
Lecture 06

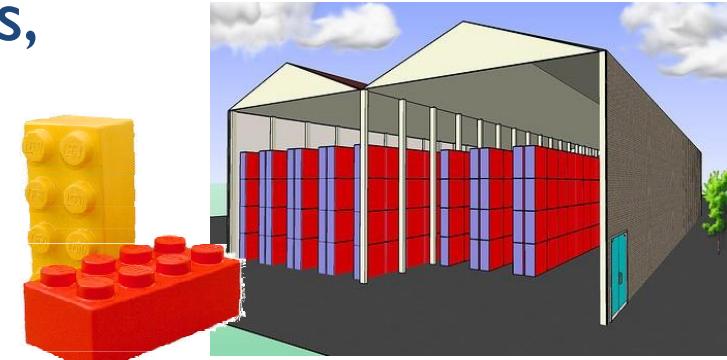
Data Warehouse Architecture (cont'd.)

Physical Model

Summary – last week

Summary

- Logical Model
 - Cubes, Dimensions, Hierarchies, Classification Levels
- This week:
- Physical Model
 - Relational Implementation through:
 - Star schema: improves query performance for often-used data
 - Snowflake schema: reduce the size of the dimension tables
 - Multidimensional implementation



Physical Model

- Defining the **physical structures**
 - Setting up the database environment
 - Performance tuning strategies (next lecture)
 - Indexing
 - Partitioning
 - Materialization
- Goal:
 - Define the actual storage architecture
 - Decide on how the data is to be accessed and how it is arranged

Physical Model (cont'd.)

- Physical implementation of the multidimensional paradigm model can be:
 - Relational
 - Snowflake-schema
 - Star-schema
 - Multidimensional
 - Matrixes



Relational Model

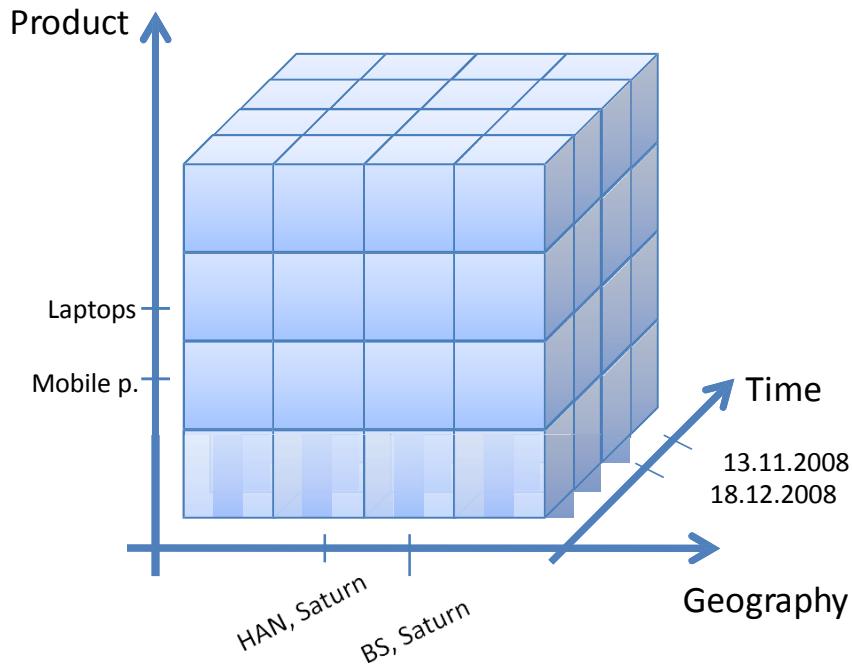
- **Relational** model, goals:
 - As low loss of semantically knowledge as possible e.g., classification hierarchies
 - The translation from multidimensional queries must be efficient – as they will be in SQL
 - The RDBMS should be able to run the translated queries efficiently
 - The maintenance of the present tables should be easy and fast e.g., when loading new data



Relational Model (cont'd.)

- Going from **multidimensional** to **relational**
 - **Representations** for cubes, dimensions, classification hierarchies and attributes
 - **Implementation** of cubes without the classification hierarchies is easy
 - A table can be seen as a **cube**
 - A column of a table can be considered as a dimension mapping
 - A tuple in the table represents a cell in the cube
 - If we interpret only a part of the columns as dimensions we can use the rest as measures
 - The resulting table is called a **fact table**

Relational Model (cont'd.)



Article	Store	Day	Sales
Laptops	Hannover, Saturn	13.11.2008	6
Mobile Phones	Hannover Saturn	18.12.2008	24
Laptops	Braunschweig Saturn	18.12.2008	3

Snowflake Schema

- Snowflake-schema
 - Simple idea: use a table for each classification level
 - This table includes the ID of the classification level and other attributes
 - 2 neighbor classification levels are connected by 1:n connections e.g., from n Days to 1 Month
 - The measures of a cube are maintained in a fact table
 - Besides measures, there are also the foreign key IDs for the smallest classification levels

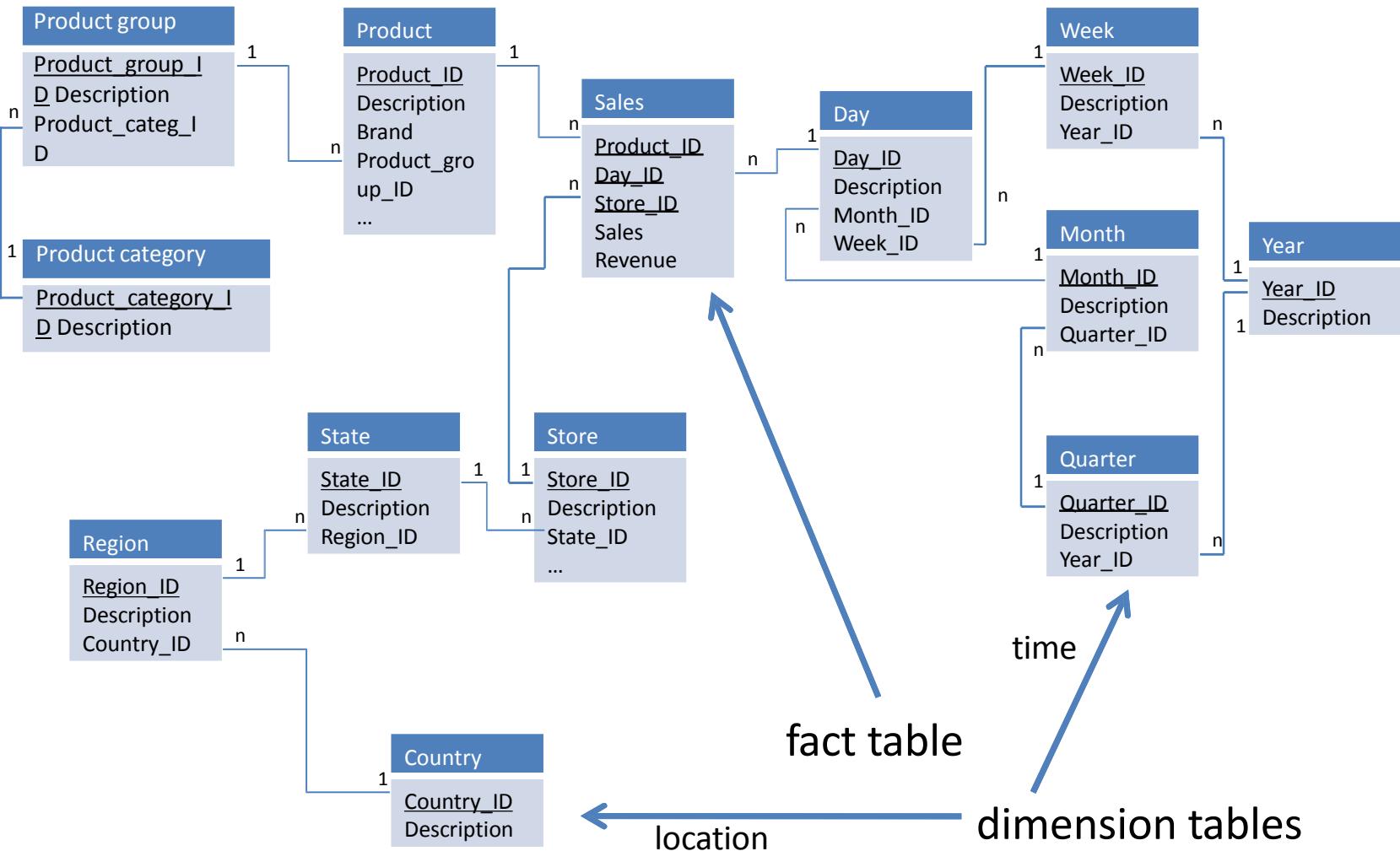


Snowflake Schema (cont'd.)

- Snowflake?
 - The facts/measures are in the center
 - The dimensions spread out in each direction and branch out with their granularity



Snowflake Schema (cont'd.)



Snowflake Schema (cont'd.)

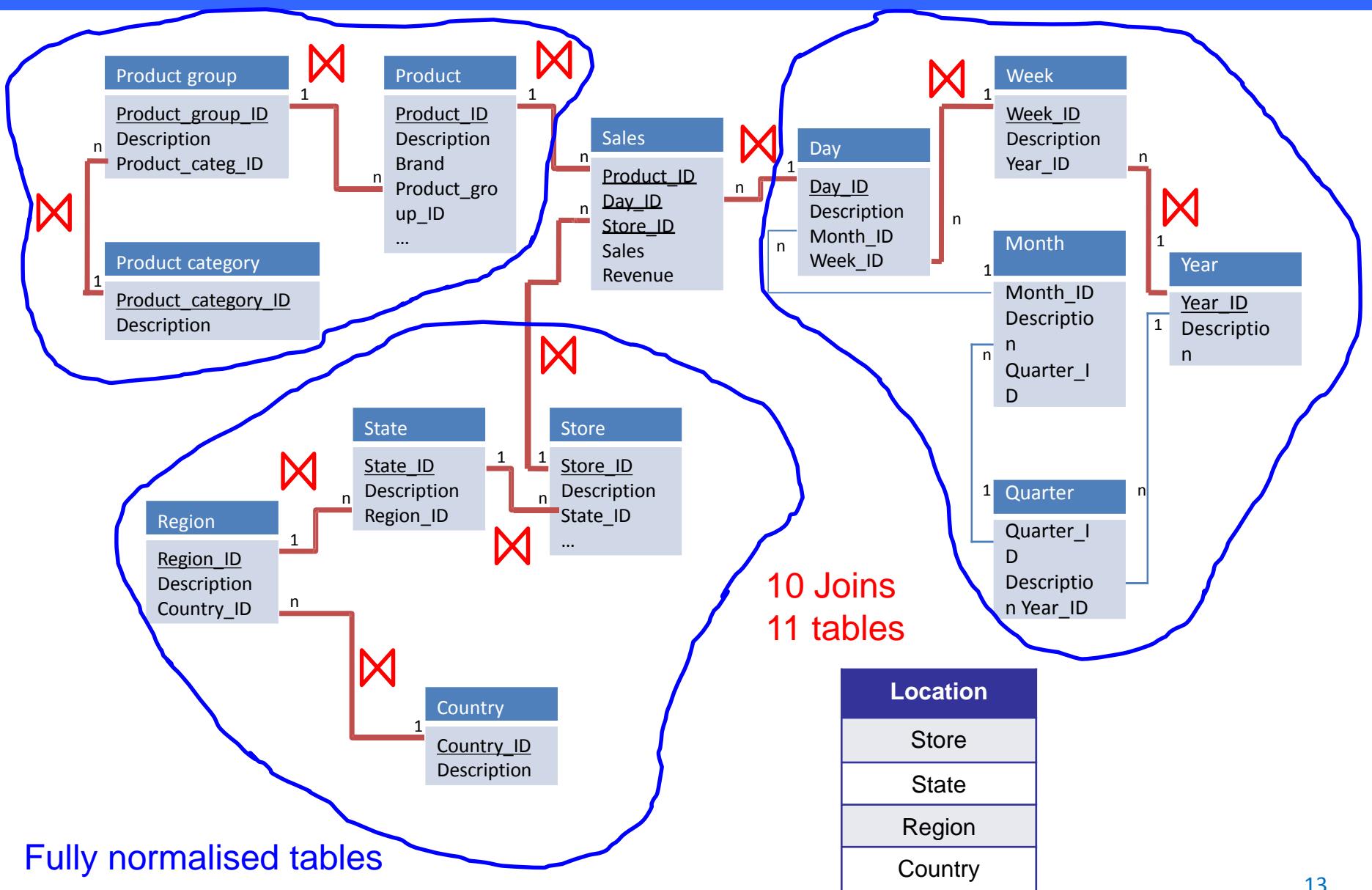
- Snowflake schema – **A**dvantages
 - With a snowflake schema the **size of the dimension tables** will be reduced and queries will run faster
 - If a dimension is **very sparse** (most measures corresponding to the dimension have no data)
 - And/or a dimension **has long list of attributes** which may be queried



Snowflake Schema (cont'd.)

- Snowflake schema – **Disadvantages**
 - Fact tables are responsible for 90% of the storage requirements
 - Thus, normalizing the dimensions usually lead to insignificant improvements
 - Normalization of the dimension tables can reduce the performance of the DW because it leads to a large number of tables
 - E.g., when connecting dimensions with coarse granularity these tables are joined with each other during queries
 - A query which connects Product category with Year and Country is clearly not performant (10 tables need to be connected)

Snowflake Schema (cont'd.)

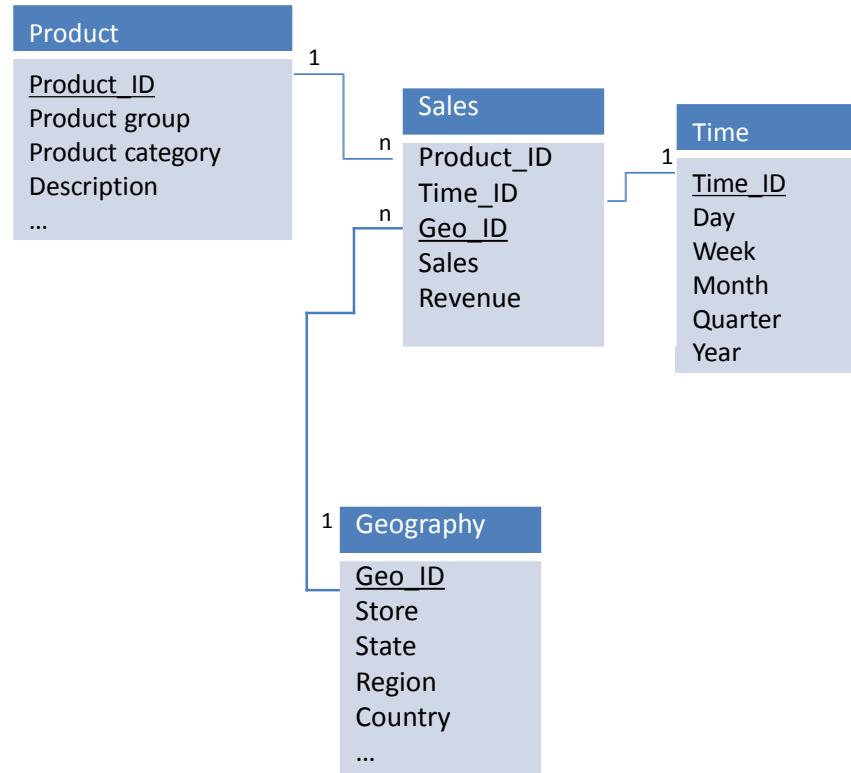


Star Schema

- Star schema
 - Basic idea: use a denormalized schema for all the dimensions
 - A star schema can be obtained from the snowflake schema through the denormalization of the tables belonging to a dimension



Star Schema (cont'd.)



Star Schema (cont'd.)

- Advantages
 - Improves query performance for often-used data
 - Less tables and simple structure
 - Efficient query processing with regard to dimensions
- Disadvantages
 - In some cases, high overhead of redundant data



Snowflake vs. Star Schema

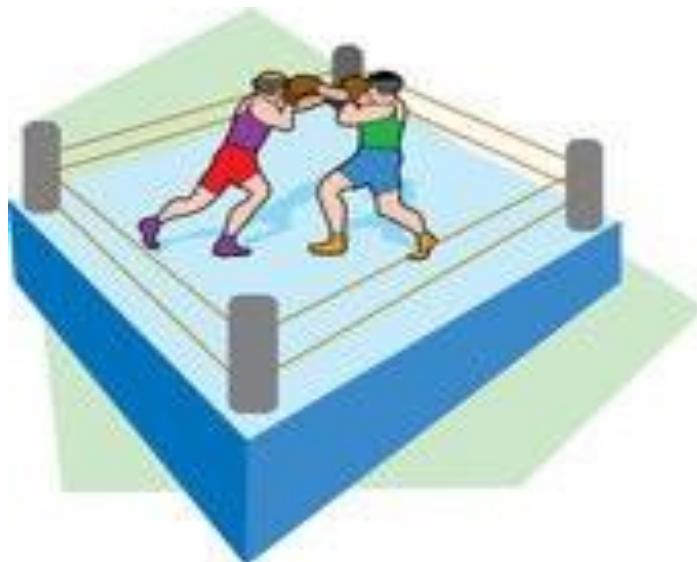
Snowflake

- The structure of the classifications are expressed in table schemas
- The fact table and dimension tables are normalized

vs.

Star

- The entire classification is expressed in just one table
- The fact table is normalized while in the dimension table the normalization is broken
 - This leads to redundancy of information in the dimension tables



Examples

- Snowflake

Product_ID	Description	Brand	Prod_group_ID
10	E71	Nokia	4
11	PS-42A	Samsung	2
12	5800	Nokia	4
	Bold	Berry	4

Prod_group_ID	Description	Prod_categ_ID
2	TV	11
4	Mobile Ph..	11

Prod_categ_ID	Description
11	Electronics

- Star

Product_ID	Description	...	Prod_group	Prod_categ
10	E71	...	Mobile Ph..	Electronics
11	PS-42A	...	TV	Electronics
12	5800	...	Mobile Ph..	Electronics
13	Bold	...	Mobile Ph..	Electronics

Snowflake to Star Schema

- When should we go from Snowflake to star?
 - Heuristics-based decision
 - When typical queries relate to coarser granularity (like product category)
 - When the volume of data in the dimension tables is relatively low compared to the fact table
 - In this case a star schema leads to negligible overhead through redundancy, but performance is improved
 - When modifications on the classifications are rare compared to insertion of fact data
 - In this case these modifications controlled through the data load process of the ETL reducing the risk of data anomalies

Do we have a winner?

- Snowflake or Star?
 - It depends on the necessity
 - Fast query processing or efficient space usage
 - However, most of the time a mixed form is used
 - The Starflake schema: some dimensions stay normalized corresponding to the snowflake schema, while others are denormalized according to the star schema



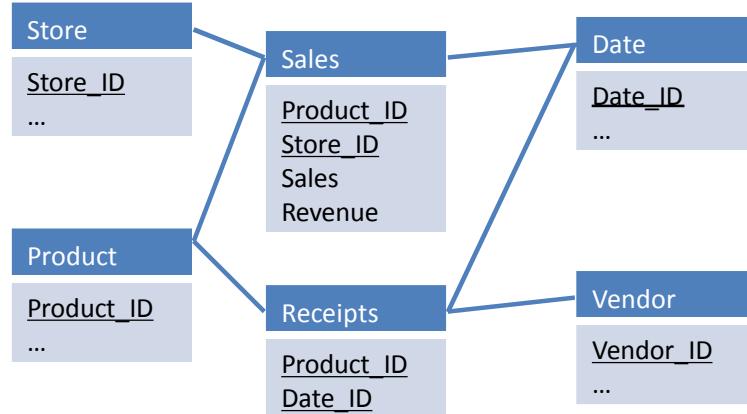
Our Focuses Combined

- The **Starflake schema**
 - The decision on how to deal with the dimensions is influenced by
 - **Frequency** of the modifications: if the dimensions change often, normalization leads to better results
 - **Amount** of dimension elements: the bigger the dimension tables, the more space normalization saves
 - **Number of classification levels** in a dimension: more classification levels introduce more redundancy in the star schema
 - **Materialization of aggregates** for the dimension levels: if the aggregates are materialized, a normalization of the dimension can bring better response time



More Schemas

- Galaxies
 - In practice we usually have more measures described by different dimensions
 - Thus, more fact tables



Even More Schemas

- Other schemas
 - Fact constellations
 - Pre-calculated aggregates
 - Factless fact tables
 - Fact tables do not have non-key data
 - Can be used for event tracking or to inventory the set of possible occurrences
 - ...



Relational Model

- Relational model - **disadvantages**
 - The representation of the multidimensional data can be implemented relationally with a finite set of transformation steps, however:
 - Multidimensional queries have to be first **translated** to the relational representation
 - A **direct interaction** with the relational data model is not fit for the end user
- What about storing the data multidimensionally?



Multidimensional Model

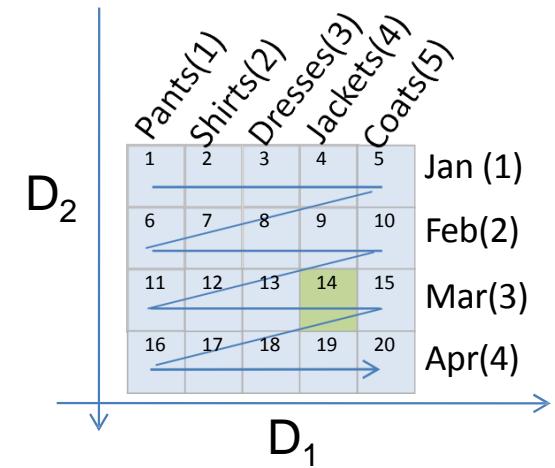
- The basic data structure for multidimensional data storage is the **array**
 - The elementary data structures are the cubes and the dimensions
- $$C=((D_1, \dots, D_n), (M_1, \dots, M_m))$$
- The storage is intuitive as arrays of arrays, physically **linearized**

Multidimensional Model (cont'd.)

- Linearization example: 2D cube $|D_1| = 5$, $|D_2| = 4$, cube cells = 20
 - Query: Jackets sold in March?
 - Measure stored in cube cell $D_1[4], D_2[3]$
 - The 2D cube is physically stored as a linear array, so

$D_1[4], D_2[3]$ becomes array cell 14

- $(\text{Index}(D_2) - 1) * |D_1| + \text{Index}(D_1)$
- Linearized Index = $2 * 5 + 4 = 14$



- Generalization

Physically

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----	----	----	----	----

Linearization

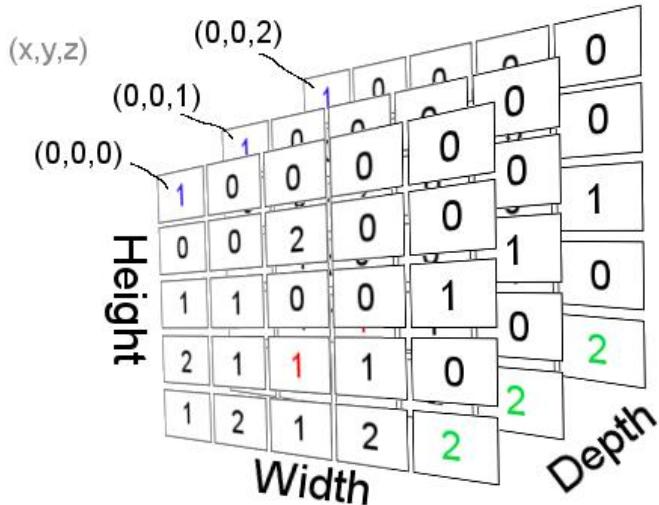
- Generalization:

- Given a cube $C = ((D_1, D_2, \dots, D_n), (M_1:\text{Type}_1, M_2:\text{Type}_2, \dots, M_m:\text{Type}_m))$,

- the index of a cube cell z with coordinates (x_1, x_2, \dots, x_n) can be linearized as follows:

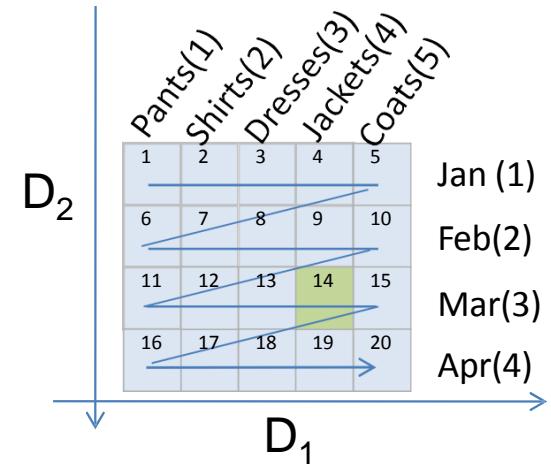
- $\bullet \text{Index}(z) = x_1 + (x_2 - 1) * |D_1| + (x_3 - 1) * |D_1| * |D_2| + \dots + (x_n - 1) * |D_1| * \dots * |D_{n-1}| =$

$$= 1 + \sum_{i=1}^n ((x_i - 1) * \prod_{j=1}^{i-1} |D_j|)$$



Problems in Array-Storage

- Influence of the **order of the dimensions** in the cube definition
 - In the cube the cells of D_2 are ordered one under the other
e.g., sales of all pants involves a **column** in the cube
 - After linearization, the information is **spread** among more data blocks or pages
 - If we consider a data block can hold 5 cells, a query over **all products sold in January** can be answered with just 1 block read, but a query of **all sold pants**, involves reading 4 blocks



Problems in Array-Storage (cont'd.)

- Solution: use **caching techniques**
 - But... caching and swapping is performed also by the operating system
 - MDBMS has to manage its caches such that the OS doesn't perform any damaging swaps



Problems in Array-Storage (cont'd.)

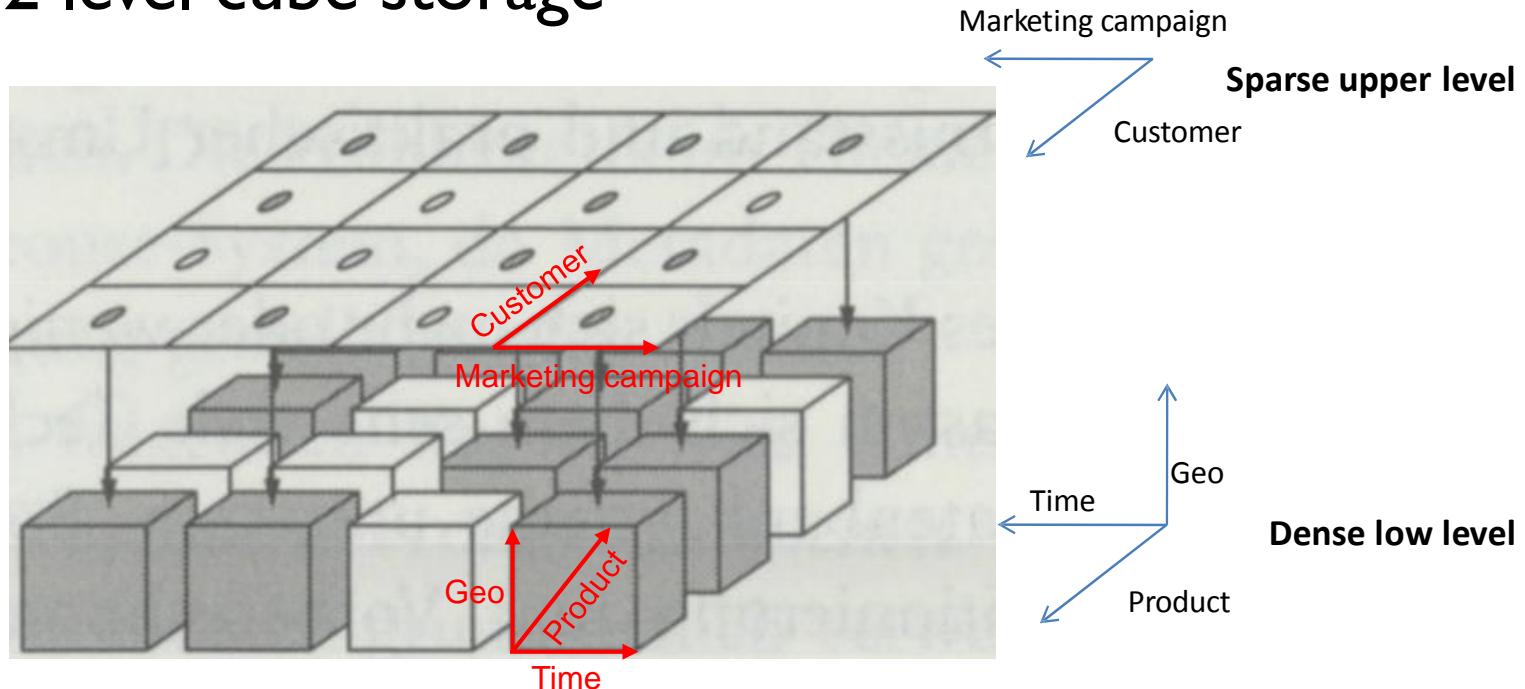
- Storage of **dense cubes**
 - If cubes are **dense**, array storage is **more efficient**. However, operations suffer due to the large cubes
 - Solution: store dense cubes not linear but on **2 levels**
 - The first contains **indexes** and the second **the data cells** stored in blocks
 - Optimization procedures like **indexes** (trees, bitmaps), physical partitioning, and **compression** (run-length-encoding) can be used

Problems in Array-Storage (cont'd.)

- Storage of **sparse cubes**
 - All the cells of a cube, including empty ones, have to be stored
 - Sparseness leads to data being stored in many physical blocks or pages
 - The query speed is affected by the large number of block accesses on the secondary memory
 - Solution:
 - Do not store empty blocks or pages but adapt the index structure
 - 2 level data structure: upper layer holds all possible combinations of the sparse dimensions, lower layer holds dense dimensions

Problems in Array-Storage (cont'd.)

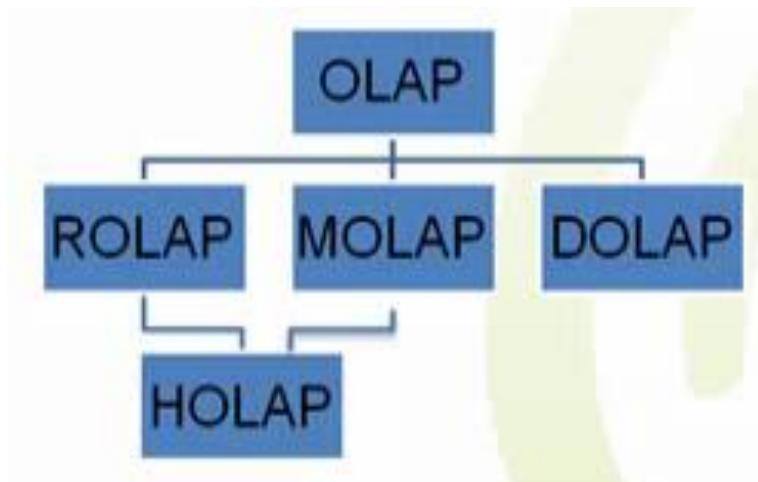
- 2 level cube storage



Physical Models

Detour

- Based on the physical model used:
 - DOLAP (Desktop OLAP)
 - MOLAP (Multidimensional OLAP)
 - ROLAP (Relational OLAP)
 - HOLAP (Hybrid OLAP)



- **DOLAP**

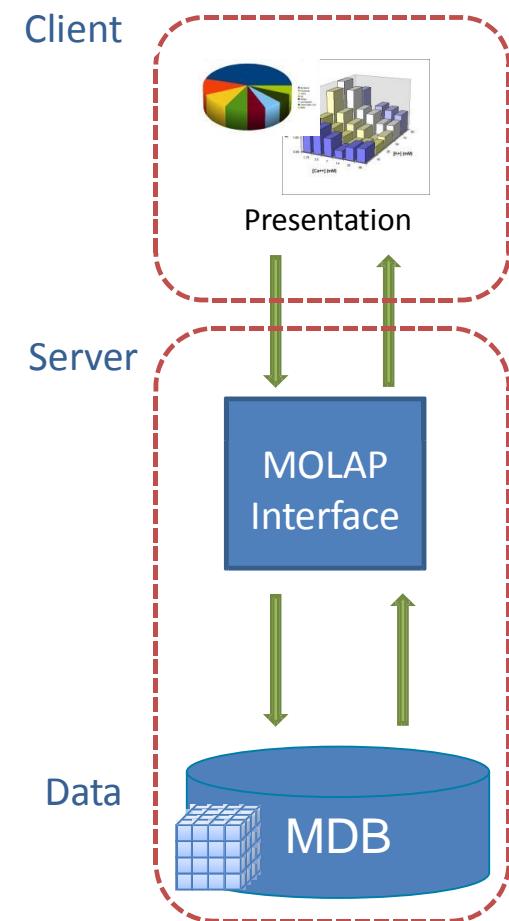
- Developed as extension to the production system reports
 - It downloads a small hypercube from a central point (data mart or DW)
 - Performs multidimensional analysis while disconnected from the data source
 - The computation occurs on the client
- Requires little investment
- They lack the ability to manage large data sets

MOLAP

Detour

- **MOLAP**

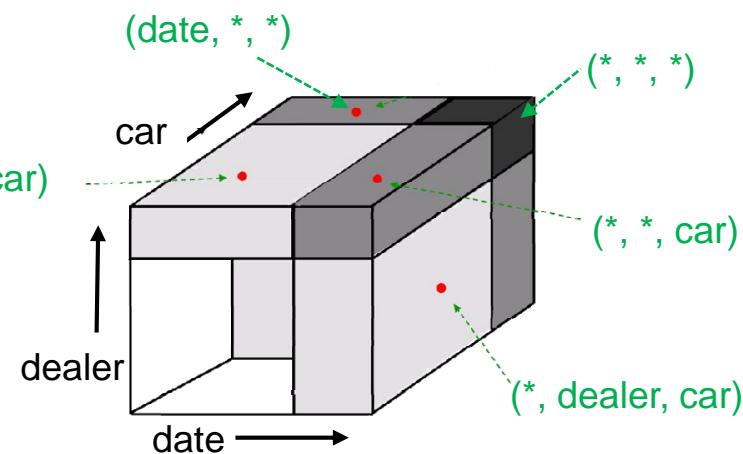
- Presentation layer provides the multidimensional view
- The OLAP server stores data in a multidimensional structure
 - Computation occurs in this layer during the loading step (not at query)



MOLAP (cont'd.)

Detour

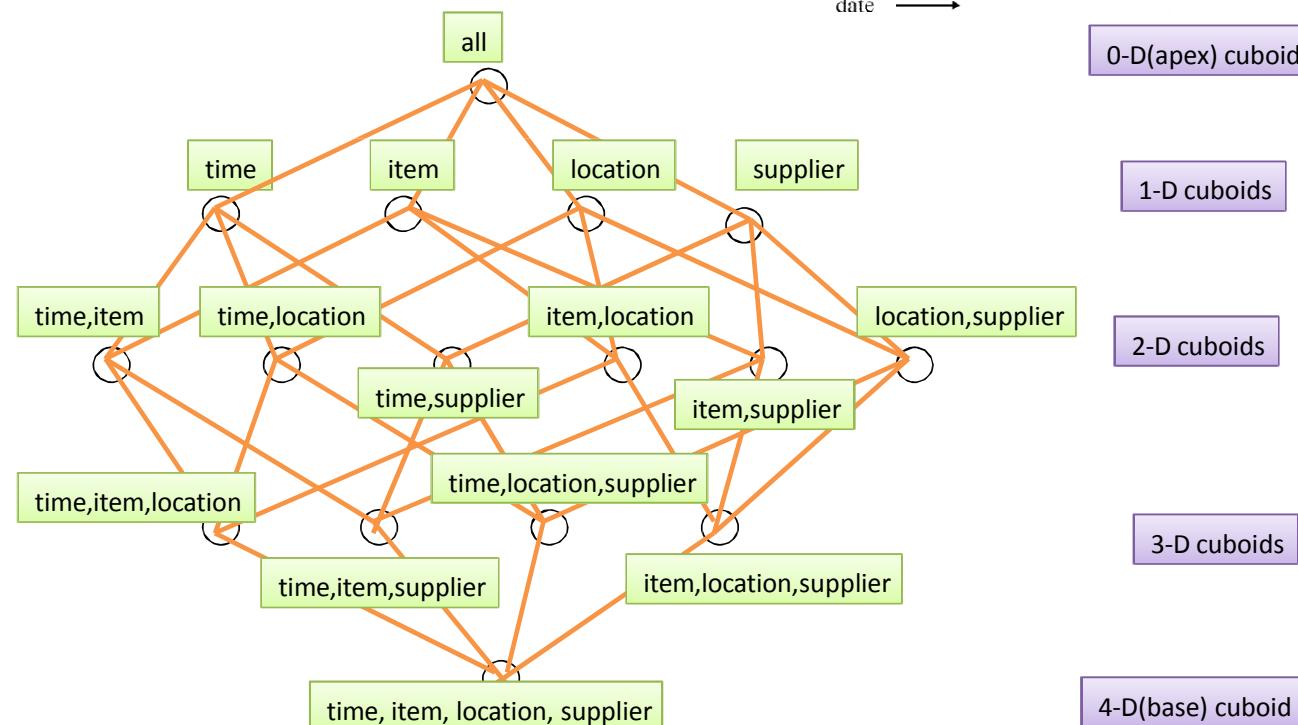
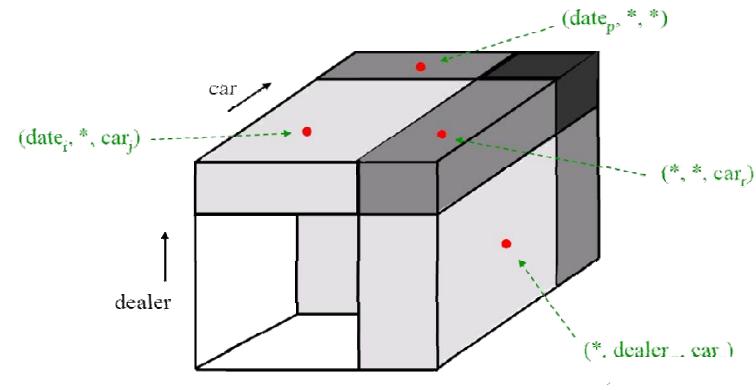
- Advantages
 - Excellent performance
 - Fast data retrieval
 - Optimal for slicing and dicing
 - Complex calculations
 - **All calculations are pre-generated when the cube is created**



MOLAP (cont'd.)

Detour

- All calculations are pre-generated when the cube is created



MOLAP (cont'd.)

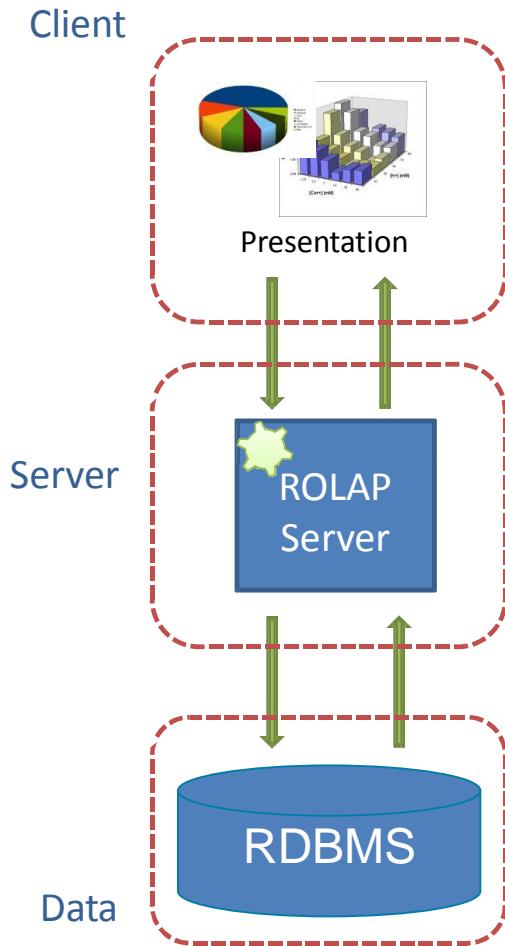
Detour

- Disadvantages
 - Limited amount of data it can handle – due to large calculations
 - Cube can be derived from large amount of data, but only **summary level information** will be included in the cube
 - Requires additional investment
 - Cube technology are often proprietary
 - Enormous amount of overhead
 - An input file of 200 MB can expand to 5 GB with calculations
- Products:
 - Cognos (IBM), Essbase (Oracle), Microsoft Analysis Service, Palo (open source)

ROLAP

Detour

- ROLAP
 - Presentation layer provides the multidimensional view
 - The ROLAP Server generates SQL queries, from the OLAP OLAP requests, to query the RDBMS
 - Data is stored in **RDBMs**



ROLAP (cont'd.)

Detour

- Special schema design: e.g., star, **snowflake**
- Special indexes: e.g., bitmap, R-Trees (next lecture)
- Advantages
 - Proven technology (relational model, DBMS)
 - Can handle large amounts of data (VLDBs)
- Disadvantages
 - Limited SQL functionalities
- Products
 - Microsoft Analysis Service, Siebel Analytics (now Oracle BI), Micro Strategy, Mondrian (open source)

MOLAP vs. ROLAP

Detour

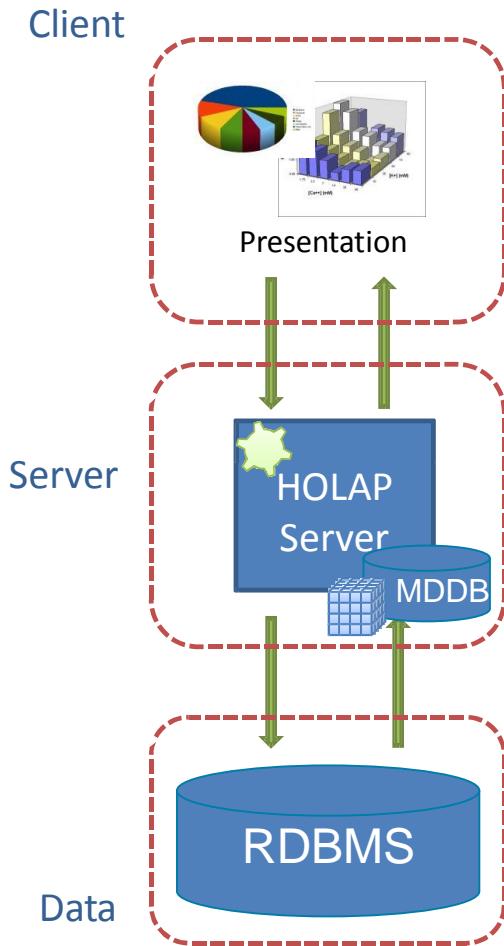
- Based on OLAP needs

OLAP needs		MOLAP	ROLAP
User Benefits	Multidimensional View	✓	✓
	Excellent Performance	✓	-
	Analytical Flexibility	✓	-
	Real-Time Data Access	-	✓
	High Data Capacity	-	✓
MIS Benefits	Easy Development	✓	-
	Low Structure Maintenance	-	✓
	Low Aggregate Maintenance	✓	-

HOLAP

Detour

- HOLAP
 - Best of both worlds
 - Storing detailed data in RDBs
 - Storing aggregated data in MDBs
 - Different partitioning approaches between MOLAP and ROLAP
 - Vertical
 - Horizontal

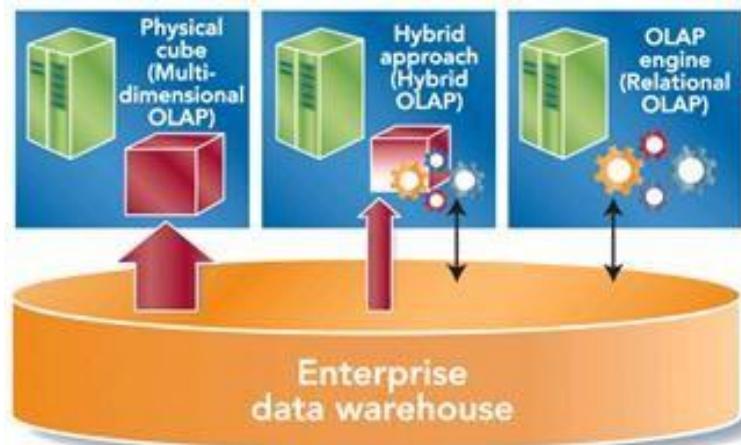


- Vertical partitioning
 - Aggregations are stored in MOLAP for fast query performance,
 - Detailed data in ROLAP to optimize time of cube processing (loading the data from the OLTP)
- Horizontal partitioning
 - HOLAP stores some slice of data, usually the more recent one (i.e. sliced by Time dimension) in MOLAP for fast query performance
 - Older data in ROLAP

HOLAP (cont'd.)

Detour

- Other approaches
 - Store some cubes in MOLAP and others in ROLAP, leveraging the fact that in a large cuboid, there will be dense and sparse sub-regions



Conclusions

Detour

- ROLAP
 - RDBMS - star/snowflake schema
 - For detailed and larger volumes of data (TB)
- MOLAP
 - MDBMS - Cube structures, array based storage
 - For summarized and relatively “small” volumes of data (50GB)
- HOLAP is emerging as the OLAP server of choice



Summary

Summary

- Physical Level
 - Relational Implementation through:
 - Star schema: improves query performance for often-used data
 - Less tables and simple structure
 - Efficient query processing with regard to dimensions
 - In some cases, high overhead of redundant data
 - Snowflake schema: reduce the size of the dimension tables
 - However, through dimension normalization - large number of tables

Summary (cont'd.)

Summary

- Physical Level
 - Array based storage
 - How to perform linearization
 - Problems:
 - Order of dimensions – solution:caching
 - Dense Cubes,Sparse Cubes - solution:2 level storage
 - MOLAP, ROLAP, HOLAP



Next Lecture

- DW Optimization / Indexes
 - Bitmap indexes
 - Tree based indexes
 - Hash indexes

