Data Warehouse - Multi-dimensional Data Modeling and Operations

Databases and Information Systems

Fabian Panse

panse@informatik.uni-hamburg.de

University of Hamburg

Acknowledgements

These slides are based on slides provided by

- Prof. Dr. Erhard Rahm
  University of Leipzig
  `http://dbs.uni-leipzig.de/`

Overview

- **Basics**
  - Facts, dimensions, cube
  - Cuboid / aggregation grid (deutsch: Aggregationsgitter)
  - Hierarchical dimensions / hierarchies of concepts
- **Cube operations**
- **Multi-dimensional representation (MOLAP)**
  - MDX
- **Relational representation of multi-dimensional data (ROLAP)**
  - Star schema
  - Variants: Snowflake-, Galaxy schema
  - Queries: Star Join, Roll-Up, Drill-Down
  - SQL Operators: CUBE, ROLLUP and GROUPING SETS

Facts

- **Also: measures, measured facts, operating numbers**
- **Fact is a (numerical) measure used for statistical purposes**
  - Usually economic measures (e.g. revenue, profit or earning power)
  - Complex relationships between facts possible
- **Facts can be associated with descriptive attributes**
  - e.g. units, domains, calculation rules
- **Types of facts**
  - Additive facts: additive aggregation possible w.r.t. all dimensions
  - Semi-additive facts: additive aggregation only possible w.r.t. some specific dimensions (e.g. current account balances which are additive for all accounts, but non-additive for several days)
  - Non-additive facts: no additive aggregation possible (e.g. average values, percentages)

Types of Fact Tables

- **Cumulative:**
  - Describes what has happened over a specific period of time
  - Contains typically only additive facts
  - Example: Sales table

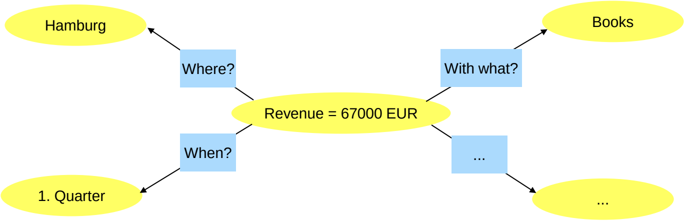
Date	Store	Product	Sales_Amount
10.01.16	HH-City	T-Shirt A	123
10.01.16	HH-Altona	T-Shirt A	21
...	...	...	...
12.01.16	HH-City	T-Shirt A	99

- **Snapshot:**
  - Describes the state of things in a particular instance of time
  - Contains often semi-additive and/or non-additive facts
  - Example: Account table

Date	Account	Current_Balance	Line_of_Credit
10.01.16	1000021	104.23	0%
10.01.16	1000024	-25.98	5.2%
...	...	...	...
12.01.16	1000024	245.78	0%

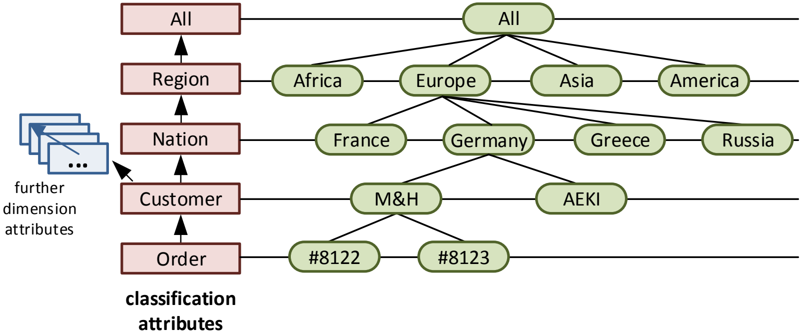
Source of information:
https://www.1keydata.com/datawarehousing/fact-table-types.html

Dimensions



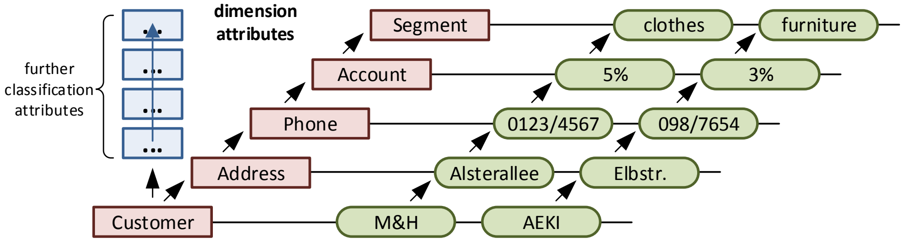
- **Numerical value of a fact without a semantic reference is meaningless**
- **Dimensions put facts in relation to characteristics/objective criteria**
- **Dimension: usually finite data type (e.g. enumeration)**
  - Example: Set of all products, regions, customers, time periods
  - Dimension element: element/instantiation/value of a dimension
  - Attributes: classification/category attributes (including a primary attribute) as well as "dimension attributes" (additional descriptive characteristics, e.g. product color/weight, address or phone number of a customer)

Classification Attributes (Example)



This graphic is from the book “Datenbanktechnologie für Data-Warehouse-Systeme” written by Wolfgang Lehner.

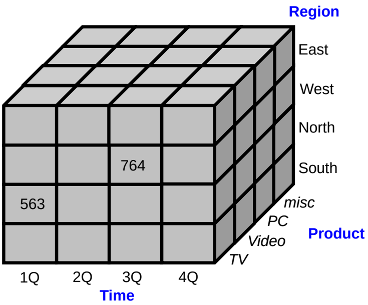
Dimension Attributes (Example)



This graphic is from the book “Datenbanktechnologie für Data-Warehouse-Systeme” written by Wolfgang Lehner.

Data Cube

- **OLAP Cube, Data Cube**
  - Dimensions: coordinates
  - Facts: cells in the intersections of the coordinates
- **Cube with respect to dimensions $D_1, \ldots, D_n$ and k facts:**
  - $W = \{(d_1, \ldots, d_n), (f_1, \ldots, f_k)\}$,
    dimension element d_i of D_i, $i = 1, \ldots, n$ and fact $f_j, j = 1, \ldots, k$
  - Unique cell address: $(d_1, \ldots, d_n)$
  - Cell content: $(f_1, \ldots, f_k)$
- **n: Dimensionality of the cube**
- **Alternative: k cubes each with one fact per cell (multi-cube)**
- **Typical 4 - 12 dimensions**
  - Time dimension is almost always included
  - Further standard dimensions:
    product, customer, seller, region, supplier/vendor, ...



Tabular Representation of Cubes

- **Direct implementation for 2 dimensions** (2D view on product $\times$ region)

Time = „Quarter 1"

	East	South	West
Product 1	30	100	100
Product 2	40	110	88
Product 3	17	70	50

Time = „Quarter 2"

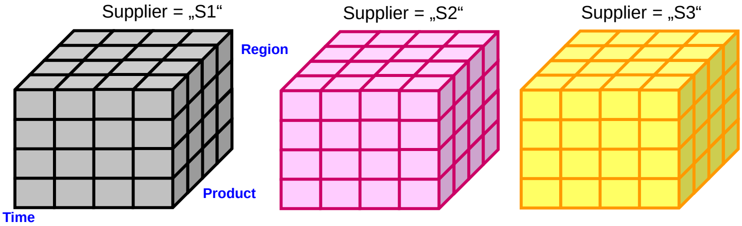
	East	South	West
Product 1	34	87	60
Product 2	32	80	103
Product 3	14	73	60

...

- **3 dimensions:** multiple 2D tables or nested tables or 3D cubes

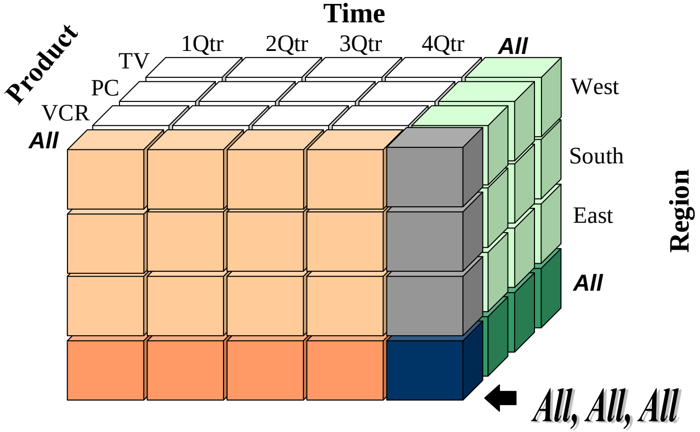
		East	South	West
Product 1	**Quarter 1**	30	100	100
	Quarter 2	34	87	60
Product 2	**Quarter 1**	40	110	88
	Quarter 2	32	80	103
Product 3	**Quarter 1**	17	70	50
	Quarter 2	14	73	60

Cube Representation

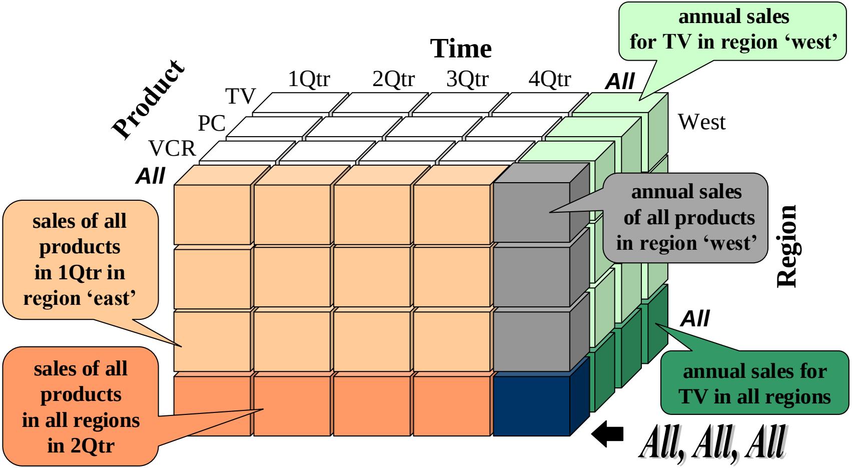


- **4D Cube can be represented as a set of 3D Cubes**
- **Aggregation:** from an n-dimensional cube a set of (n-1)-dimensional sub-cubes (also called cuboids) can be derived
  - Basic cuboid: n-dimensional cube
  - Apex cuboid (dt. Scheitel-Cuboid): 0-dim. aggregation over all dimensions
  - From the basic cuboid, we can derive cuboids with less dimensions
    $\Rightarrow$ Data cube corresponds to a lattice of cuboids
  - n-dimensional cube has 2^n cuboids including the basic cuboid (without considering dimension hierarchies)

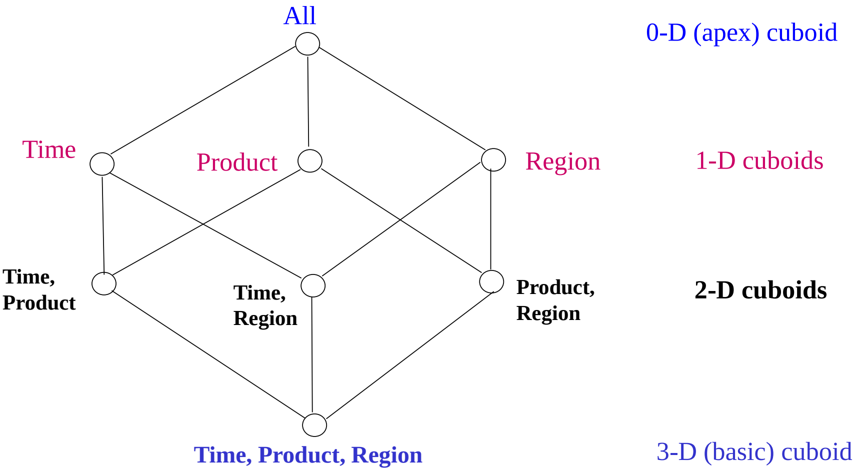
Data Cube: 3D Example with Aggregation



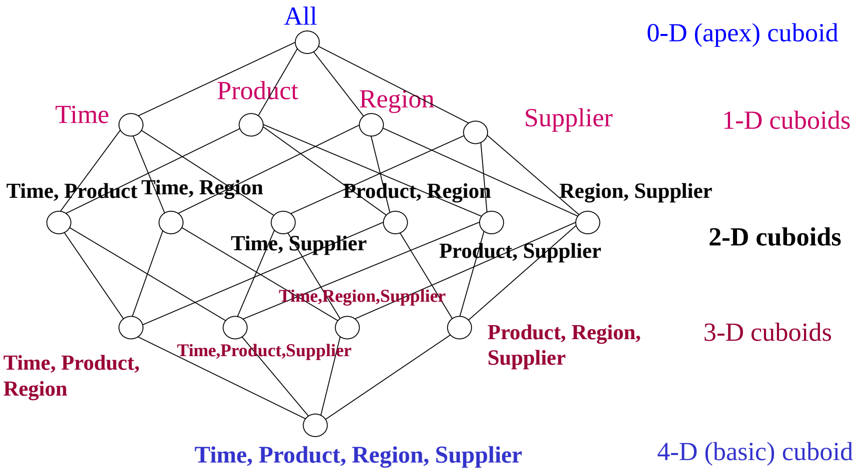
Data Cube: 3D Example with Aggregation



Corresponding Cuboids (Aggregation Grid)

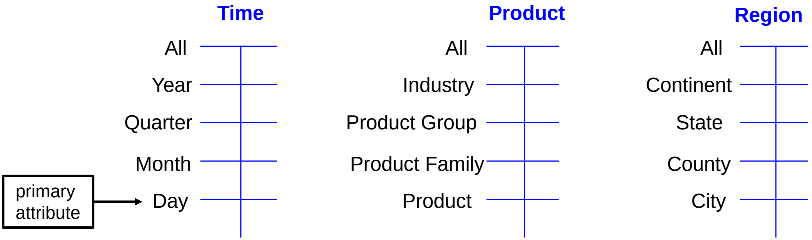


Cube: Lattice of Cuboids



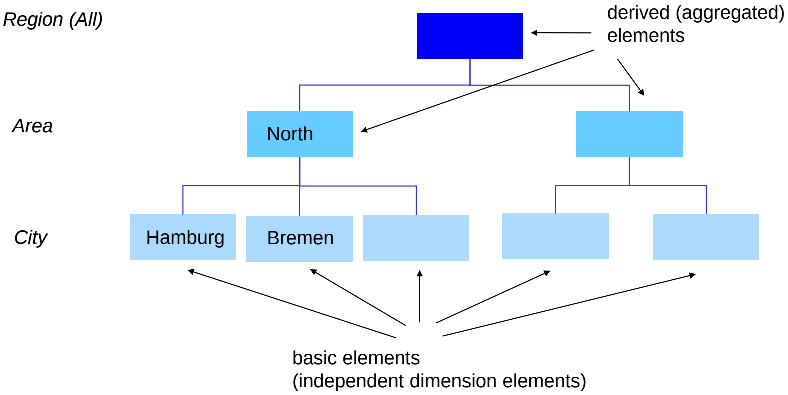
Dimension Hierarchies (Concept Hierarchies)

- **Often hierarchical relationships between dimension elements**
  - Top-level per hierarchy for all dimension elements (sum, top, All)
  - Primary attribute: lowest (precisest) level
  - Functional dependencies between primary attribute and classification attributes of higher levels
- **Examples:**



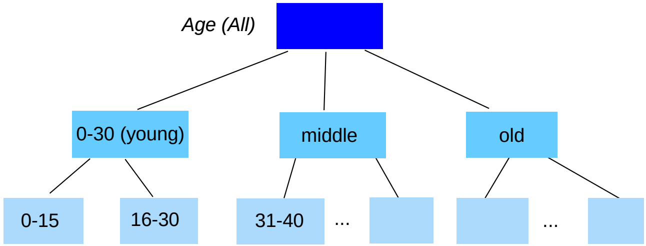
Example of a Concept Hierarchy (Region)

- **Simple hierarchy (per element maximal one superior element) vs. parallel hierarchy or semi-ordering** (e.g. day - week - month - year)

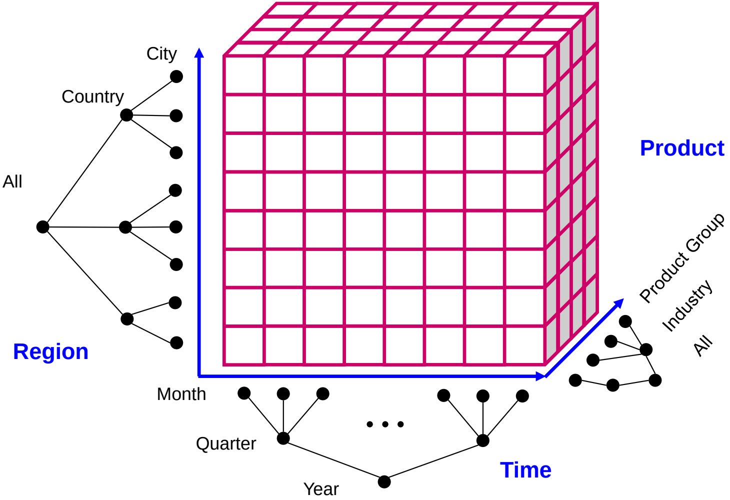


Concept Hierarchies

- **Hierarchies:** most often defined on schema level by classification attributes and their functional dependencies
- **Alternative variant:** hierarchization by grouping/discretization of values ("Set-grouping Hierarchies")
  - Appropriate classification on the basis of the given values by using calculation rules
  - Can be converted into a schema-based modeling by adding additional classification attributes

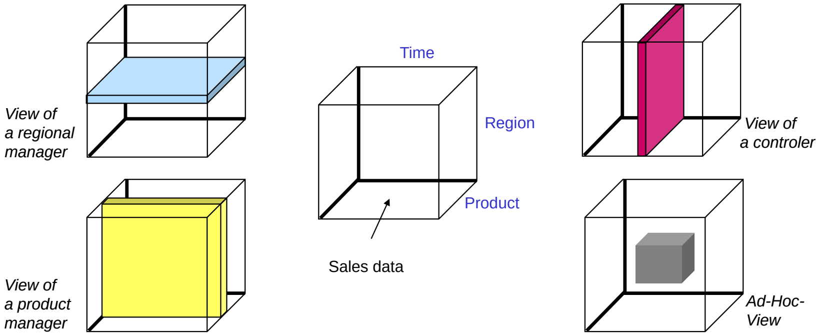


Cube with Hierarchical Dimensions

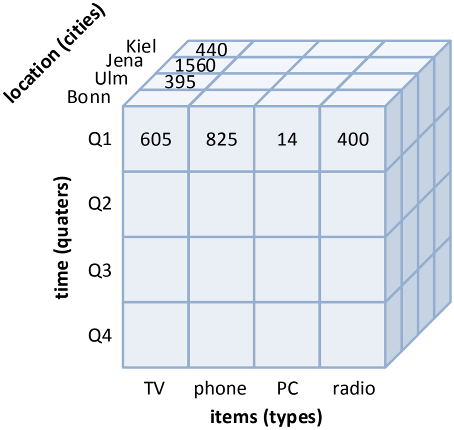


Cube Operations

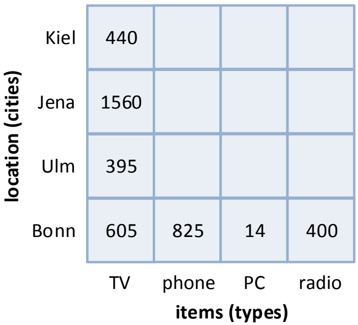
- **Slice:** cut out of a "slice" from the cube by choosing a single element for one of its dimensions (i.e. reduction of the number of dimensions by one)
- **Dice:** cut out of a "sub-cube" by discarding/restricting some dimensions
- Different multi-dimensional aggregations/groupings
- Pivot (switch of dimensions), sorting, top-k-queries, ...



Example: Slice



Slice
for time = „Q1"



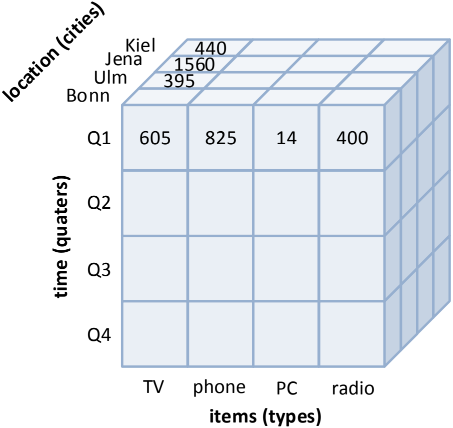
Example: Pivot

location (cities) \ items (types)	TV	phone	PC	radio
Kiel				
Jena				
Ulm				
Bonn	605	825	14	400

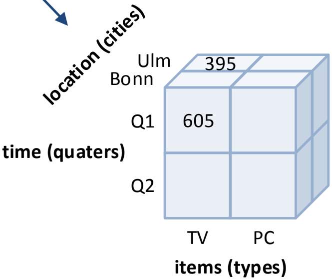
Pivot

items (types) \ location (cities)	Kiel	Jena	Ulm	Bonn
TV				605
phone				825
PC				14
radio				400

Example: Dice

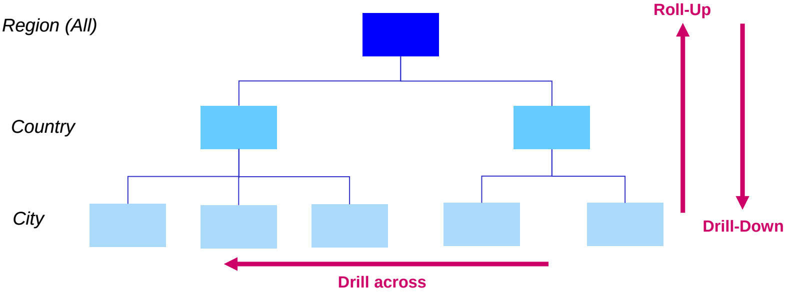


Dice for
(location = „Ulm“ or „Bonn“)
and (time = „Q1“ or „Q2“)
and (item = „TV“ or „PC“)



Navigation within Hierarchies

- **Drill-Down**
  - Navigation downwards in a hierarchy
  - Increasing the level of detail: from high consolidated/aggregated data to less consolidated/aggregated data
- **Roll-Up (Drill-Up)**
  - Navigation upwards in a hierarchy
  - From less consolidated/aggregated data to high consolidated/aggregated data

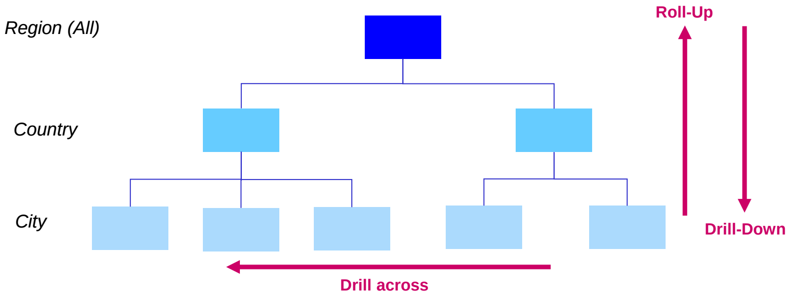


Navigation within Hierarchies

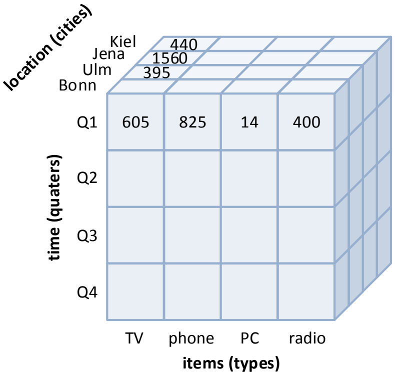
- **Drill-Across**
  - Navigation within a hierarchy level
  - Change of the considered dimension element

Example:

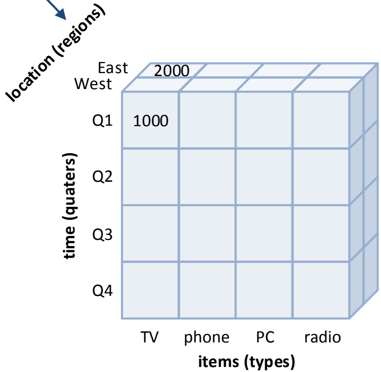
  - Given is a slice that represents the sales for different products at different years in the city 'Hamburg'
  - Drill-Across: Switch to the city 'Bremen'



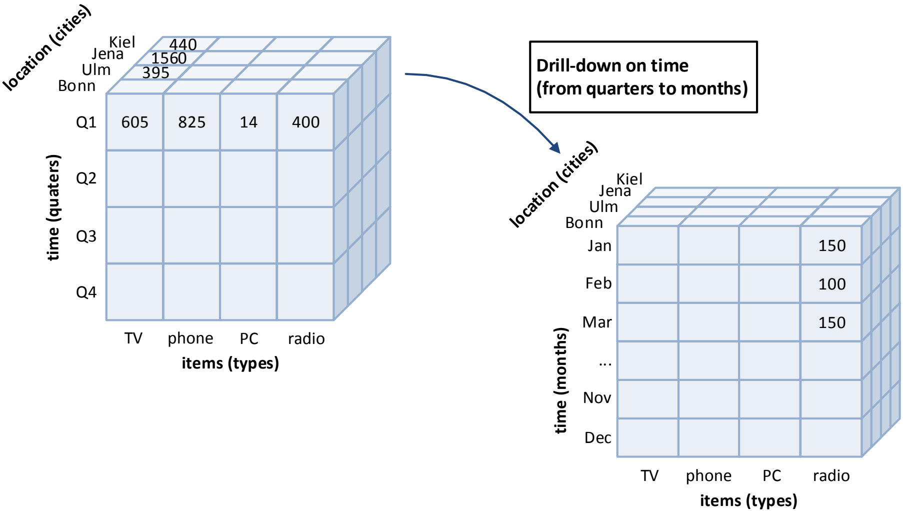
Example: Roll-Up



Roll-up on location (from cities to regions)



Example: Drill-Down



Drill-Down / Roll-Up (2D)

ProductGroup	*East*	*South*	*North*	*West*
Electronics	1800	1500	1450	2000
Toys	500	1700	600	1500
Clothes	1200	1200	400	1000

Drill-Down ↓ ↑ **Roll-Up**

Electronics	*East*	*South*	*North*	*West*
TV	800			
DVD-Player	650			
Camcorder	350			

Aggregation: 2D Example

- **Totaling (e.g. calculating the sum)**

ProductGroups	*East*	*South*	*North*	*West*	**All**
Electronics	1800	1500	1450	2000	6750
Toys	500	1700	600	1500	4300
Clothes	1200	1200	400	1000	3800
All	3500	4400	2450	4500	14850

- **Precalculation (Materialization) of the aggregated values for a quick response to aggregation requests**
- **Requires much memory and many updates in the case of many dimensions** ⇒ often only a small part of the required aggregated values is precalculated and the rest is calculated on demand
- **Example of an update:** Insert of a new product group, customer or year

Cube Size

- **Size of the basic cuboid**
  - Number of cells corresponds to the arithmetic product of the cardinalities of all dimensions ⇒ cell number results in $\prod_{i=1}^{n} |D_i|$
  - Example: $1,000$ days, $100,000$ products, 1 million customers ⇒ 10^{14} cells
  - Each additional dimension, e.g. region or supplier, leads to a strong increase of the data space
- **Precalculation of (aggregated) cuboids increases memory requirement**

Cube Size (2)

- **Size of a hierarchically aggregated cube**
  - Aggregation is possible for each dimension element of one of the higher hierarchy levels
  - Combination with each element of one of the hierarchy levels of the other $n-1$ dimensions
- **Number of cuboids of an n-dimensional cube:**
  - L_i: #Levels of dimension i (without top level), $T = \prod_{i=1}^{n}(L_i + 1)$

Time ($L_1 = 3$)		Product ($L_2 = 3$)		Costumer ($L_3 = 2$)	
All	1	All	1	All	1
Quarter	12	Industry	50	Customer Group	10,000
Month	36	Product Group	5,000	Customer	1 million
Day	1,000	Product	100,000		

$\Rightarrow\ T = 4 \times 4 \times 3 = 48$

without hierarchies: $L_i = 1, i = 1, \ldots, n \Rightarrow T = 2^n = 8$

Implementation of the multi-dimensional Model

- **Aspects**
  - Data storage
  - Formulation/evaluation of operations
- **MOLAP: Direct storage in multi-dimensional memory structures**
  - Cube operations are simple to formulate and can be evaluated efficiently
  - Scalability with respect to large data sets is limited
- **ROLAP: Relational storage in tables**
  - Efficient storage of large data sets
  - Query formulation is more complicated
  - Standard-SQL is not sufficient (only 1-dimensional grouping, ...)
- **HOLAP: Hybrid solution**
  - Relational storage of the detailed data, multi-dimensional access interface
  - Different combinations with multi-dimensional storage/evaluation of aggregated data
- **Precalculation of aggregations is often necessary to ensure an acceptable performance**

Multi-dimensional Data storage

- **Data storage with multi-dimensional matrix**
  - Direct implementation of the logical cube concept
  - Precalculation and storage of facts based on the cross-product of the domains of all considered dimensions
  - Fast and direct access to each fact based on its index position $(x_1, x_2, \dots, x_n)$

multi-dimensional (contingency table)

	Berlin	**Hamburg**	**Bremen**
TV	100	150	200
DVD-Player	50	170	150
Camcorder	20	120	100

relational

Product	**Region**	**Sales**
TV	Hamburg	150
Camcorder	Berlin	20
...	...	...

- **Queries:**
  - How many DVD-Player have been sold in Berlin?
  - How many Camcorder have been sold in total?

Multi-dimensional Data storage (2)

- **Multi-dimensional storage often leads to sparse matrices**
- **Example (sales per customer by region)**

REGION *multi-dimensional (2-dimenstional)*

Customer	*B*	*S*	*NRW*	*SH*	*BW*	*SA*	*MVP*	*HH*	*TH*
Customer 1	100	-	-	-	-	-	-	-	-
Customer 2	-	-	150	-	-	-	-	-	-
Customer 3	-	-	-	-	200	-	-	-	-
Customer 4	-	50	-	-	-	-	-	-	-
Customer 5	-	-	-	170	-	-	-	-	-
Customer 6	-	-	-	-	-	-	-	-	100
Customer 7	-	-	-	-	-	20	-	-	-
Customer 8	-	-	-	-	-	-	120	-	-
Customer 9	-	-	-	-	-	-	-	100	-

relational

Customer	**Region**	**Revenue**
Customer 1	B	100
Customer 2	NRW	150
Customer 3	BW	200
Customer 4	S	50
Customer 5	SH	170
Customer 6	TH	100
Customer 7	SA	20
Customer 8	MVP	120
Customer 9	HH	100

- **Completely filled matrices usually only for higher dimension levels:**
- **Support for sparse matrices required (loss in performance)**
  - Decomposition of a cube into sub-cubes (“chunks”) that fit in main memory
  - Two-level addressing: chunk-Id, cell within a chunk

Query Language MDX[1]

- **MDX: MultiDimensional eXpressions**
  - Microsoft specification for cube access/queries in the context of OLEDB for OLAP
  - Based on SQL
  - Extraction of aggregated sub cubes/cuboids from the cube
- **Support by Microsoft and various tool vendors**
- **Main statement:**

```
SELECT   [<axis_specification> [, <axis_specification>...]]
FROM     [<cube_specification>]
[WHERE   [<slicer_specification>]]
```

  - axis_specification: considered dimension elements
  - 5 predefined axes: columns, rows, pages, chapters and sections
  - Slicer: Selection of the values which should be represented

[1] https://msdn.microsoft.com/en-us/library/ms145506.aspx

MDX: Example

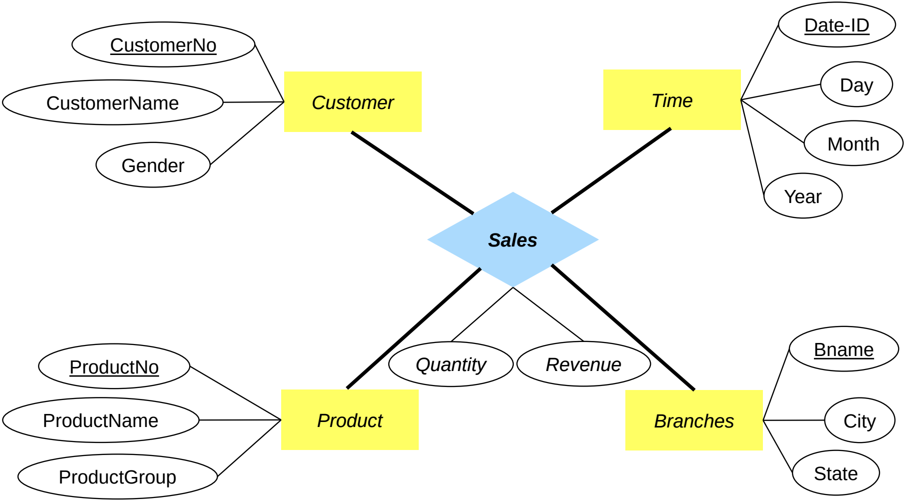
```
SELECT   Region.CHILDREN ON COLUMNS,
         Product.CHILDREN ON ROWS
FROM     Sales
WHERE    (Revenue, Time.[2007])
```

	Region		
Product	**East**	**West**	**…**
P 1	1200	1350	…
P 2	…	…	…
P 3	…	…	…
…	…	…	…

```
SELECT   Measures.MEMBERS ON COLUMNS,
         TOPCOUNT(Branch.City.MEMBERS, 10, Measures.Quantity) ON ROWS
FROM     Sales
```

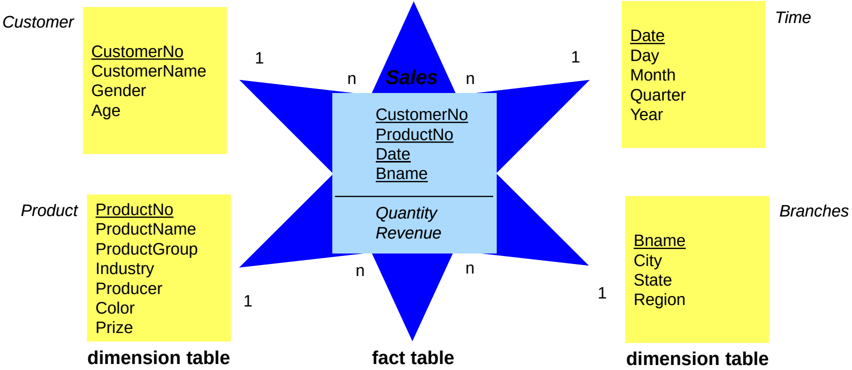
City	**Quantity**	**Revenue**	…
City 1	124	10000	…
City 2	35	3500	…
…	…	…	…
City 10	5	200	…

ER-Schema of a multi-dimensional Data model



Relational Storage: Star Schema

- Fact table corresponds to the center of the star Schema and contains the detailed data with the facts/operating numbers that should be analyzed
- One dimension table per dimension that is only connected with the fact table (⇒ starlike arrangement of the tables)



Sample Instance

Sales					
Date	BName	ProductNo	CustomerNo	Quantity	Revenue
7654	HH4	1847	4711	2	56000
...	...	...	...	...	...

Branches			
BName	City	State	Region
HH4	Hamburg	Hamburg	East
...	...	...	...

Customer			
CustomerNo	CustomerName	Gender	Age
4711	Weber	M	39
...	...	...	...

Time					
Date	Day	Month	Year	Quarter	...
7654	25	April	2005	2	...
...	...	...	...	...	

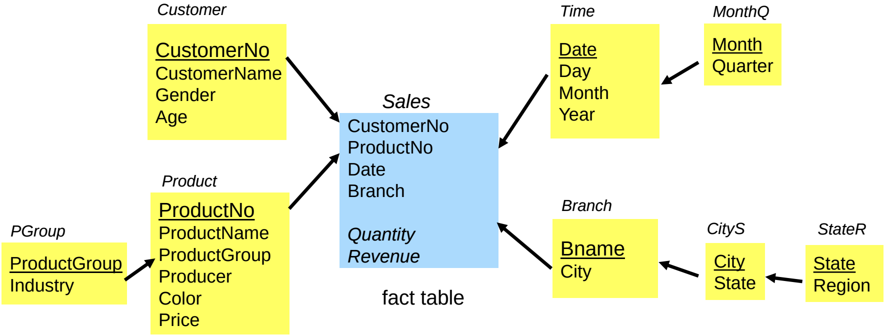
Product					
ProductNo	ProductName	ProductGroup	Producer	Color	Prize
1847	Passat XY	Car	VW	Blue	28000
...	...	...	...	...	...

Star Schema (2)

- **Formal definition: Star schema consists of a set of tables $D_1, \ldots, D_n, F$ with**
  - Dimension tables D_i consisting of a (usually surrogate) primary key d_i and dimension attributes
  - Fact table F consisting of the foreign keys $d_1, \ldots, d_n$ plus the facts as additional attributes
  - Dimension tables are usually denormalized, i.e. not in third normal form
- **Observations**
  - Number of data records in the fact table corresponds to the number of used (filled) cells in a multi-dimensional matrix
  - Because only relevant combinations are stored in the fact table, empty combinations of dimensions do not pose a problem
  - Nonetheless fact tables are often very large
  - Dimension tables most often are relatively small, but can be large as well (customers, products, etc.)

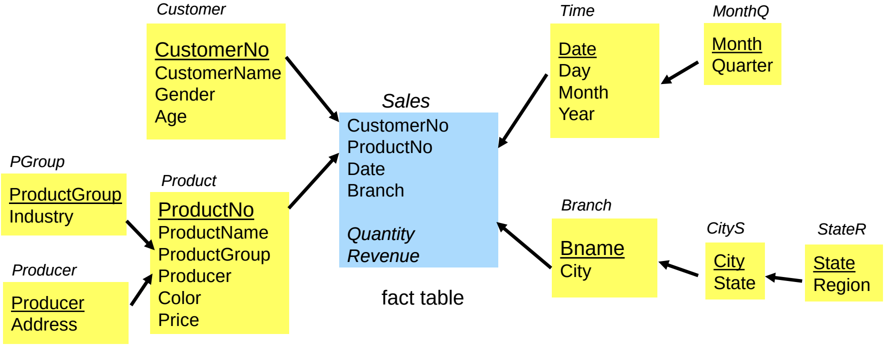
Snowflake Schema

- **Explicit representation of the dimension hierarchies**
- **Normalized dimension tables**
  - Less redundancy, less effort in the case of updates
  - Increased access costs (more Joins required)
- **Star schema is usually more suitable than snowflake schema**



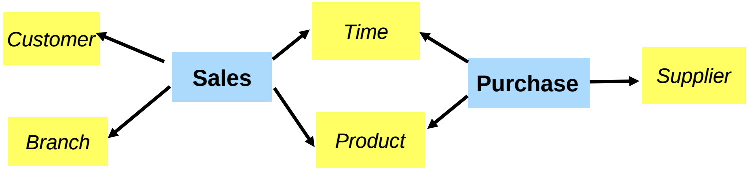
Snowflake Schema

- **Explicit representation of the dimension hierarchies**
- **Normalized dimension tables**
  - Less redundancy, less effort in the case of updates
  - Increased access costs (more Joins required)
- **Star schema is usually more suitable than snowflake schema**



Galaxy Schema

- **Data Warehouses usually have more than one fact table**
  $\Rightarrow$ Multi-star schema (galaxy schema, "Fact Constellation Schema")
- **Shared use of dimension tables**
- **Storage of precalculated aggregates**
  - Separate fact table per aggregate
  - Incorporation of aggregates into the fact table that consists the detailed data



HOLAP

- **Combination of MOLAP and ROLAP**
- **Vertical partitioning approach:**
  - Dense matrices most often only for higher dimension levels
  - Aggregates in MOLAP (fast query performance)
  - Detailed data in ROLAP (less storage requirements)
- **Horizontal partitioning approach:**
  - Most recent data in MOLAP and older data in ROLAP, or
  - Dense subregions of large cubes in MOLAP and sparse subregions of large cubes in ROLAP

Handling of Changes in the Dimensions

- **Types of changes**
  - New dimension elements
    (e.g. new product, product group or time period)
  - Change of values in one dimension element
    (e.g. new family status/residence of a customer)
  - New hierarchy levels in one dimension
  - New dimension
- **Handling on schema level (schema evolution) or tuple level**
- **Change of dimension elements**
  - **Solution 1:** Overwriting of old values (i.e. query results computed based on older time periods are now maybe incorrect)
  - **Solution 2:** Versioning of dimension elements on tuple level, e.g. extended key values
  - **Solution 3:** Versioning on schema level (i.e. new time attributes for validity period or alteration time)

Queries on the Star Schema

- **Star Join**
  - Starlike Join of the (relevant) dimension tables with the fact table
  - Restriction of the dimensions
  - Consolidation of the facts by grouping and aggregation

- **General form:**

Aggregated Facts

select g_1, ... g_k, agg(f_1), ... agg (f_m)
from D_1, ..., D_n, F
where <condition on D_1> **and**
... and
< condition on D_n> **and**
$D_1.d_1 = F.d_1$ and
... and
$D_n.d_n = F.d_n$
group by g_1, ... g_k
order by ...;

Relations of the Star Schema

Join Condition

Dimensionality of the Result

Example of a Star Join

- **In which year have female customers bought the most cars in Hamburg in the 1. Quarter?**

```
SELECT      z.Year as Year, SUM(s.Quantity) as TotalQuantity
FROM        Branches b, Product p, Time t, Customer c, Sales s
WHERE       t.Quarter = 1               AND  c.Gender = 'W'
   AND      p.ProductGroup = 'car'      AND  b.state = 'Hamburg'
   AND      s.Date = t.Date             AND  s.ProductNo = p.ProductNo
   AND      s.Branch = b.BName          AND  s.CustomerNo = c.CustomerNo
GROUP BY    t.Year
ORDER BY    TotalQuantity DESC;
```

Year	TotalQuantity
2004	745
2005	710
2003	650

Multi-dimensional Aggregation with Group-By

- **Number of attributes in group by-clause defines dimensionality**

```
SELECT p.Producer, t.Year, SUM(s.Quantity) as Quantity
FROM Sales s, Product p, Time t
WHERE s.ProductNo = p.ProductNo
AND s.Date= t.Date AND p.ProductGroup = 'car'
GROUP BY p.Producer, t.Year;                    2 dimensions
```

Producer	Year	*Quantity*
VW	2003	*2,000*
VW	2004	*3,000*
VW	2005	*3,500*
Opel	2003	*1,000*
...	...	...
BMW	2005	*1,500*
Ford	2003	*1,000*
Ford	2004	*1,500*
Ford	2005	*2,000*

```
SELECT p.Producer, SUM(s.Quantity) as Quantity
FROM Sales s, Product p
WHERE s.ProductNo = p.ProductNo
AND p.ProductGroup = 'car'
GROUP BY p.Producer;                            1 dimension
```

Producer	*Quantity*
VW	*8,500*
Opel	*3,500*
Ford	*4,500*
BMW	*3,000*

```
SELECT SUM(s.Quantity) as Quantity
FROM Sales s, Product p
WHERE s.ProductNo = p.ProductNo
AND p.ProductGroup = 'car';                     0 dimensions
```

Quantity
19,500

Relational Storage of Aggregated Values

- **Contingency table**

Year / Producer	2003	2004	2005	Σ
VW	2.000	3.000	3.500	8.500
Opel	1.000	1.000	1.500	3.500
BMW	500	1.000	1.500	3.000
Ford	1.000	1.500	2.000	4.500
Σ	4.500	6.500	8.500	19.500

- **Relational representation (2D cube)**

Producer	**Year**	***Quantity***
VW	2003	*2.000*
VW	2004	*3.000*
VW	2005	*3.500*
Opel	2003	*1.000*
Opel	2004	*1.000*
Opel	2005	*1.500*
BMW	2003	*500*
BMW	2004	*1.000*
BMW	2005	*1.500*
Ford	2003	*1.000*
Ford	2004	*1.500*
Ford	2005	*2.000*
VW	**ALL**	*8.500*
Opel	**ALL**	*3.500*
BMW	**ALL**	*3.000*
Ford	**ALL**	*4.500*
ALL	2003	*4.500*
ALL	2004	*6.500*
ALL	2005	*8.500*
ALL	**ALL**	*19.500*

Materialization of Aggregation Results

```
CREATE TABLE Car2DCube (Producer varchar (20), Year varchar (4), Quantity integer);

INSERT INTO Car2DCube
  (SELECT p.Producer, t.Year, SUM(s.Quantity)                          2 dimensions
  FROM Sales s, Product p, Time t
  WHERE s.ProductNo = p.ProductNo AND p.ProductGroup = 'car' AND s.Date = t.Date
  GROUP BY t.Year, p.Producer)
UNION
  (SELECT p.Producer, ALL, SUM(s.Quantity)                             1 dimension
  FROM Sales s, Product p
  WHERE s.ProductNo = p.ProductNo AND p.ProductGroup = 'car'
  GROUP BY p.Producer)
UNION
  (SELECT ALL, t.Year, SUM(s.Quantity)                                 1 dimension
  FROM Sales s, Product p, Time t
  WHERE s.ProductNo = p.ProductNo AND p.ProductGroup = 'car' AND s.Date = t.Date
  GROUP BY t.Year)
UNION
  (SELECT ALL, ALL, SUM(s.Quantity)                                    0 dimensions
  FROM Sales s, Product p
  WHERE s.ProductNo = p.ProductNo AND p.ProductGroup = 'car');
```

CUBE Operator

- **SQL extension for n-dimensional grouping and aggregation**
  - Syntax: GROUP BY CUBE (D1, D2, ..., Dn)
  - Generates a table with aggregated values (ALL-Tuple) as a result
  - Implemented in MS SQL-Server, DB2, Oracle
- **Avoids a redundant computation of the same aggregation**
  - Avoids 2^n union-queries (for n attributes in the group by clause / n dimensions)
  - Simple formulation of queries
  - Efficient computation by DBS (reuse of interim results)
- **Example:**

```
SELECT p.Producer, t.Year, c.Gender, SUM(s.Quantity)
FROM Sales s, Product p, Time t, Customer c
WHERE s.ProductNo = p.ProductNo AND s.Date = t.Date
  AND s.Customer = c.CustomerNo AND p.ProductGroup = 'car'
GROUP BY CUBE (p.Producer, t.Year, c.Gender);
```

Without cube operator:
8 union-queries!

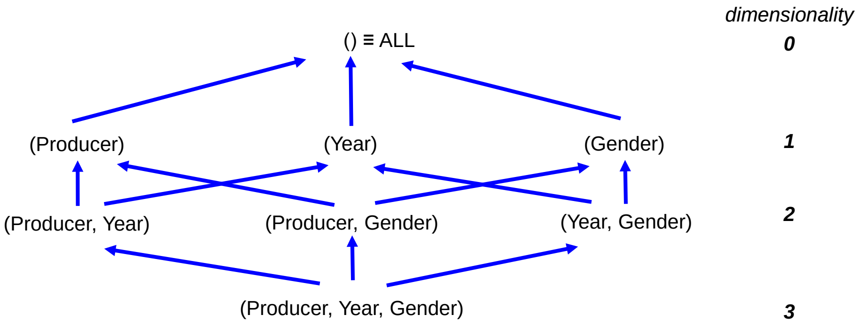
3D Cube in Relational Form

Producer	Year	Gender	Quantity
VW	2003	m	1300
VW	2003	w	700
VW	2004	m	1900
VW	2004	w	1100
VW	2005	m	2300
...	...	...	...
Opel	2003	m	800
Opel	2003	w	200
...	...	...	...
BMW	...	...	...
...	...	...	...

CUBE

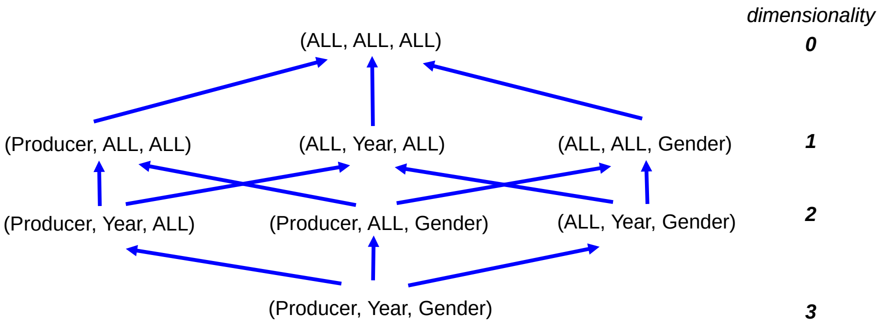
Producer	Year	Gender	Quantity
VW	2003	m	1300
VW	2003	w	700
...	...	...	...
VW	2003	ALL	2.000
...	...	ALL	...
Ford	2005	ALL	2.000
VW	ALL	m	5.400
...	...	...	...
Ford	ALL	w	...
ALL	2001	m	...
...	...		
VW	ALL	ALL	8.500
...	...		
ALL	2001	ALL	...
...			
ALL	ALL	m	...
...			
ALL	ALL	ALL	19.500

Cube Aggregation Grid



- Low-level aggregates/cuboids can be derived from high-level ones
- Materialization/Caching of frequently used aggregates enables query optimization

Cube Aggregation Grid



- Low-level aggregates/cuboids can be derived from high-level ones
- Materialization/Caching of frequently used aggregates enables query optimization

ROLLUP Operator

- **CUBE Operator: inter-dimensional grouping/aggregation**
  - Generates aggregates for all 2^n possible combinations of n dimensions
  - Too expensive for Roll-Up/Drill-Down within a single dimension
- **ROLLUP Operator: intra-dimensional aggregation**
- **ROLLUP for $a_1, a_2, \ldots, a_n, f()$**
  produces only the cuboids

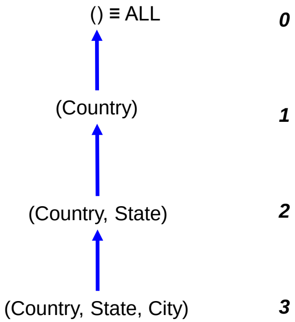
$$
\begin{aligned}
&a_1, a_2, \ldots, a_{n-1}, a_n, f(),\\
&a_1, a_2, \ldots, a_{n-1}, \text{ALL}, f(),\\
&\ldots\\
&a_1, \text{ALL}, \ldots, \text{ALL}, f(),\\
&\text{ALL}, \text{ALL}, \ldots, \text{ALL}, f()
\end{aligned}
$$

$\boxed{n+1 \text{ cuboids}}$

- **Order of the attributes is relevant!**

ROLLUP Operator: Example 1

- SELECT c.Country, c.State, c.City, SUM(s.Quantity)
  FROM Sales s, Customer c
  WHERE s.Customer = c.CustomerNo AND c.Age BETWEEN 20 AND 30
  GROUP BY ROLLUP (c.Country, c.State, c.City);



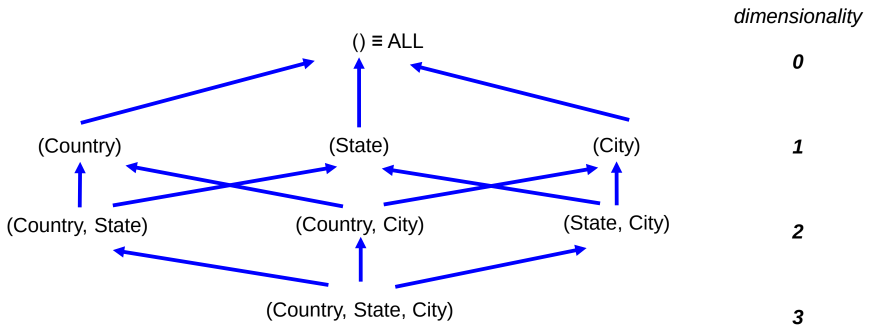
Functional Dependencies:

- City $\rightarrow$ State, Country
- State $\rightarrow$ Country

$\Rightarrow$ Group By (State, Country, City)
$\equiv$ Group By (State, City)
$\equiv$ Group By (Country, City)
$\equiv$ Group By (City)

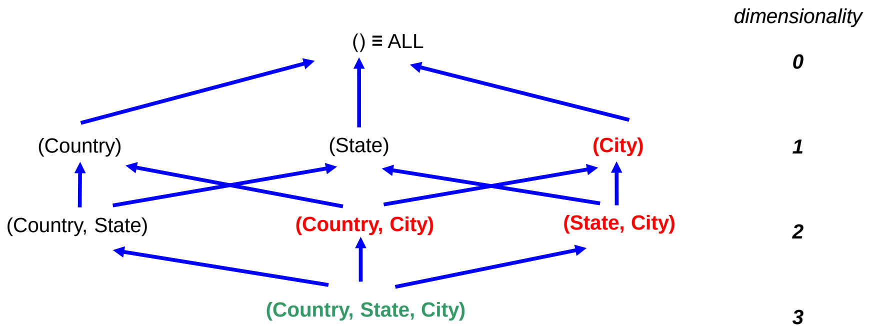
$\Rightarrow$ Group By (State, Country)
$\equiv$ Group By (Country)

ROLLUP Operator: Example 1 (Aggregation Grid)



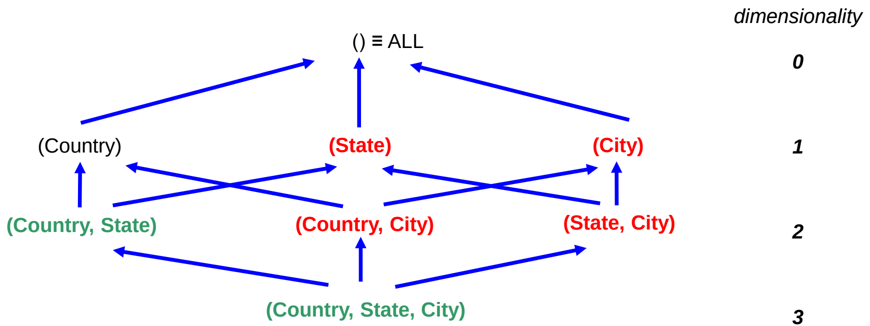
- Because of the functional dependencies there are only four different groupings
- RollUp operator does not restrict considered set of groupings
  (⇒ only avoids redundant computation of same results)

ROLLUP Operator: Example 1 (Aggregation Grid)



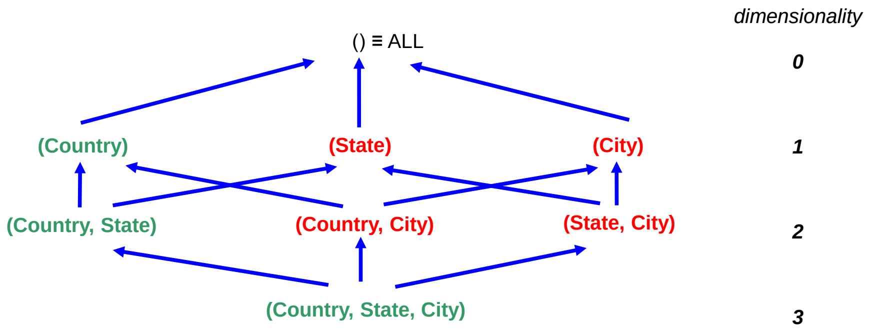
- Because of the functional dependencies there are only four different groupings
- RollUp operator does not restrict considered set of groupings ($\Rightarrow$ only avoids redundant computation of same results)

ROLLUP Operator: Example 1 (Aggregation Grid)



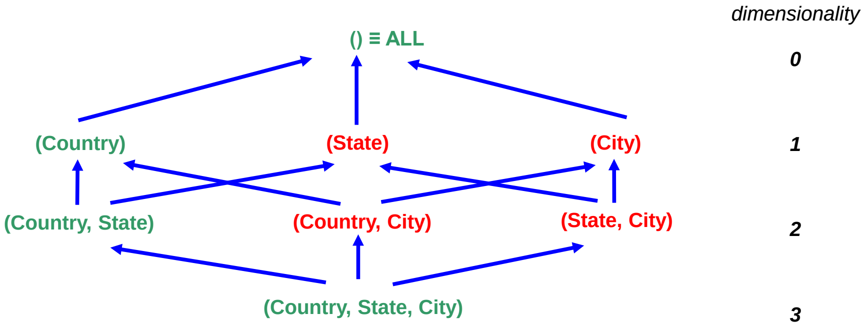
- Because of the functional dependencies there are only four different groupings
- RollUp operator does not restrict considered set of groupings ($\Rightarrow$ only avoids redundant computation of same results)

ROLLUP Operator: Example 1 (Aggregation Grid)



- Because of the functional dependencies there are only four different groupings
- RollUp operator does not restrict considered set of groupings ($\Rightarrow$ only avoids redundant computation of same results)

ROLLUP Operator: Example 1 (Aggregation Grid)



- Because of the functional dependencies there are only four different groupings
- RollUp operator does not restrict considered set of groupings ($\Rightarrow$ only avoids redundant computation of same results)

ROLLUP Operator: Example 1

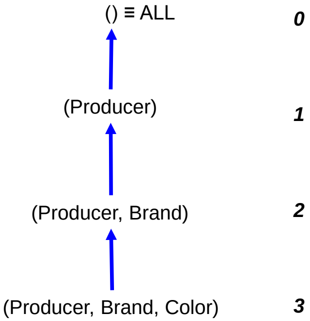
Country	State	City	Quantity
Germany	Hessen	Kassel	800
Germany	Hessen	Gießen	600
Germany	Hessen	Frankfurt	600
Germany	Bayern	München	1.200
Germany	Bayern	Hof	800
Germany	Bayern	Bamberg	1.000
Germany	...	...	1.400
...	...	...	...
USA	Texas	Austin	400
USA	Texas	Houston	300
USA	Texas	Dallas	300
...	...		

→ **ROLLUP**

Country	State	City	Quantity
Germany	Hessen	Kassel	800
Germany	Hessen	Gießen	600
Germany	Hessen	Frankfurt	600
Germany	Bayern	München	1.200
Germany	Bayern	Hof	800
Germany	Bayern	Bamberg	1.000
Germany	...	...	1.400
...	...	...	...
USA	Texas	Austin	400
USA	Texas	Houston	300
USA	Texas	Dallas	300
...	...	...	...
Germany	Hessen	ALL	2.000
Germany	Bayern	ALL	3.000
Germany	...	ALL	3.500
USA	Texas	ALL	1.600
USA	...	ALL	...
Germany	ALL	ALL	8.500
USA	ALL	ALL	3.500
ALL	ALL	ALL	12.000

ROLLUP Operator: Example 2

- SELECT p.Producer, p.Brand, p.Color, SUM(s.Quantity)
  FROM Sales s, Product p
  WHERE s.ProductNo = p.ProductNo AND p.Producer IN ('VW','Opel')
  GROUP BY ROLLUP (p.Producer, p.Brand, p.Color);



- Attributes are not part of a dimension hierarchy

$\Rightarrow$ no functional dependencies

$\Rightarrow$ in theory every combination would lead to another grouping

$\Rightarrow$ RollUp operator actually restricts considered set of groupings

ROLLUP Operator: Example 2

Producer	Brand	Color	Quantity
VW	Passat	rot	800
VW	Passat	weiß	600
VW	Passat	blau	600
VW	Golf	rot	1.200
VW	Golf	weiß	800
VW	Golf	blau	1.000
VW	...	rot	1.400
...	...	...	...
Opel	Vectra	rot	400
Opel	Vectra	weiß	300
Opel	Vectra	blau	300
...	...		

→ **ROLLUP**

Producer	Brand	Color	Quantity
VW	Passat	rot	800
VW	Passat	weiß	600
VW	Passat	blau	600
VW	Golf	rot	1.200
VW	Golf	weiß	800
VW	Golf	blau	1.000
VW	...	rot	1.400
...	...	...	...
Opel	Vectra	rot	400
Opel	Vectra	weiß	300
Opel	Vectra	blau	300
...	...		
VW	Passat	ALL	2.000
VW	Golf	ALL	3.000
VW	...	ALL	3.500
Opel	Vectra	ALL	1.600
Opel	...	ALL	...
VW	ALL	ALL	8.500
Opel	ALL	ALL	3.500
ALL	ALL	ALL	12.000

GROUPING SETS Operator

- **Multiple groupings per query:**

  GROUP BY GROUPING SETS (< Groupspecification list>)
  Groupspecification: (< Groupspecification list >) |
  CUBE < Groupspecification list > |
  ROLLUP < Groupspecification list >
  Empty specification list ( ) possible: Aggregation on whole table

- **Example:**

```
SELECT p.Producer, p.Color, SUM(s.Quantity)
FROM Sales s, Product p
WHERE s.ProductNo = p. ProductNo
    AND p.Producer IN ('VW','Opel')
GROUP BY GROUPING SETS
                ((p.Producer), (p.Color),());
```

Producer	**Color**	***Quantity***
VW	**ALL**	*8500*
Opel	**ALL**	*3500*
ALL	blau	*3100*
ALL	rot	*6200*
ALL	weiß	*2700*
ALL	**ALL**	*12000*

- **Cube, RollUp and Group-By correspond to specific Grouping Sets**

Grouping Sets Equivalents

- Group By Cube (A,B) $\equiv$ Group By Grouping Sets ((A,B),(A),(B),())
- Group By RollUp (A,B) $\equiv$ Group By Grouping Sets ((A,B),(A),())
- Group By A,B $\equiv$ Group By Grouping Sets ((A,B))
- Group By A, Grouping Sets ((B),(C),())
  $\equiv$
  Group By Grouping Sets ((A,B),(A,C),(A))
- Group By Grouping Sets ((A,B),(B,C)), Grouping Sets ((D,E),(D),())
  $\equiv$
  Group By Grouping Sets ((A,B,D,E),(A,B,D),(A,B),(B,C,D,E), (B,C,D),(B,C))

Single Steps in Designing a Multi-dimensional Schema

- **Which business processes should be modeled and analyzed?**
- **Definition of the facts**
  - Where do they come from?
  - Granularity of the facts. Which OLAP-precision is necessary?
- **Determination of the dimensions**
  - Shared characteristics of the facts
  - Specification of the dimension attributes
  - Constant vs. varying dimension attributes
  - Establishment/usage of a uniform terminology
- **Physical design decisions**
  - Architecture (ROLAP, MOLAP and HOLAP)
  - Precalculation of aggregations
  - Identifying memory requirements
- **Definition of the length of history, handling of old data**
- **Refresh rate with respect to the source systems**

Summary

- **Simplicity of the multi-dimensional modeling approach essential for success of Data Warehousing**
  - Cube-based representation with facts and hierarchical dimensions
  - Operations: Slice and Dice, Roll-Up, Drill-Down, ...
- **Multi-dimensional storage**
  - Problem of sparse matrices
  - Primary relevant for aggregated data, less relevant for managing the detailed facts
- **Relational storage on the basis of the star schema**
  - Support of large data sets, scalability
  - New requirements with respect to an efficient evaluation of Star Joins, multi-dimensional grouping and aggregation ...
- **Precalculation of aggregated data can be essential for an adequate performance**
- **SQL-extensions:** CUBE, ROLLUP and GROUPING SETS operators

Exercise (1)

Compute the results of the following SQL-queries:

- SELECT Player, Saison,
  SUM(Quantity) as Goals
  FROM Goals
  GROUP BY ROLLUP (Player, Saison);
- SELECT Player, Saison,
  SUM(Quantity) as Goals
  FROM Goals
  GROUP BY CUBE (Player, Saison);
- SELECT Player, Saison,
  SUM(Quantity) as Goals
  FROM Goals
  GROUP BY GROUPING SETS ((Player), (Saison),());

Player	**Saison**	***Quantity***
Elber	1999	*13*
Elber	2000	*14*
Elber	2001	*15*
Scholl	1997	*5*
Scholl	1998	*9*
Scholl	1999	*4*
Scholl	2000	*6*
Scholl	2001	*9*

Exercise (2)

- SELECT Player, Saison, SUM(Quantity) as Goals
  FROM Goals
  GROUP BY ROLLUP (Player, Saison);

Player	**Saison**	***Quantity***
Elber	1999	*13*
Elber	2000	*14*
Elber	2001	*15*
Scholl	1997	*5*
Scholl	1998	*9*
Scholl	1999	*4*
Scholl	2000	*6*
Scholl	2001	*9*

→

Player	**Saison**	***Goals***
Elber	1999	*13*
Elber	2000	*14*
Elber	2001	*15*
Scholl	1997	*5*
Scholl	1998	*9*
Scholl	1999	*4*
Scholl	2000	*6*
Scholl	2001	*9*
Elber	ALL	*32*
Scholl	ALL	*33*
ALL	ALL	*65*

Exercise (3)

- SELECT Player, Saison, SUM(Quantity) as Goals
  FROM Goals
  GROUP BY CUBE (Player, Saison);

Player	**Saison**	***Quantity***
Elber	1999	*13*
Elber	2000	*14*
Elber	2001	*15*
Scholl	1997	*5*
Scholl	1998	*9*
Scholl	1999	*4*
Scholl	2000	*6*
Scholl	2001	*9*

→

Player	**Saison**	***Goals***
Elber	1999	*13*
Elber	2000	*14*
Elber	2001	*15*
Scholl	1997	*5*
Scholl	1998	*9*
Scholl	1999	*4*
Scholl	2000	*6*
Scholl	2001	*9*
ALL	1997	*5*
ALL	1998	*9*
ALL	1999	*17*
ALL	2000	*20*
ALL	2001	*24*
Elber	ALL	*32*
Scholl	ALL	*33*
ALL	ALL	*65*

Exercise (4)

- SELECT Player, Saison, SUM(Quantity) as Goals
  FROM Goals
  GROUP BY GROUPING SETS ((Player), (Saison),());

Player	**Saison**	***Quantity***
Elber	1999	*13*
Elber	2000	*14*
Elber	2001	*15*
Scholl	1997	*5*
Scholl	1998	*9*
Scholl	1999	*4*
Scholl	2000	*6*
Scholl	2001	*9*

$\rightarrow$

Player	**Saison**	***Goals***
Elber	ALL	*32*
Scholl	ALL	*33*
ALL	1997	*5*
ALL	1998	*9*
ALL	1999	*17*
ALL	2000	*20*
ALL	2001	*24*
ALL	ALL	*65*