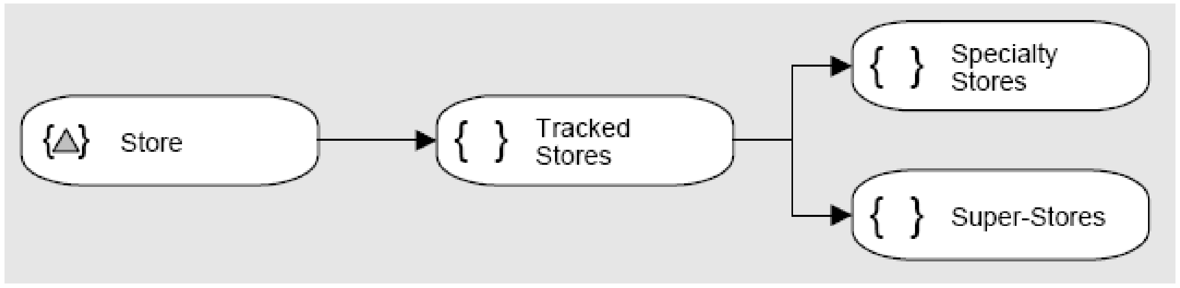
◇	Attribute
{●}	Member



Adapt: Modeling of Dimension Scopes

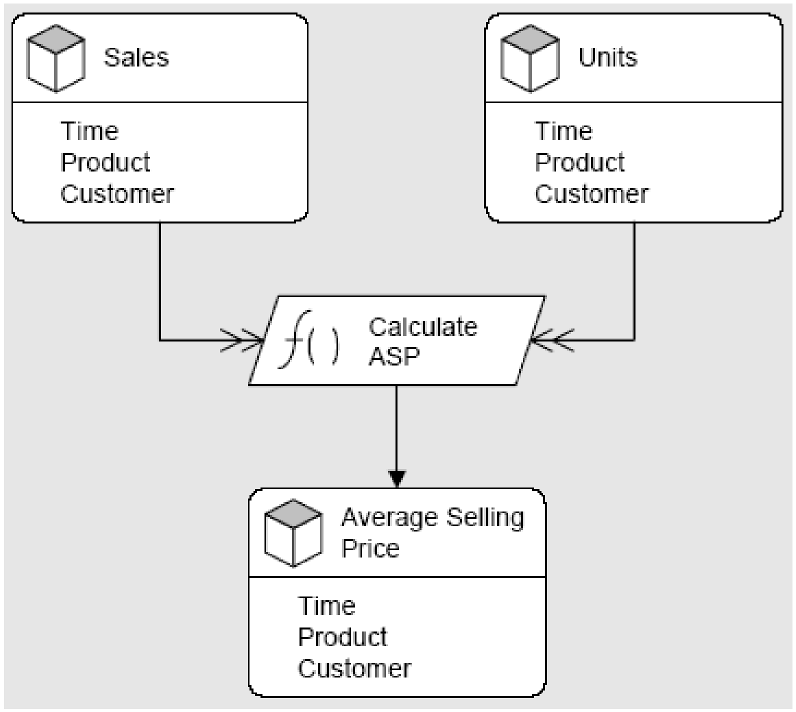
{ } Scope: set of dimension elements



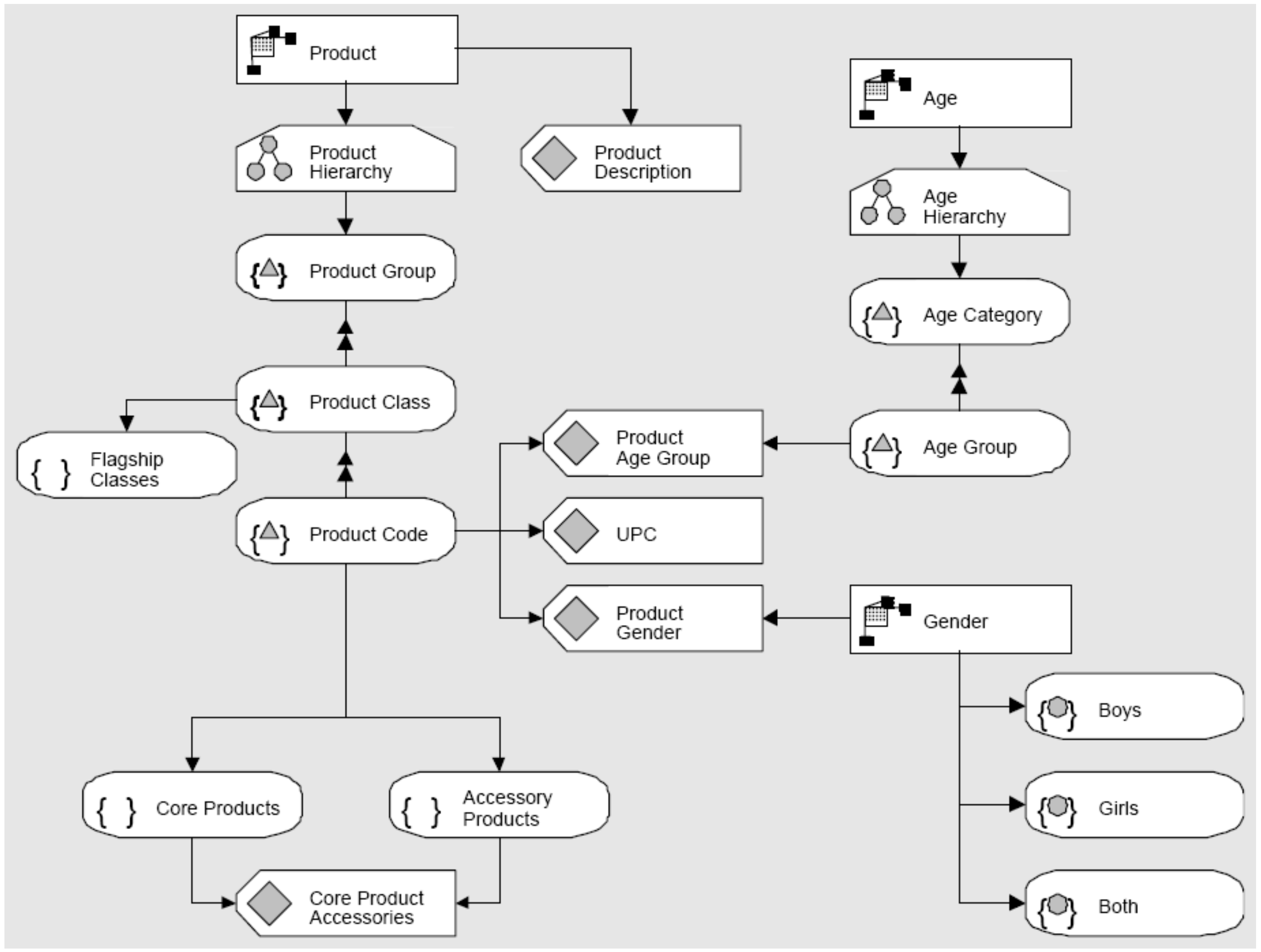


Adapt: Modeling of Calculations

ƒ( )   algebraic processes model calculations



Adapt: Sample Product Dimension


Product
Product Hierarchy
Product Description
Product Group
Product Class
Flagship Classes
Product Code
Product Age Group
UPC
Product Gender
Core Products
Accessory Products
Core Product Accessories
Age
Age Hierarchy
Age Category
Age Group
Gender
Boys
Girls
Both