Business Intelligence, Data Warehousing and Multidimensional Databases

Torben Bach Pedersen

AALBORG UNIVERSITY

daisy

Center for Data-intensive Systems

Business Intelligence Overview

- Why Business Intelligence?
- Data analysis problems
- Data Warehouse (DW) introduction
- Analysis technologies that **use** the DW
  - OLAP
  - Data mining
  - Visualization

  - A good DW is a **prerequisite** for using these technologies

What is Business Intelligence?

- Combination of technologies
  - Data Warehousing (DW)
  - On-Line Analytical Processing (OLAP)
  - Data Mining (DM)
  - Data Visualization (VIS)
  - Decision Analysis (what-if)
  - Customer Relationship Management (CRM)
  - Vertical solutions composed of the base technologies
- Buzzword compliant (still ?)
  - Extension/integration of the technologies above

BI Is Important

- Palo Alto Management Group: BI = $113 bio. in 2002
- The Web makes BI more necessary
  - Customers do not appear ”physically” in the store
  - Customers can change to other stores more easily
- Thus:
  - Know your customers using data and BI!
  - Web logs makes is possible to analyze customer behavior in a more detailed than before (what was **not** bought?)
  - Combine web data with traditional customer data
- Next step is the Wireless Internet
  - Customers are always ”online”
  - Customer’s position is known
  - Combine position and customer knowledge => very valuable!

Data Analysis Problems

- The same data found in many different systems
  - Example: customer data in 14 (now 23) systems!
  - The same concept is defined differently (Nykredit)
- Data is suited for operational systems (OLTP)
  - Accounting, billing, etc.
  - Do not support analysis across business functions
- Data quality is bad
  - Missing data, imprecise data, different use of systems
- Data are ”volatile”
  - Data deleted in operational systems (6 months)
  - Data change over time – no historical information

Data Warehousing

- Solution: new analysis environment (DW) where data are
  - Subject oriented (versus function oriented)
  - Integrated (logically and physically)
  - Stable (data not deleted, several versions)
  - Time variant (data can always be related to time)
  - Supporting management decisions (different organization)
- Data from the operational systems are
  - Extracted
  - Cleansed
  - Transformed
  - Aggregated?
  - Loaded into DW
- ”Getting multidimensional data into the DW”
- A good DW is a **prerequisite** for successful BI

DW: Purpose and Definition

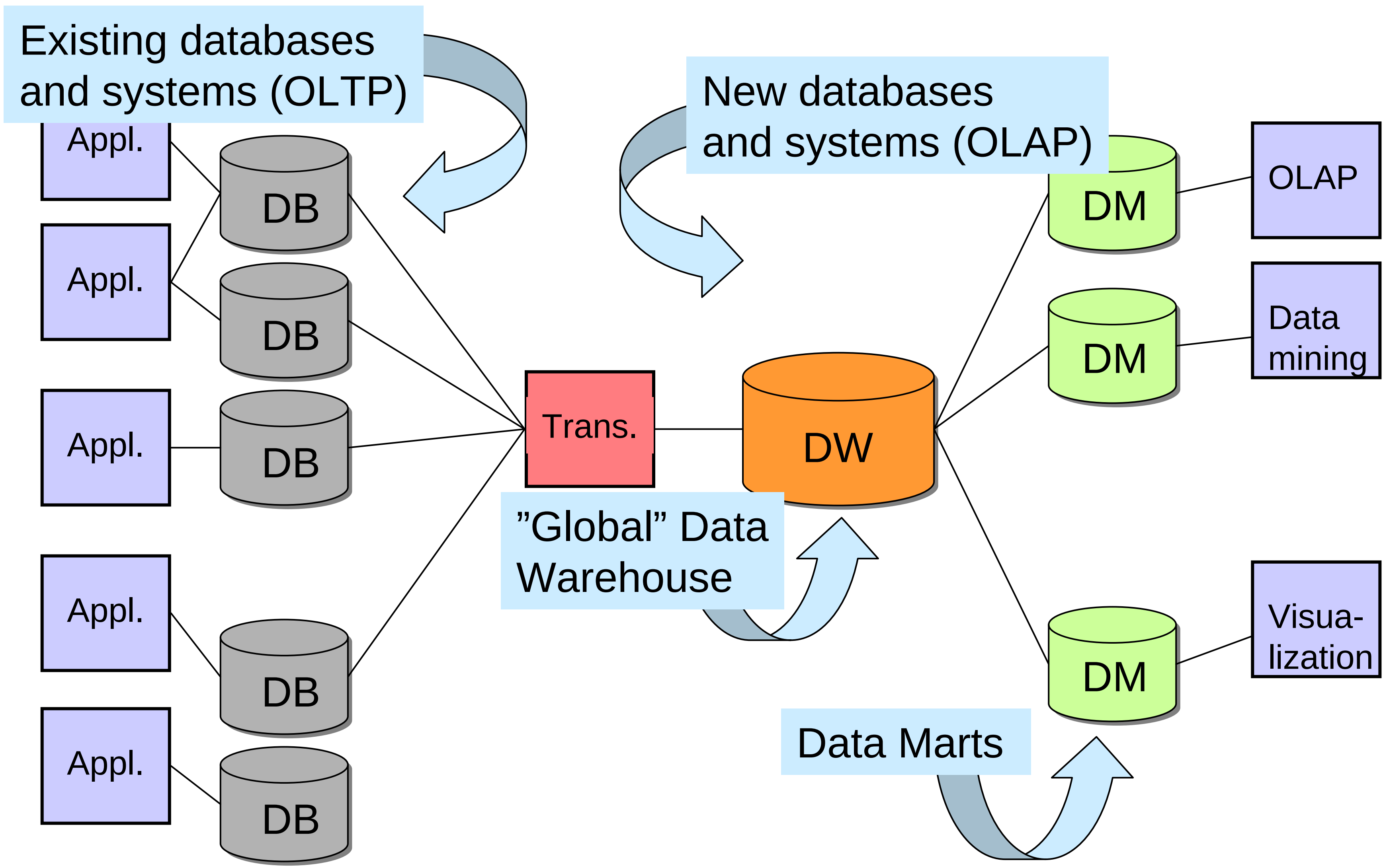
- The purpose of a data warehouse is to support **decision making**

- Data is collected from a number of different sources
  - Finance, billing, web logs, personnel, …

- It is made easy to perform advanced analyses
  - Ad-hoc analyses and reports
  - Data mining: identification of trends
  - Management Information Systems

- A data warehouse is a **store of information** organized in a unified data model.

DW Architecture – Data as Materialized Views

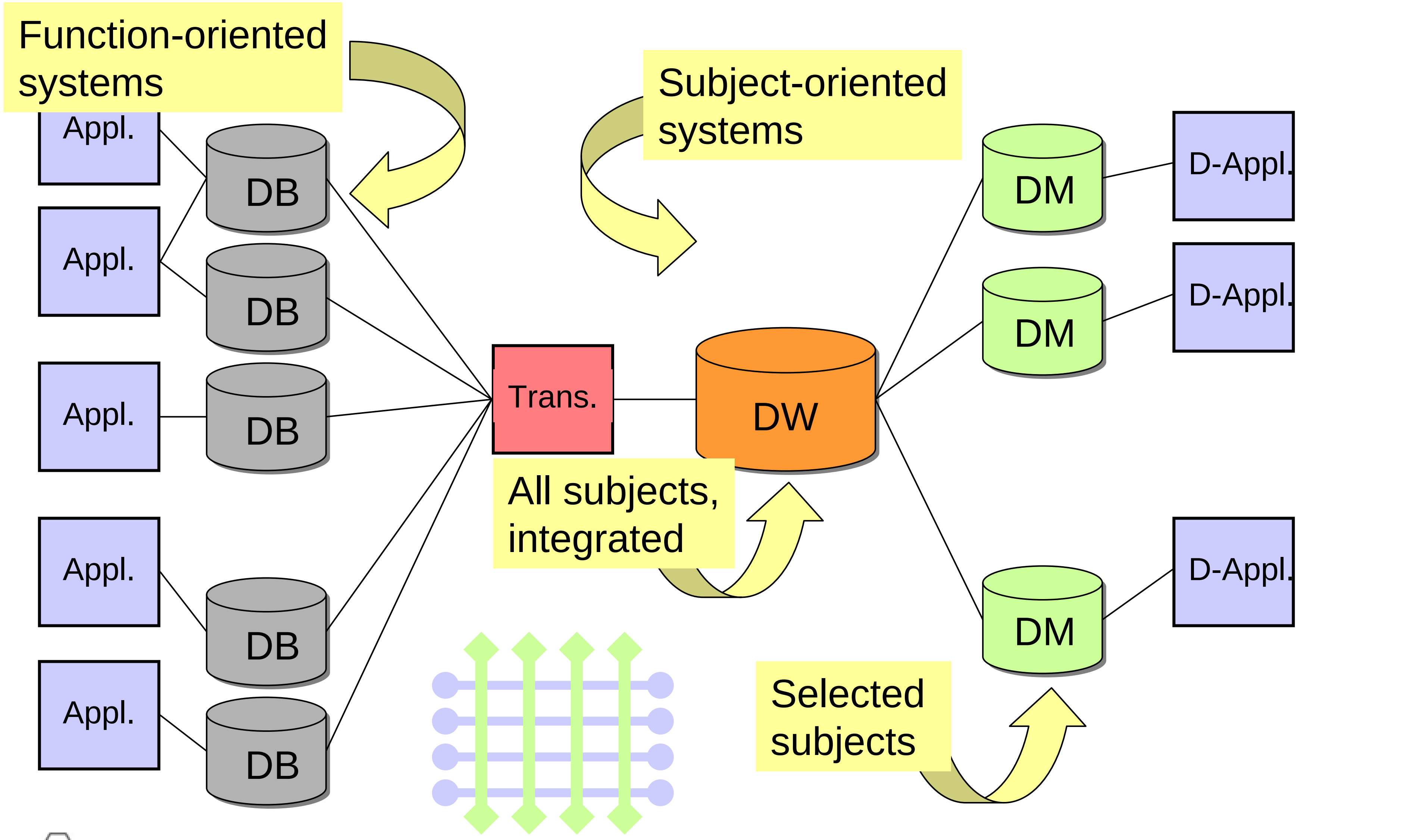


OLTP vs. OLAP

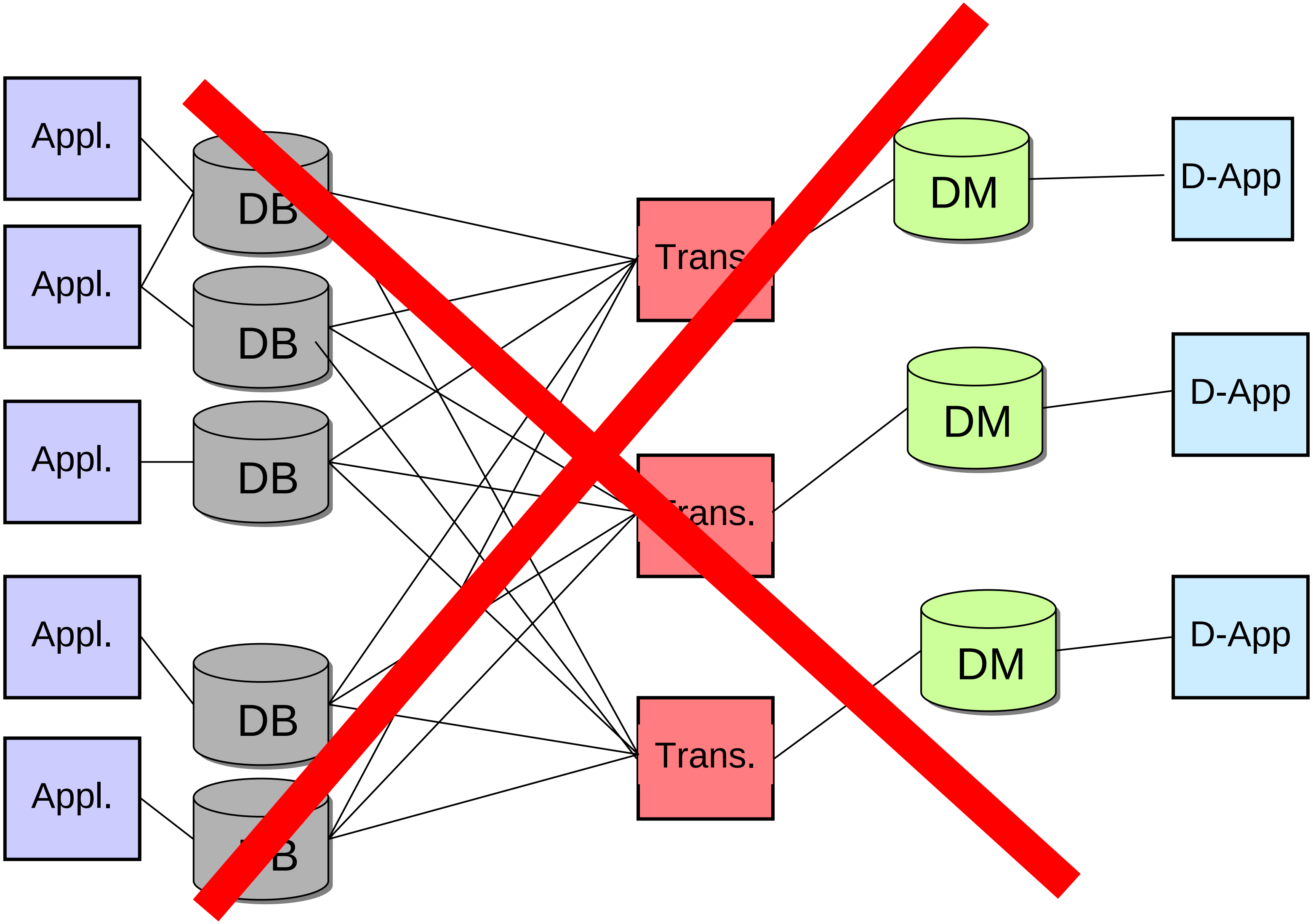
- On-Line Transaction Processing
  - Many, ”small” queries
  - Frequent updates
  - The system is always available for both updates and reads
  - Smaller data volume (few historical data)
  - Complex data model (normalized)

- On-Line Analytical Processing
  - Fewer, but ”bigger” queries
  - Frequent reads, in-frequent updates (daily)
  - 2-phase operation: either reading or updating
  - Larger data volumes (collection of historical data)
  - Simple data model (multidimensional/de-normalized)

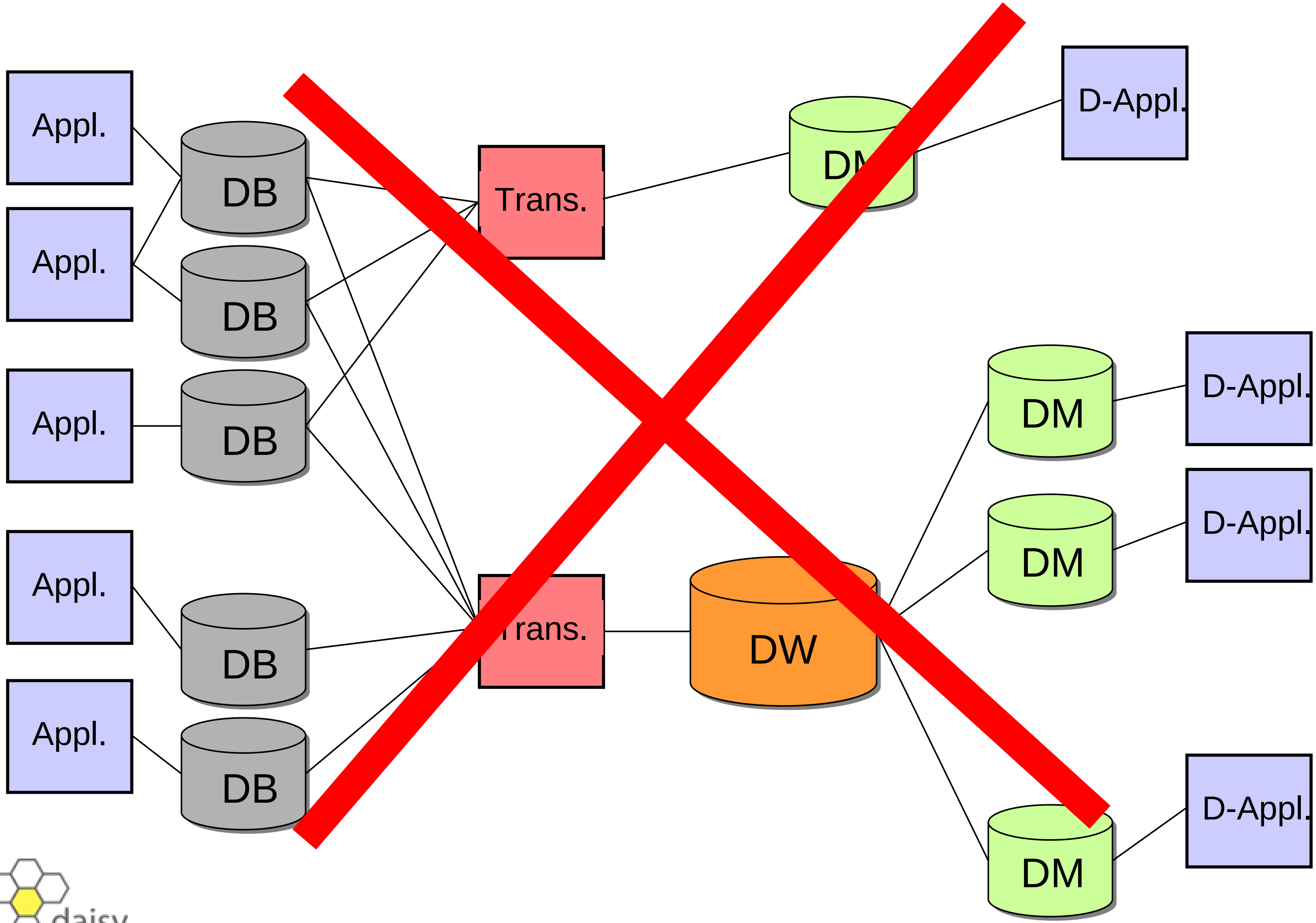
Function- vs. Subject Orientation



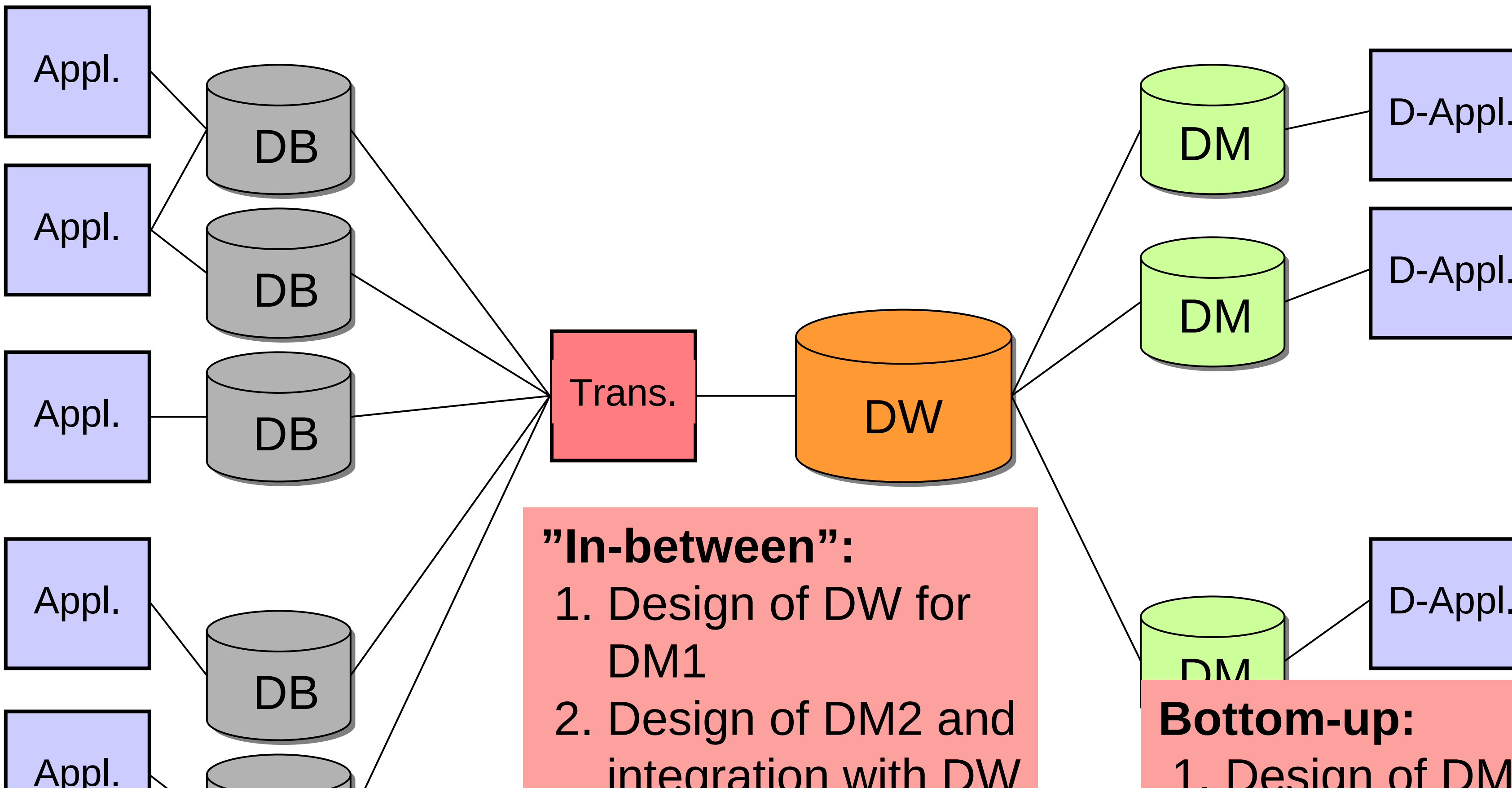
n x ***m*** versus ***n*** + ***m***



Architecture Alternative



Top-down vs. Bottom-up



"In-between":

1. Design of DW for DM1
2. Design of DM2 and integration with DW
3. Design of DM3 and integration with DW
4. ...

Top-down:

1. Design of DW
2. Design of DMs

Bottom-up:

1. Design of DMs
2. Maybe integration of DMs in DW
3. Maybe no DW

Data’s Way To The DW

- Extraction
  - Extract from many heterogeneous systems
- Staging area
  - Large, sequential bulk operations => flat files best ?
- Cleansing
  - Data checked for missing parts and erroneous values
  - Default values provided and out-of-range values marked
- Transformation
  - Data transformed to decision-oriented format
  - Data from several sources merged, optimize for querying
- Aggregation?
  - Are individual business transactions needed in the DW ?
- Loading into DW
  - Large bulk loads rather than SQL INSERTs
  - Fast indexing (and pre-aggregation) required

Common DW Issues

- Metadata management
  - Need to **understand** data = metadata needed
  - Greater need that in OLTP applications as ”raw” data is used
  - Need to know about:
    - Data definitions, dataflow, transformations, versions, usage, security
- DW project management
  - DW projects are **large** and **different** from ordinary SW projects
    - 12-36 months and 1+ mio. US$ per project
    - Data marts are smaller and ”safer” (bottom up approach)
  - Reasons for failure
    - Lack of proper design methodologies
    - High HW+SW cost (not so much anymore)
    - Deployment problems (lack of training)
    - Organizational change is hard… (new processes, data ownership,..)
    - Ethical issues (security, privacy,…)

daisy

BI Summary

- Why Business Intelligence?
- Data analysis problems
- Data Warehouse (DW) introduction
- Analysis technologies that **use** the DW
  - OLAP
  - Data mining
  - Visualization

- BI can provide many advantages to your organization
  - A good DW is a prerequisite for BI
  - But, a DW is a **means** rather than a **goal**…it is only when it is heavily used that success is achieved

Multidimensional Databases

Torben Bach Pedersen

Aalborg University

Overview

- Motivation
- Cubes
- Dimensions
- Facts
- Measures
- Data warehouse queries
- Relational design
- Redundancy
- Strengths and weaknesses of the multidimensional model
- Case study
    - The grocery store

Why a new model?

- We know E/R and OO modeling
- All types of data are “equal”
- E/R and OO models: many purposes
  - ***Flexible***
  - ***General***
- No difference between:
  - What **is** important
  - What just **describes** the important
- ER/OO models are **large**
  - 50-1000 entities/relations/classes
  - Hard to get an overview
- ER/OO models implemented in RDBMSes
  - Normalized databases **spread** information
  - When analyzing data, the information must be **integrated** again

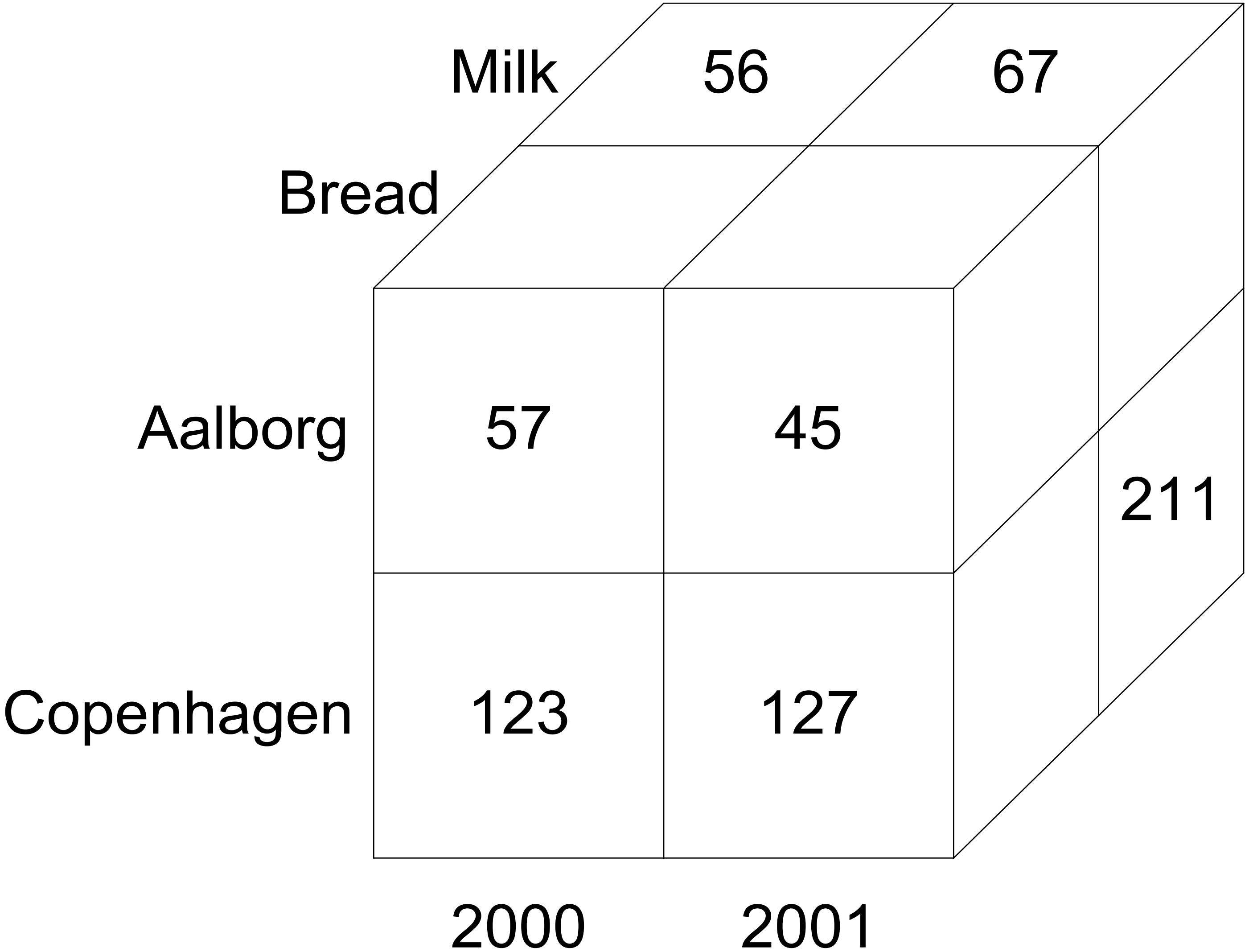
The multidimensional model

- One purpose
  - **Data analysis**
- Better at **that** purpose
  - Less flexible
  - Not suited for OLTP systems
- More **built in** “meaning”
  - What **is** important
  - What **describes** the important
  - What we want to **optimize**
  - Automatic aggregations means easy querying
- Recognized by OLAP/BI tools
  - Tools offer powerful query facilities based on MD design
  - Example: TARGIT Analysis

The multidimensional model

- Data is divided into:
  - **Facts**
  - **Dimensions**
- Facts are the **important** entity: a sale
- Facts have **measures** that can be aggregated: sales price
- Dimensions **describe** facts
  - A sale has the dimensions Product, Store and Time
- Facts “live” in a multidimensional **cube** (dice)
  - Think of an array from programming languages
- Goal for dimensional modeling:
  - Surround facts with as much context (dimensions) as possible
  - Hint: redundancy may be ok (in well-chosen places)
  - But you should **not** try to model **all** relationships in the data (unlike E/R and OO modeling!)

Cube Example



Cubes

- A “cube” may have **many** dimensions!
  - More than 3 - the term ”hypercube” is sometimes used
  - Theoretically no limit for the number of dimensions
  - Typical cubes have 4-12 dimensions
- But only 2-3 dimensions can be viewed at a time
  - Dimensionality reduced by queries via projection/aggregation
- A cube consists of **cells**
  - A given combination of dimension values
  - A cell can be empty (no data for this combination)
  - A **sparse** cube has few non-empty cells
  - A **dense** cube has many non-empty cells
  - Cubes become sparser for many/large dimensions

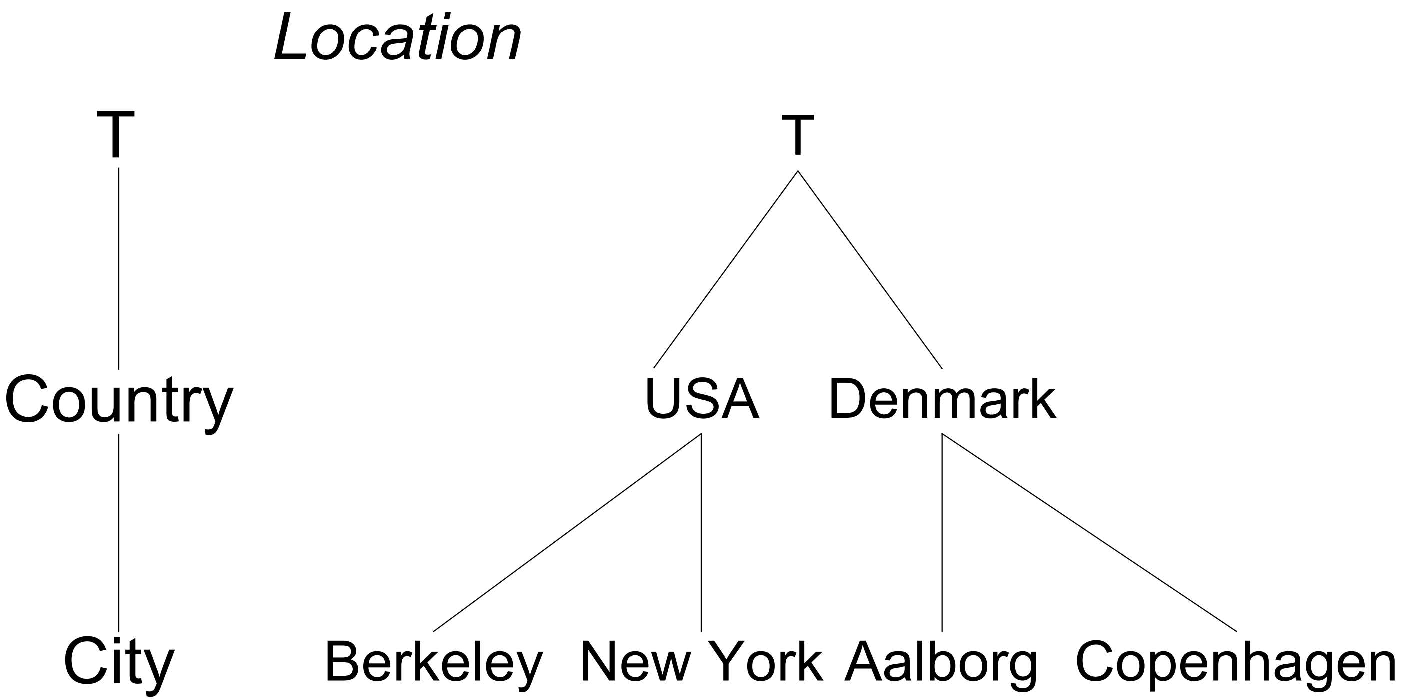
Dimensions

- Dimensions are the core of multidimensional databases
  - Other types of databases do not support dimensions
- Dimensions are used for
  - **Selection** of data
  - **Grouping** of data at the right level of detail
- Dimensions consist of **dimension values**
  - Product dimension have values ”milk”, ”cream”, …
  - Time dimension have values ”1/1/2001”, ”2/1/2001”,…
- Dimension values may have an **ordering**
  - Used for comparing cube data across values
  - Example: ”percent sales increase compared with last month”
  - Especially used for Time dimension

Dimensions

- Dimensions have **hierarchies** with **levels**
  - Typically 3-5 levels (of detail)
  - Dimension values are organized in a **tree structure**
  - **Product**: Product->Type->Category
  - **Store**: Store->Area->City->County
  - **Time**: Day->Month->Quarter->Year
  - Dimensions have a **bottom level** and a **top level** (ALL)
- Levels may have **attributes**
  - Simple, non-hierarchical information
  - Day has Workday as attribute
- Dimensions should contain much information
  - Time dimensions may contain holiday, season, events,…
  - Good dimensions have 50-100 or more attributes/levels

Dimension Example



Schema

Instance

Facts

- Facts represent the **subject** of the desired analysis
  - The ”important” in the business that should be analyzed
- A fact is most often identified via its dimension values
  - A fact is a non-empty cell
  - Some models give facts an explicit identity
- Generally a fact should
  - Be attached to **exactly one** dimension value in each dimension
  - Only be attached to dimension values in the bottom levels
  - Some models do not require this

Types Of Facts

- **Event** fact (transaction)
  - A fact for every **business event** (sale)
- ”**Fact-less**” facts
  - A fact per event (customer contact)
  - **No** numerical measures
  - An event has happened for a given dimension value combination
- **Snapshot** fact
  - A fact for every dimension combination at given time intervals
  - Captures **current** status (inventory)
- **Cumulative snapshot** facts
  - A fact for every dimension combination at given time intervals
  - Captures **cumulative** status up to now (sales in year to date)
- Every type of facts answers **different** questions
  - Often both event facts and both kinds of snapshot facts exist

Granularity

- **Granularity** of facts is important
  - What does a single fact mean?
  - **Level of detail**
  - Given by combination of bottom levels
  - Example: ”total sales per store per day per product”
- Important for number of facts
  - Scalability
- Often the granularity is a single business transaction
  - Example: sale
  - Sometimes the data is aggregated (**total** sales per store per day per product)
  - Might be necessary due to scalability
- Generally, transaction detail can be handled
  - Except perhaps huge clickstreams etc.

Measures

- Measures represent the fact property that the users want to **study and optimize**
  - Example: total sales price
- A measure has two components
  - **Numerical value**: (sales price)
  - **Aggregation formula** (SUM): used for aggregating/combining a number of measure values into one
  - Measure value determined by dimension value combination
  - Measure value is meaningful for all aggregation levels
- Most multidimensional models have measures
  - A few do not

Types Of Measures

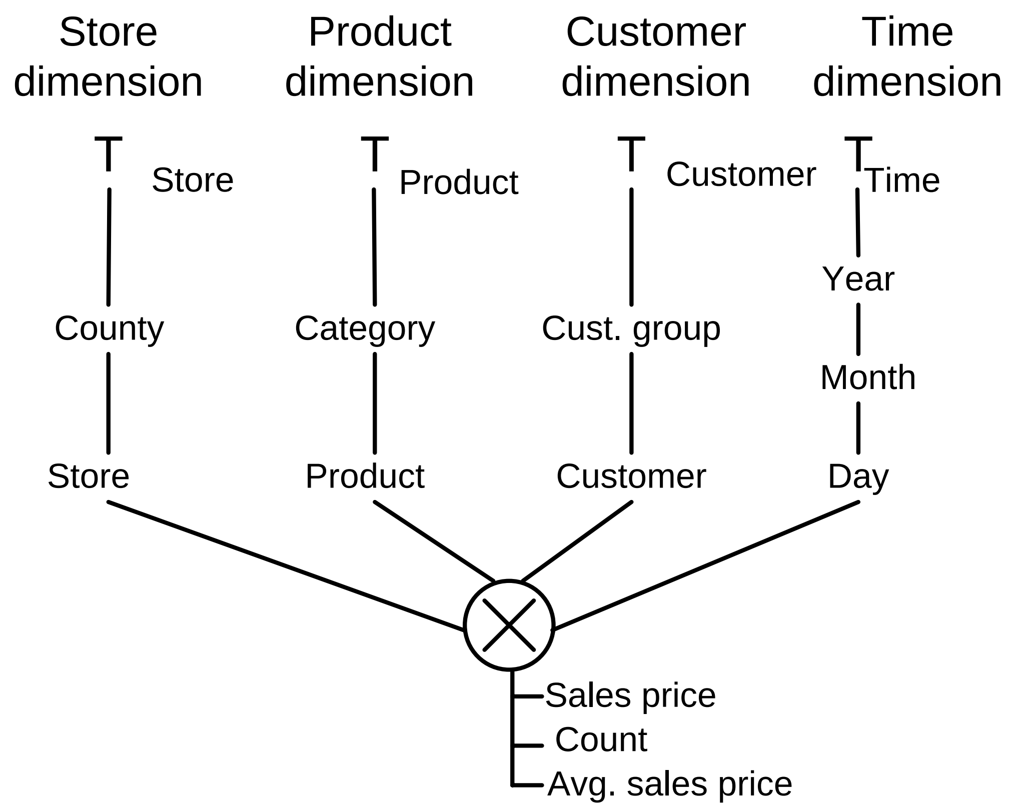
- Three types of measures

- Additive
  - Can be aggregated over **all** dimensions
  - Example: **sales price**
  - Often occur in event facts
- Semi-additive
  - **Cannot** be aggregated over **some** dimensions - typically time
  - Example: **inventory**
  - Often occur in snapshot facts
- Non-additive
  - **Cannot** be aggregated over **any** dimensions
  - Example: **average sales price**
  - Occur in all types of facts

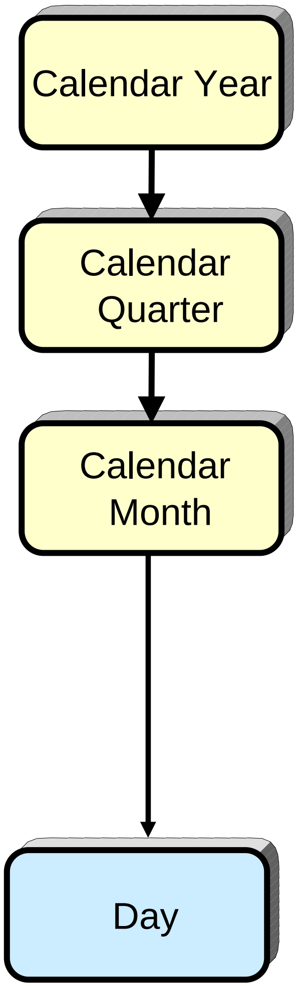
Documentation Of Schema

- No well-defined standard

- Our own notation
  - Seen to the right
  - T level corresponds to ALL
- Modeling and OLAP tools have their own notation



Kimball Dimension Notation



- The granularity is Day
- There is an implicit ”top” value which means ”all days” or ”the whole time axis”.
  - This is selected by not mentioning the dimension in a query

ROLAP

- Relational OLAP
- Data stored in relational tables
  - Star (or snowflake) schemas used for modeling
  - SQL used for querying
- Pros
  - Leverages investments in relational technology
  - Scalable (billions of facts)
  - Flexible, designs easier to change
  - New, performance enhancing techniques adapted from MOLAP
    - Indices, materialized views, special treatment of star schemas
- Cons
  - Storage use (often 3-4 times MOLAP)
  - Response times

MOLAP

- Multidimensional OLAP
- Special multidimensional data structures used
- Pros
  - Less storage use (“foreign keys” not stored)
  - Faster query response times
- Cons
  - Up till now not so good scalability (changing)
  - Less flexible, e.g., cube must be re-computed when design changes
  - Does not reuse an existing investment (but often bundled with RDBMS)
  - ”New technology”
  - Not as open technology

HOLAP

- Hybrid OLAP
- Aggregates stored in multidimensional structures (MOLAP)
- Detail data stored in relational tables (ROLAP)
- Pros
  - Scalable
  - Fast
- Cons
  - Complexity

Relational Implementation

- The cube is often implemented in an RDBMS
- Fact table stores facts
  - One column for each measure
  - One column for each dimension (foreign key to dimension table)
  - Dimensions keys make up composite primary key
- Dimension table stores dimension
  - Integer key column (surrogate keys)
  - Don't use production keys in DW!
- Goal for dimensional modeling: ”surround the facts with as much context (dimensions) as we can”
- **Granularity** of the fact table is important
  - What does one fact table row represent ?
  - Important for the size of the fact table
  - Often corresponding to a single business transaction (sale)
  - But it can be aggregated (sales per product per day per store)

Relational Design

- One completely de-normalized table
  - Bad: inflexibility, storage use, bad performance, slow update
- Star schemas
  - One fact table
  - De-normalized dimension tables
  - One column per level/attribute
- Snowflake schemas
  - Dimensions are normalized
  - One dimension table per level
  - Each dimension table has integer key, level name, and one column per attribute

Star Schema Example

ProductID	Product	Type	Category
1	Top	Beer	Beverage

TimeID	Day	Month	Year
1	25.	Maj	1997

ProductId	StoreId	TimeId	Sale
1	1	1	5.75

StoreID	Store	City	County
1	Trøjborg	Århus	Århus

Snow-flake Schema Example

TypeID	Type	CategoryID
1	Beer	1

ProductID	Product	TypeID
1	Top	1

MonthID	Month	YearID
1	May	1

TimeID	Day	MonthID
1	25.	1

ProductId	StoreId	TimeId	Sale
1	1	1	5.75

StoreID	Store	CityID
1	Trøjborg	1

CityID	City	CountyId
1	Århus	1

(Relational) OLAP Queries

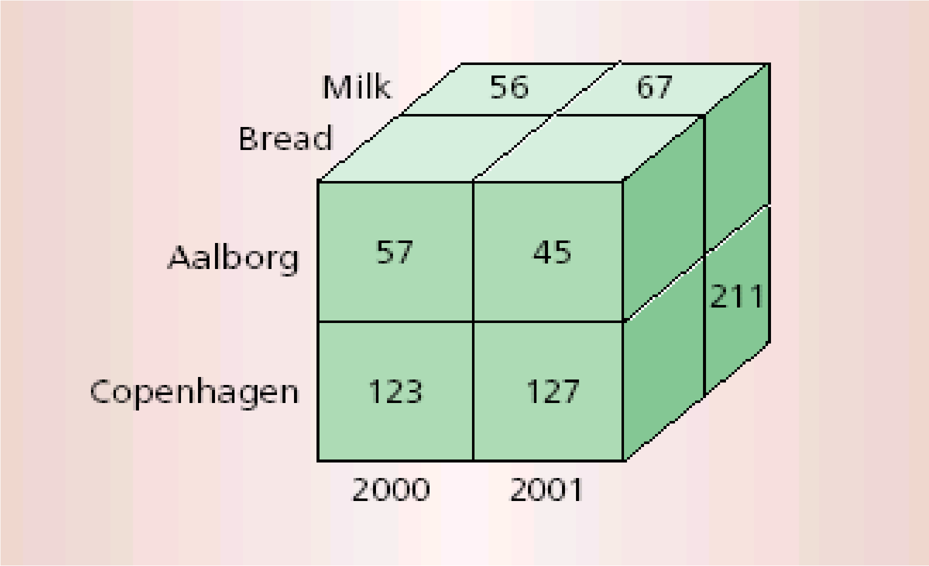
- **Aggregating** data, e.g., with SUM
- **Starting level**: (Quarter, Product)
- **Roll Up**: less detail, Quarter->Year
- **Drill Down**: more detail, Quarter->Month
- **Slice/Dice**: selection, Year=1999
- **Drill Across**: “join” on common dimensions
- **Visualization** and **exceptions**
- Note: only **two** kinds of queries
  - **Navigation queries** examine one dimension
    - SELECT DISTINCT l FROM d [WHERE p]
  - **Aggregation queries** summarize fact data
    - SELECT d1.l1,d2.l2,SUM(f.m) FROM d1,d2,f
      WHERE f.dk1=d1.dk1 AND f.dk2=d2.dk2 [AND p]
      GROUP BY d1.l1,d2.l2

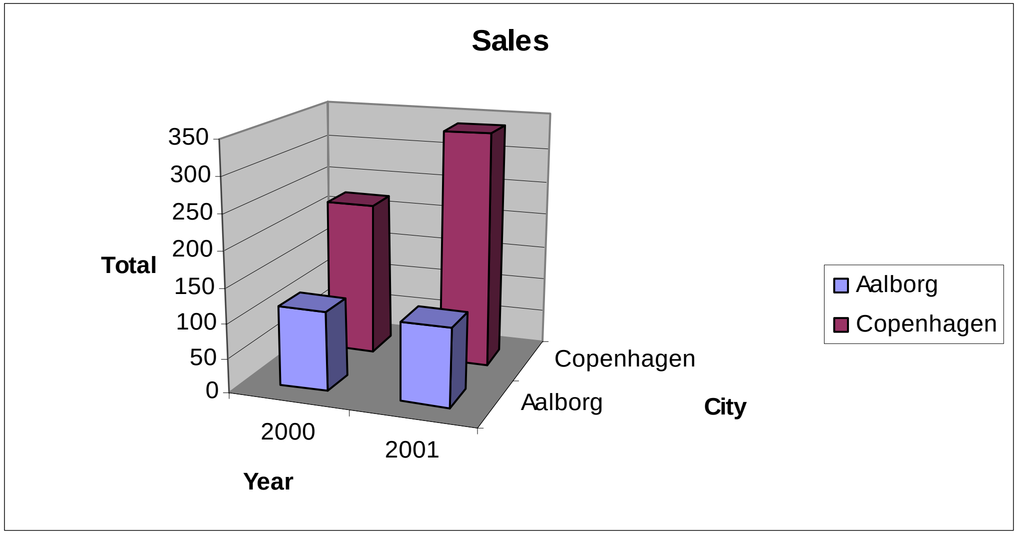
OLAP Queries

- Fast, interactive analysis of large amounts of data
  - Sales, web, …
- "Spreadsheets on stereoids"

- **Aggregation queries**
  - **Per City and Year**
- Roll up - get overview
- Drill down – more detail

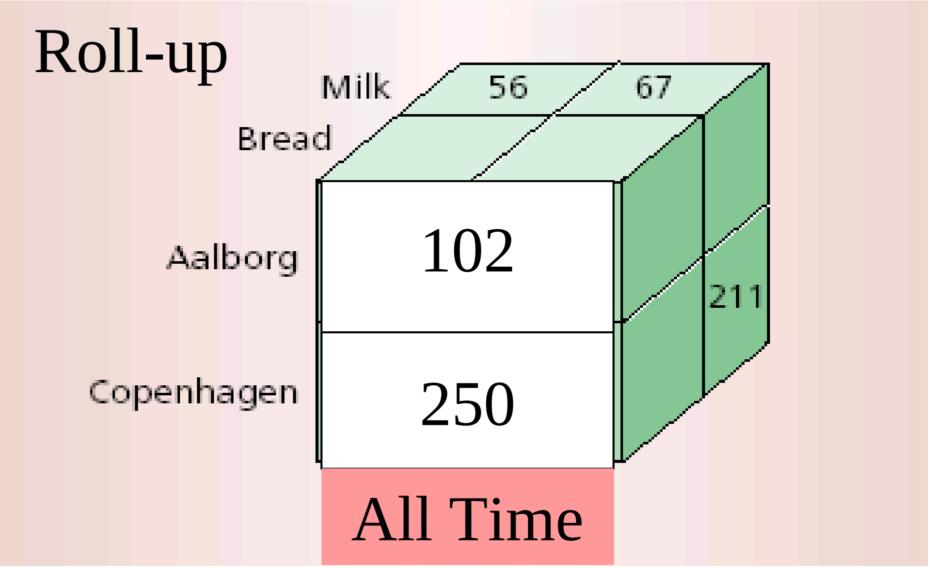
- Fast answers required
  - A few seconds response time even for many gigabytes data
  - Achieved by pre-computation (pre-aggregation)

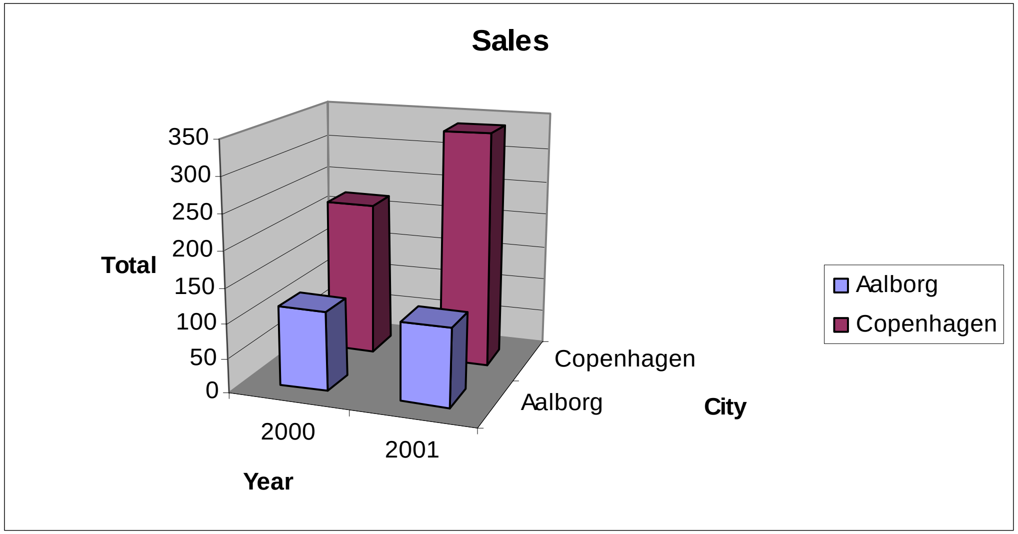




OLAP Queries

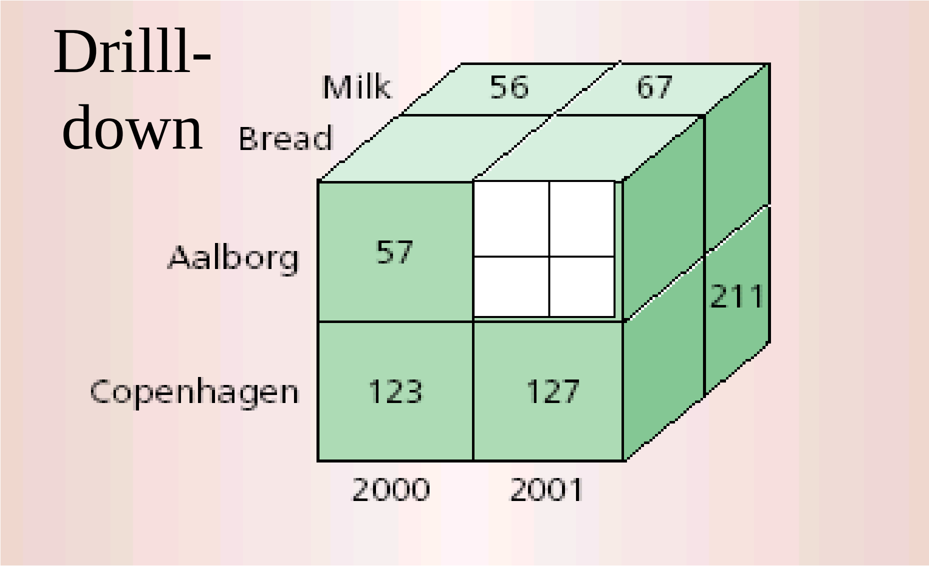
- Fast, interactive analysis of large amounts of data
  - Sales, web, …
- "Spreadsheets on stereoids"
- Aggregation queries
  - Per City and Year
- **Roll up - get overview**
- Drill down – more detail
- Fast answers required
  - A few seconds response time even for many gigabytes data
  - Achieved by pre-computation (pre-aggregation)

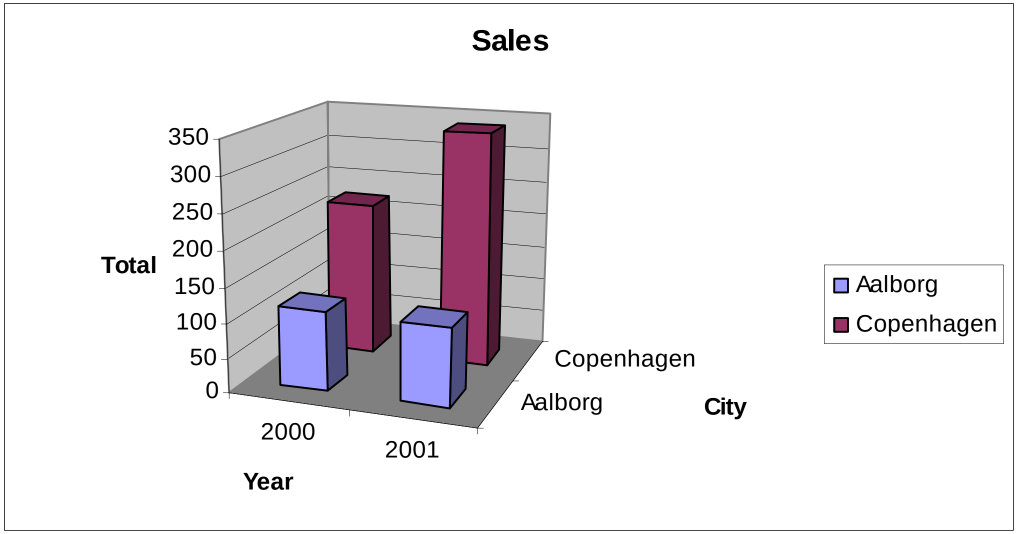




OLAP Queries

- Fast, interactive analysis of large amounts of data
  - Sales, web, …
- “Spreadsheets on stereoids”
- Aggregation queries
  - Per City and Year
- Roll up - get overview
- **Drill down – more detail**
- Fast answers required
  - A few seconds response time even for many gigabytes data
  - Achieved by pre-computation (pre-aggregation)





Star Schemas

- \+ Simple and easy overview -> ease-of-use
- \+ Relatively flexible
- \+ Fact table is normalized
- \+ Dimension tables often relatively small
- \+ “Recognized” by many RDBMSes -> good performance
- \- Hierarchies are ”hidden” in the columns
- \- Dimension tables are de-normalized

Snow-flake Schemas

- \+ Hierarchies are made explicit/visible
- \+ Very flexible
- \+ Dimension tables use less space
- \- Harder to use due to many joins
- \- Worse performance

Redundancy In The DW

- Only very little redundancy in fact tables
  - Order head data copied to order line facts
  - The same fact data (generally) only stored in one fact table
- Redundancy is mostly in dimension tables
  - Star dimension tables have redundant entries for the higher levels
- Redundancy problems?
  - Inconsistent data – the central load process helps with this
  - Update time – the DW is optimized for querying, not updates
  - Space use: dimension tables typically take up less than 5% of DW
- So: **controlled** redundancy is good
  - Up to a certain limit

Limits – And Strengths

- Many-to-one relationship from fact to dimension
- Many-to-one relationships from lower to higher levels in the hierarchies
- Therefore, it is impossible to ”count wrong”
- Hierarchies have a fixed height
- Hierarchies don’t change?

The Grocery Store

- Stock Keeping Units (SKUs)
- Universal Product Codes (UPCs)
- Point Of Sale (POS) system
- Stores
- Promotions

DW Design Steps

- Choose the **business process(es)** to model
  - Sales
- Choose the **grain** of the business process
  - SKU by Store by Promotion by Day
  - Low granularity is needed
  - Are individual transactions necessary/feasible ?
- Choose the **dimensions**
  - Time, Store, Promotion, Product
- Choose the **measures**
  - Dollar_sales, unit_sales, dollar_cost, customer_count

- Resisting normalization and preserving browsing
  - Flat dimension tables makes browsing easy and fast

The Grocery Store Dimensions

- The Time dimension
  - Explicit time dimension is needed (events, holidays,..)
- The Product dimension
  - Six-level hierarchy allows drill-down/roll-up
  - **Many** descriptive attributes (often more than 50)
- The Store dimension
  - Many descriptive attributes
  - The Time dimension is an **outrigger** table (First opened,..)
- The Promotion dimension
  - Example of a **causal** dimension
  - Used to see if promotions work/are profitable
  - Ads, price reductions, end-of-aisle displays, coupons
    - Highly correlated (only 5000 combinations)
    - Separate dimensions ? (size&efficiency versus simplicity&understanding)

Time Dimension

- The Time dimension
- Explicit time dimension is needed
- Fiscal years
- Events
- Holidays
- …

TimeID
DayNoInMonth
Month
Quarter
Year
FiscalPeriod
DayNumberInYear
DayNumberOverall
MonthNumberInYear
MonthNumberOverall
Season/weather
Events
LastDayOfMonth
Holiday
…

Product Dimension

- The Product dimension
- Six-level hierarchy allows drill-down/roll-up
- **Many** descriptive attributes (often more than 50)
- Calculate sales per shelf space!

ProductID
SKU-Number
SKU_Description
Brand
Diet
Subcategory
Category
Department
ShelfWidth
ShelfHeight
ShelfDepth
PackageSize
RetailCaseSize
Weight
…

Store Dimension

- The Store dimension
- Many descriptive attributes
- The Time dimension is an **outrigger** table (First opened,..)

StoreID
StreetAddress
Phone
Fax
Email
Manager
ZIP
City
County
SalesArea
Floorplan
Area_sqft
First_opened
Photo_processing
…

Promotion Dimension

- Example of a **causal** dimension
- Used to see if promotions work/are profitable
- Ads, price reductions, end-of-aisle displays, coupons
- Highly correlated (only 5000 combinations)
- Separate dimensions ? (size&efficiency versus simplicity&understanding)
- Start+EndDate outrigger to Time dimension

PromotionID
PromotionName
Ads
AdMedia
Displays
PriceReduction
Coupons
StartDate
EndDate
Cost
…

The Grocery Store Measures

- Dollar_sales
- Unit_sales
- Dollar_cost
- All **additive** across all dimensions
- Gross profit
  - Computed from sales and cost
  - Additive
- Gross margin
  - Computed from gross profit and sales
  - **Non-additive** across all dimensions
- Customer_count
  - Additive across time, promotion, and store
  - **Non-additive** across product
  - **Semi-additive**

Database Sizing

- Time dimension: 2 years = 730 days
- Store dimension: 300 stores reporting each day
- Product dimension: 30,000 products, only 3000 sell per day
- Promotion dimension: 5000 combinations, but a product only appears in one combination per day
- Number of fact records: 730*300*3000*1 = 657,000,000
- Number of fields: 4 key + 4 fact = 8 fields
- Total DB size: 657,000,000 * 8 fields * 4 bytes = 21 GB
- **Small** database by today's standards?
- Transaction level detail is feasible today

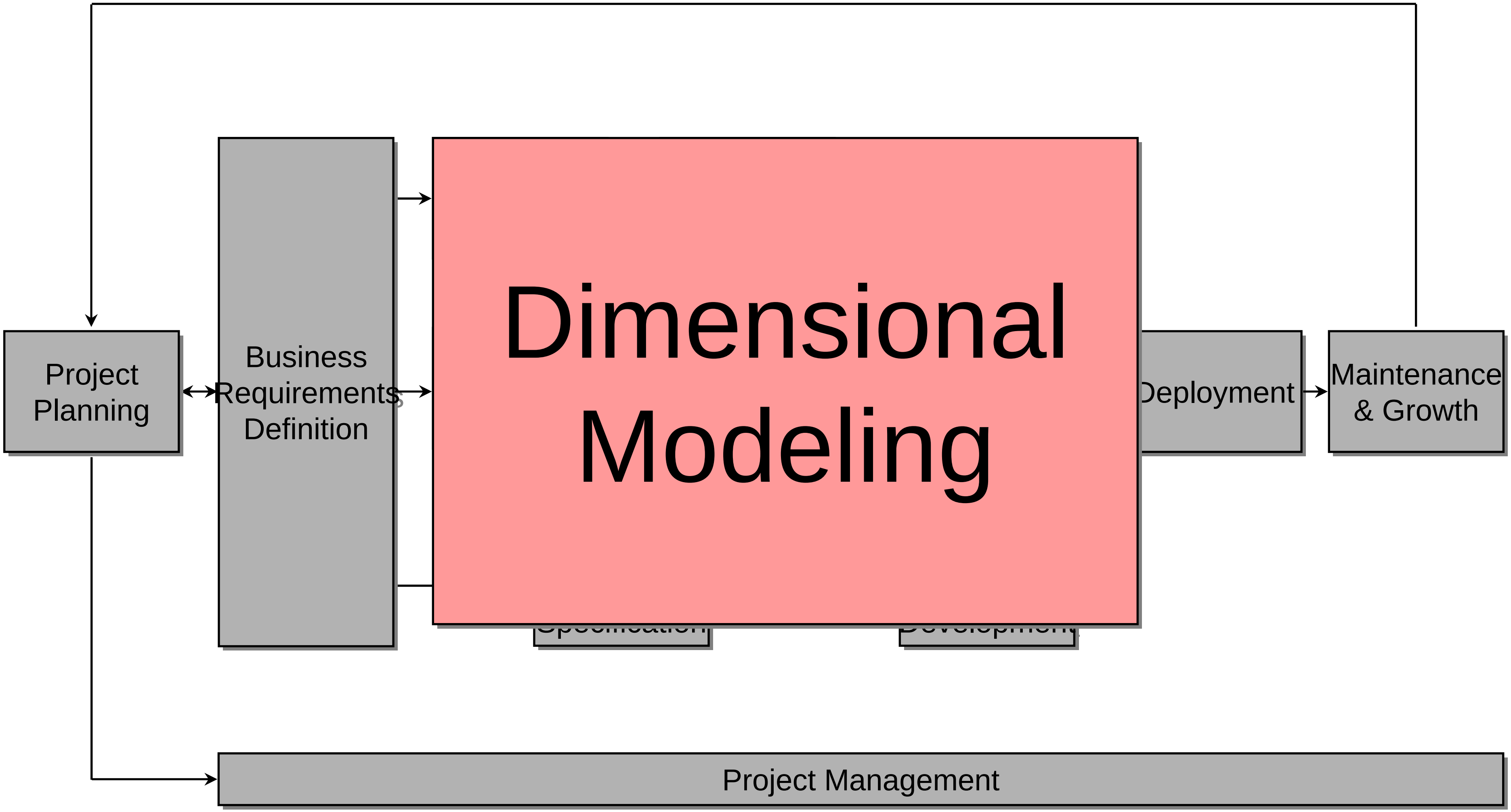
Typical Fact Tables (Again)

- Event/transaction table
  - One record for every business event (sale)
- Snapshot table
  - One record for every dimension combination at given time intervals
  - Records **current** status (inventory)
  - Often, both event and snapshot tables are needed
- Cumulative snapshot table
  - One record for every dimension combination at given time intervals
  - Records **cumulative** status up till now (sales in year to date)
- Fact-less fact table
  - One record per event (customer contact)
  - **No** numeric measures
  - Used to capture that an event has happened for a particular dimension combination

MD Summary

- Motivation
- Cubes
- Dimensions
- Facts
- Measures
- Data warehouse queries
- Relational design
- Redundancy
- Strengths and weaknesses of the multidimensional model
- Case study
  - The grocery store

Business Dimensional Lifecycle



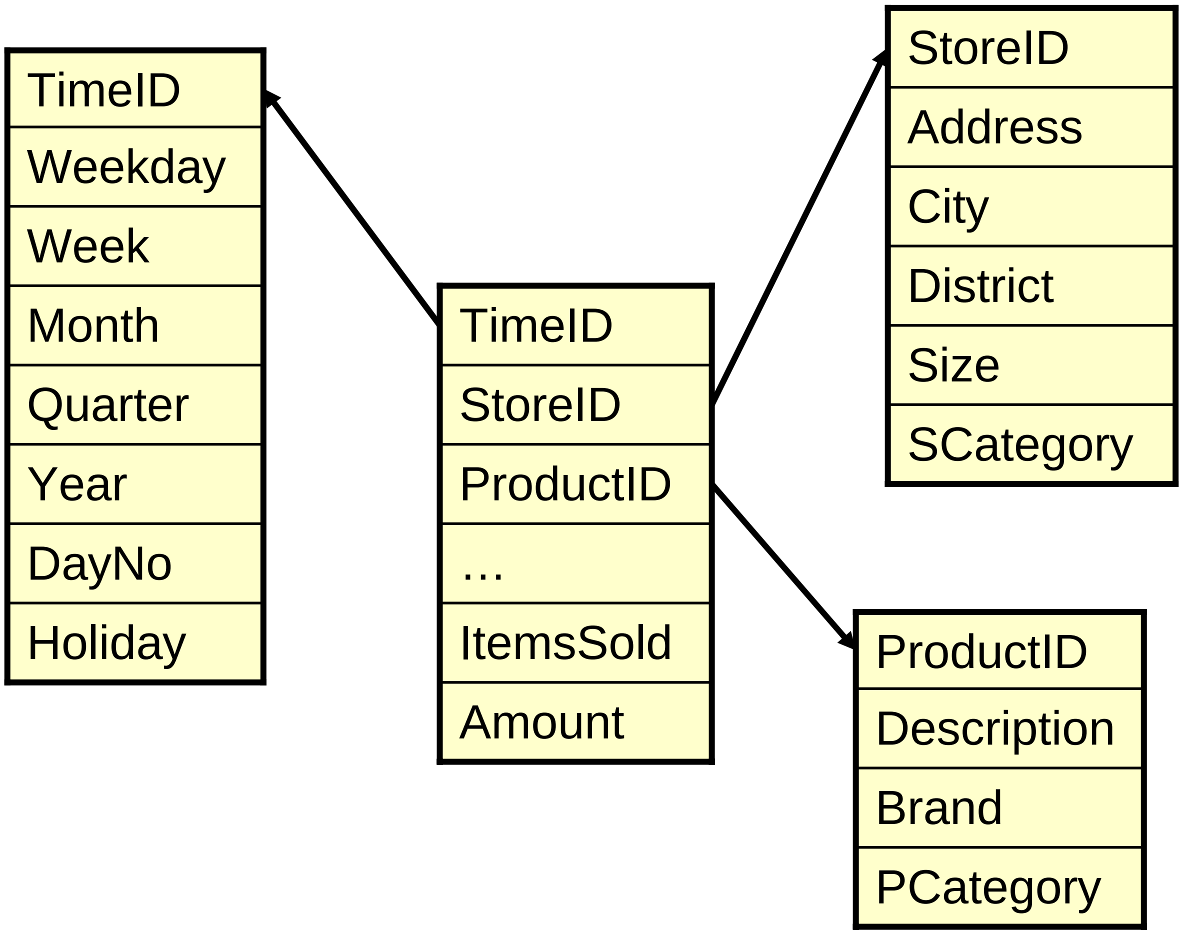
Advanced MD modeling I - Overview

- Handling change over time
- Changes in dimensions
  - No special handling
  - Versioning dimension values
  - Capturing the previous and the actual value
  - Timestamping
  - Split into changing and constant attributes

Changing Dimensions I

- So far, we have implicitly assumed that dimensions are stable over time.
  - At most, new rows in dimension tables are inserted.
  - The existing rows do not change.
- This assumption is not valid in practice.
  - The phenomenon is called “slowly changing dimensions”.
  - The intuition is, that dimension information change, but changes are (relatively) rare.
- We will look at a number of techniques for handling changes in dimensions.
- Schema changes are not considered now.
  - Then it becomes really funny!

Changing Dimensions II



- Descriptions of stores and products vary over time.
- A store is enlarged and changes Size.
- A product changes Description.
- Districts are changed.

- Problems
  - If we update the dimensions, wrong information will result.
  - If we don’t update the dimensions, the DW is not up-to-date.

Changing Dimensions III

StoreID	…	ItemsSold	…
001		2000	

StoreID	…	Size	…
001		250	

StoreID	…	ItemsSold	…
001		2000	

StoreID	…	Size	…
001		450	

StoreID	…	ItemsSold	…
001		2000	
001		2500	

StoreID	…	Size	…
001		450	

Changing Dimensions IV

- **Solution 1**: Overwrite the old values that change, in the dimension tables.
- Consequences
  - Old facts point to rows in the dimension tables with incorrect information.
  - New facts point to rows with correct information.
    - New facts are facts that are inserted after the dimension rows they point to are inserted/changed.
- Pros
  - Easy to implement
  - Ideal if the changes are due to erroneous registrations.
  - In some cases, the ”imprecision” can be disregarded.
- Cons
  - ”The solution” does not solve the problem of capturing change.

Changing Dimensions V

- **Solution 2**: Versioning of rows with changing attributes.
  - The *key* that links dimension and fact table, should now identify a *version o*f a row, not just a ”row”.
  - The key is generalized.
  - If ”stupid” (”non information-bearing”, ”surrogate”) keys are used, there is no need for changes.
- Consequences
  - Larger dimension tables
- Pros
  - Correct information captured in DW
  - No problems when formulating queries
- Cons
  - It is not possible to capture the development over time of the subjects the dimensions describe.

Changing Dimensions VI

StoreID	…	ItemsSold	…
001		2000	

StoreID	…	Size	…
001		250	

StoreID	…	ItemsSold	…
001		2000	

StoreID	…	Size	…
001		250	
002		450	

StoreID	…	ItemsSold	…
001		2000	
002		2500	

StoreID	…	Size	…
001		250	
002		450	

Changing Dimensions VII

- **Solution 3**: Create two versions of each changing attribute
  - One attribute contains the actual value
  - The other attribute contains the previous value
- Consequences
  - Two values are attached to each fact row.
- Pros
  - It is possible to compare across the change in dimension value (which is a problem with Solution 2).
  - Such comparisons are interesting in certain situations, where it is logical to work simultaneously with two alternative values.
  - Example: Categorization of stores and products.
- Cons
  - Not possible to see when the old value changed to the new.
  - Only possible to capture the two latest values.

Changing Dimensions VIII

StoreID	…	ItemsSold	…
001		2000	

StoreID	…	DistrictOld	DistrictNew	…
001		37	37	

StoreID	…	ItemsSold	…
001		2000	

StoreID	…	DistrictOld	DistrictNew	…
001		37	73	

StoreID	…	ItemsSold	…
001		2000	
001		2100	

StoreID	…	DistrictOld	DistrictNew	…
001		37	73	

Changing Dimensions IX

- **Solution 2.1**: Use special facts for capturing changes in dimensions via the Time dimension.
  - When a change occurs and there is no simultaneous, new fact referring to the new dimension row, a new special fact is create that points to the new dimension row and thus timestamps the row via the fact row’s reference to the Time dimensions.
- Pros
  - It is possible to capture the development over time of the subjects that the dimensions describe.
- Cons
  - Even larger tables

Changing Dimensions X

StoreID	TimeID	…	ItemsSold	…
001	234		2000	

StoreID	…	Size	…
001		250	

StoreID	TimeID	…	ItemsSold	…
001	234		2000	
002	345		-	

StoreID	…	Size	…
001		250	
002		450	

StoreID	TimeID	…	ItemsSold	…
001	234		2000	
002	345		-	
002	456		2500	

StoreID	…	Size	…
001		250	
002		450	

Changing Dimensions XI

- **Solution 2.2**: Versioning of rows with changing attributes like in Solution 2 + timestamping of rows.
- Pros
  - Correct information captured in DW
- Cons
  - Larger dimension tables
  - Consider whether Time dimension values and timestamps describe the same aspect of time.

Changing Dimensions XII

StoreID	TimeID	…	ItemsSold	…
001	234		2000	

StoreID	Size	From	To
001	250	98	-

StoreID	TimeID	…	ItemsSold	…
001	234		2000	

StoreID	Size	From	To
001	250	98	99
002	450	00	-

StoreID	TimeID	…	ItemsSold	…
001	234		2000	
002	456		2500	

StoreID	Size	From	To
001	250	98	99
002	450	00	-

Changing Dimensions XIII

- **Solution 2.2**: examples
- Product descriptions are versioned, when products are changed, e.g., new package sizes.
- New facts can refer to both the newest and older versions of products, as old versions are still in the stores.
- Thus, the Time value for a fact should not necessarily be between the From and To values in the fact's Product dimension row.
- This is unlike changes in Size for a store, where all facts from a certain point in time will refer to the newest Size value.
- This is also unlike alternative categorizations that one wants to choose between.

Changing Dimensions XIV

- Handling “rapidly changing dimensions”.
  - Difference between “slowly” and “rapidly” is subjective.
- Solution 2 is often still feasible.
  - The problem is the size of the dimension.
- Example
  - Assume an Employee dimension with 100,000 employess, each using 2K and many changes every year.
  - Kimball recommends Solution 2.2.
- Other typical examples of (large) dimensions with many changes are Product and Customer.
- Example
  - Some Customer dimensions can have 10M customers.
  - Use Solution 2 and suitable indexing!

Changing Dimensions XV

- Handling “rapidly changing monster dimensions”.
- The more attributes in a dimension table, the more changes per row can be expected.

- Solution 2 yields a dimension that is too large.

- Example
  - A Customer dimension with 100M customers and many attributes.

Changing Dimensions XVI

CustID
Name
PostalAddress
Gender
DateofBirth
Customerside
…
NoKids
MaritialStatus
CreditScore
BuyingStatus
Income
Education
…

CustID
Name
PostalAddress
Gender
DateofBirth
Customerside
…

DemographyID
NoKids
MaritialStatus
CreditScoreGroup
BuyingStatusGroup
IncomeGroup
EducationGroup
…

Changing Dimensions XVII

- Solution
  - Make a ”minidimension” with the often-changing (demograhic) attributes.
  - Convert (numeric) attributes with many possible values into attributes with few possible values, representing groups of the original values.
  - Insert rows for all combinations of values from these new domains.
    - With 6 attributes with 10 possible values each, the dimension gets 1,000,000 rows.
    - Alternatively, (combination) rows can be inserted when needed.
  - If the minidimension is too large, it can be split into two or more minidimensions.
    - Here, synchronous attributes must be considered (and placed in the same minidimension).
    - The same attribute can be repeated in another minidimension.

Changing Dimensions XVIII

- Pros
  - DW size (dimension tables) is kept down.
  - Changes in a customer's demographic values do not result in changes in dimensions.
    - With the alternative solution, rows must be inserted into the minidimension.
- Cons
  - More dimensions and more keys in the star schema.
  - Using value groups gives less detail.
  - The construction of groups is irreversible and makes it hard to make other groupings.
  - Navigation of customer attributes is more cumbersome as these are in more than one dimension.
    - An ActualDemography attribute can be added to the dimension with the stable values.

Changing dimensions - Summary

- Multidimensional models realized as star schemas support change over time to a large extent.
- This is important!
  - Applications change.
  - The modeled reality changes.
- A number of techniques for handling change over time at the instance level was described.
  - Solution 2 (and the derived, 2.1 og 2.2) is the most useful.
  - It is possible to capture change precisely.