12. Fundamentals of Data Warehousing

→ Introduction

- Data warehouse
  - Architecture
  - Model
  - Data Marts
  - Query
  - OLAP
- Providing Semantics to Data Warehouse
- Summary

New Business Environment

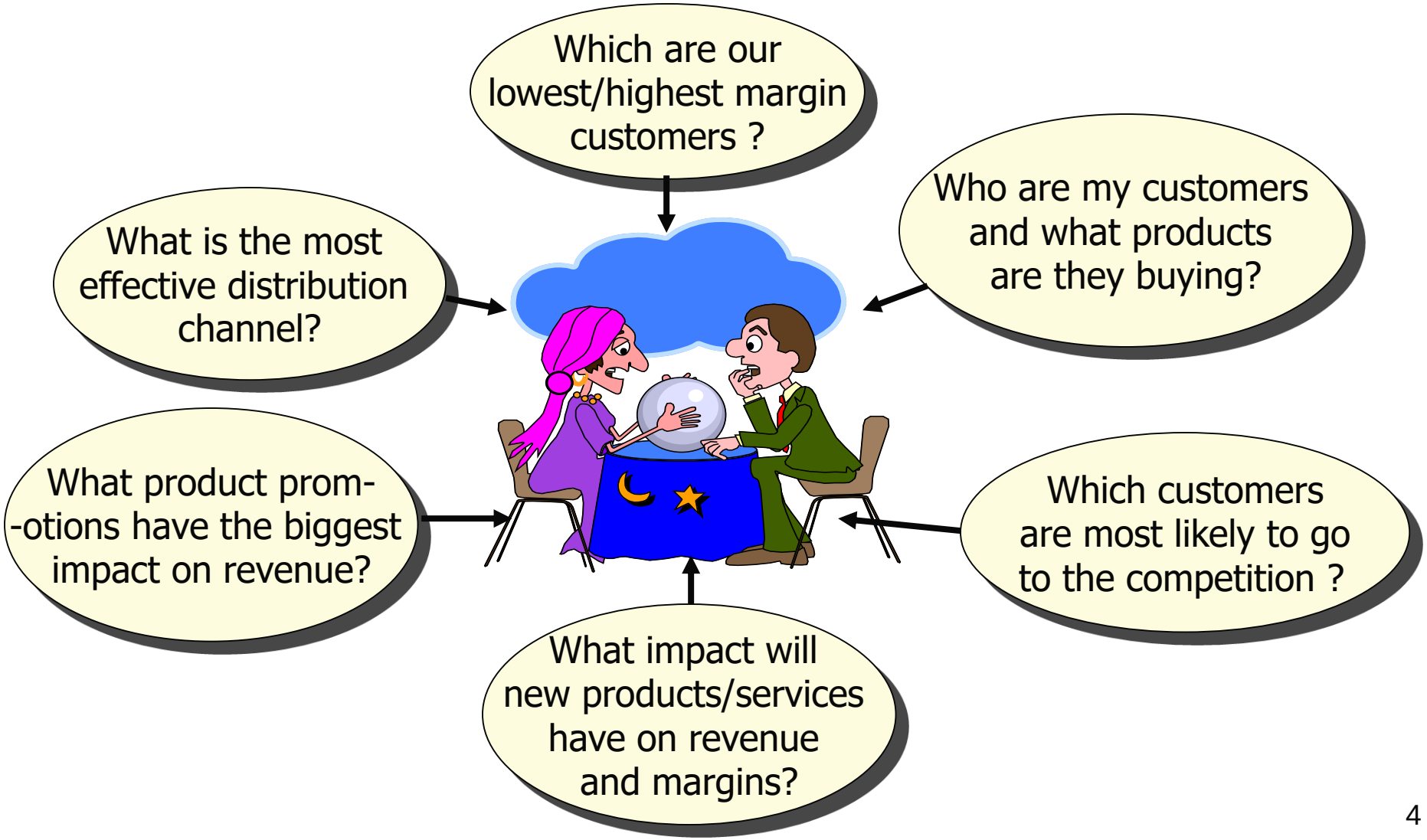
- Economic crisis
  - importance of market & credit risk management for corporations
- Deregulation
  - intensifying competition
  - heightened interest in retaining & acquiring customers
- Merger & Acquisition
  - need for consolidated view of business
  - leveraging big data within large corporations in decision making
- E-business
  - easy way of reaching customers

Changes in Business Strategies

- Old-style one focuses on
  - reducing costs
  - improving product penetration (finding a customer for a product not vice versa)

- New-style aggressive one focuses on
  - getting closer to customers
  - finding new ways to increase revenue from customers
  - satisfaction → loyalty → more customers → revenue

A producer wants to know….

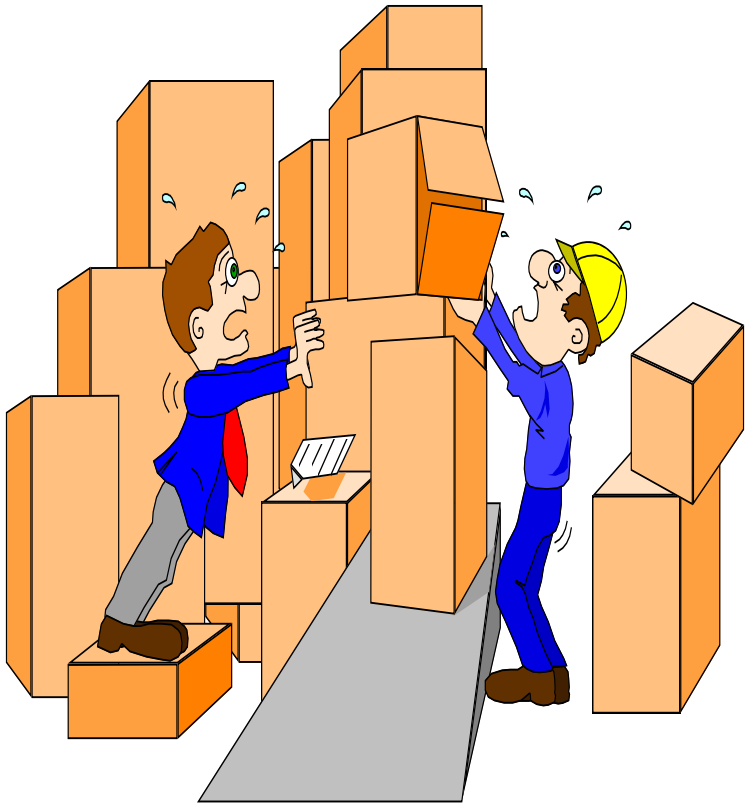


Data, data everywhere. Yet ...

- I can't find the data I need
  - data is scattered over the network
  - many versions, subtle differences
- I can't get the data I need
  - need an expert to get the data
- I can't understand the data I found
  - available data poorly documented
- I can't use the data I found
  - results are unexpected
  - data needs to be transformed from one form to other

What are the users saying...

- Data should be integrated across the enterprise
- Summary data has a real value to the organization
- Historical data holds the key to understanding data over time
- What-if capabilities are required



Decision Support

- Used to manage business
- Data is historical or point-in-time
- Optimized for inquiry rather than update
- Use of the system is loosely defined and can be ad-hoc
- Used by managers and end-users to understand the business and make judgements

How does IT play its role?

- Target
  - Bring together large amounts of customer-related data
  - Gain a better understanding of the customer behavior

  → enable management to make quick and competitive business decisions.

- However,
  - in most organizations, data about specific parts of business is there - lots and lots of data, somewhere, in some form.
  - data is available but *not information* -- and *not the right information at the right time*.

Solution - warehousing business data

- Integrate data from multiple different sources to provide a **single, complete,** and **consistent** store, and make it available to end users in a way they can understand and use in a business context.

- To do more - mining business data
  - Discover knowledge from business data
  - Turn the huge volume of data into a mine of gold/diamond

Decision Support - DSS

Computers are used to process data and to provide information to support decision making. - *Dr. P. Gray*

- 1970's: Decision Support Systems (DSS)
  - Information systems that use models and data to help managers solve managerial problems
    - from budget decisions using simple speadsheets to optimal site location using scientific programming.
  - Problem: **managers must be able to create and operate the systems.**

Decision Support - EIS

- 1980's: Executive Information Systems (EIS)
  - Information systems that include a wide abroad of information about firms and external environment.
  - Problem: **managers can only know from their EISs what is going on, and no analytical capabilities provided by EISs.**
- Both EIS and DDS lacked a strong database component
  - **Have to create their own databases, which is demanding and time consuming.**

Decision Support - RDBMS

- 1980's: Relational Database Management Systems (RDBMS)
  - Mainly On-Line Transaction Processing (OLTP) systems
    - To automate day-to-day clerical data processing tasks such as order entry and banking transactions.
  - Problem: limited On-Line Analytical Processing (OLAP) capabilities.
    - Processing complex analysis queries takes too long time, and often ties up the systems so that they cannot perform the intended transactions.

OLAP & Decision Support Queries

- Decision support queries typically consist of
  - *Long* and often *complex read-only* queries that access large portions of the database (Databases for decision support)
    - " *What were the sales volumes by region and product category last year ?* "
    - " *How did the share price of computer manufacturers correlate with quarterly profits over the last 10 years ?* "
    - *"Which of my customers are most likely to go to the competition?* "
    - *"What product promotions have the biggest impact on revenue?* "

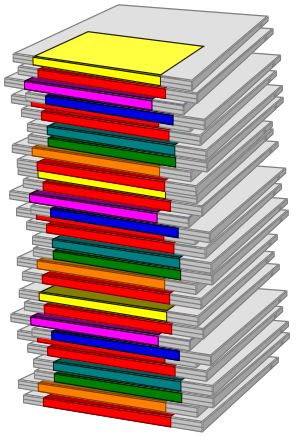
Help knowledge workers make faster and better decisions!

OLTP vs. OLAP

	OLTP	OLAP
user	clerk, IT professional	knowledge worker
function	day to day operations	decision support
data	current, up-to-date, detailed, flat relational isolated	historical, summarized, multidimensional, integrated, consolidated
usage	repetitive	ad-hoc
access	read/write	lots of scans
unit of work	short, simple transaction	complex query
# records accessed	tens	millions
# users	thousands	hundreds
database size	100MB-GB	100GB-TB-EB
metrics	transaction throughput	query throughput, response

Decision Support - DWS

- 1990's: Data Warehousing Systems (DWS)
  - storing huge amounts of data
    - need not contain real-time or up-to-the minute information, as decision support applications tend to process large amounts of data which would not be affected significantly by individual transactions.
  - transforming data into information and making it available to users in a timely enough manner to make a difference



Data Warehouse

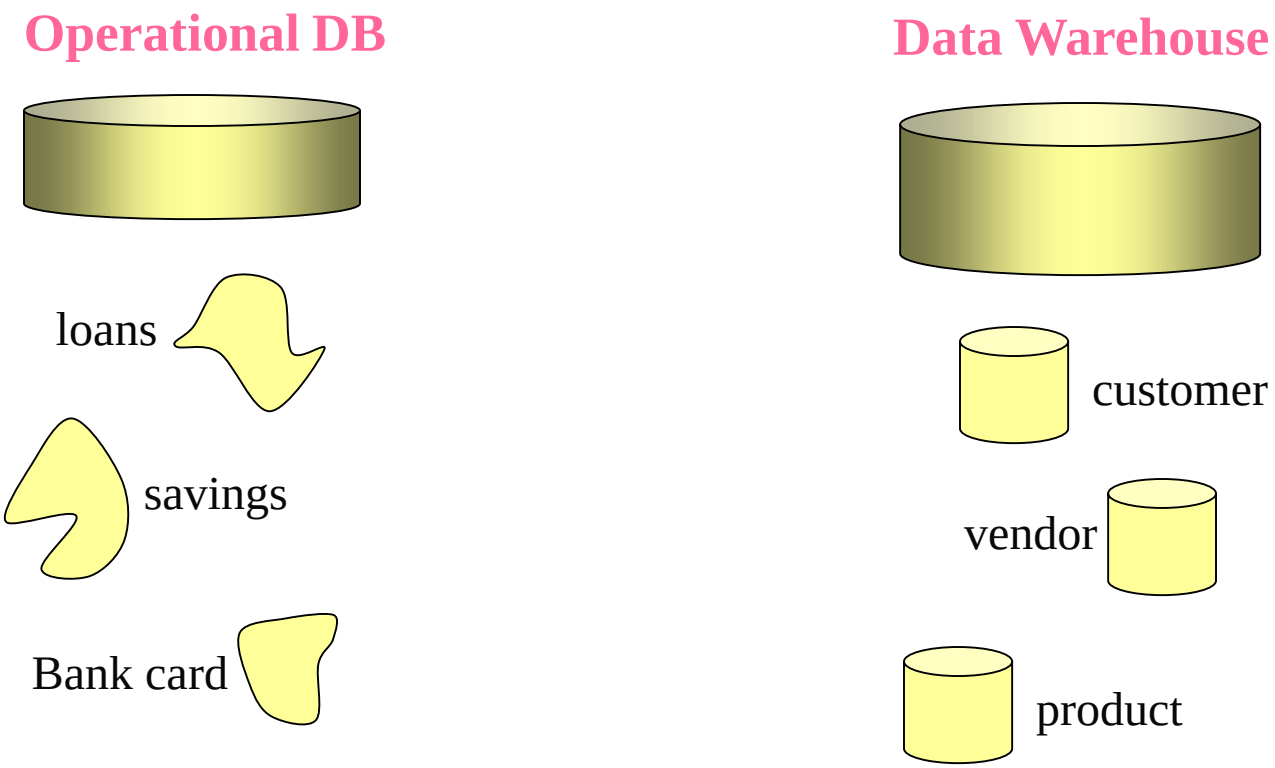
- A decision support database
  - used primarily in organizational decision making.
  - integrate large amounts of extracted and summarized (pre-aggregated) data from multiple sources for *direct* querying and analysis.

Terabytes -- 10^{12} bytes
Petabytes -- 10^{15} bytes
Exabytes -- 10^{18} bytes
Zettabytes -- 10^{21} bytes
Zottabytes -- 10^{24} bytes

- A data warehouse is subject-oriented, integrated, time-varying, and non-volatile.

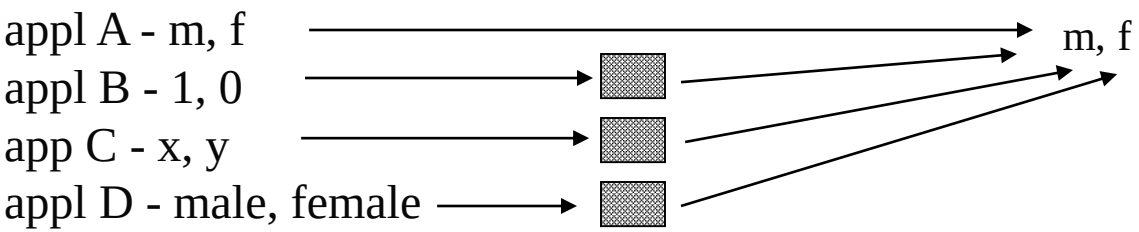
Data Warehouse: Subject-Oriented

- A data warehouse has a strong subject orientation
  - focus on data modeling and database design exclusively (i.e., process design is not part of the data warehouse environment)



Data Warehouse: Integrated

- There is no consistency among different data sources.
- When data is moved to the warehouse, it is converted and integrated.



Data Warehouse: Time Variant

- The time horizon for the data warehouse is significantly longer than that for the operational systems.

- The key structure of the data warehouse always contains some element of time; while the key structure of the operational data may or may not.

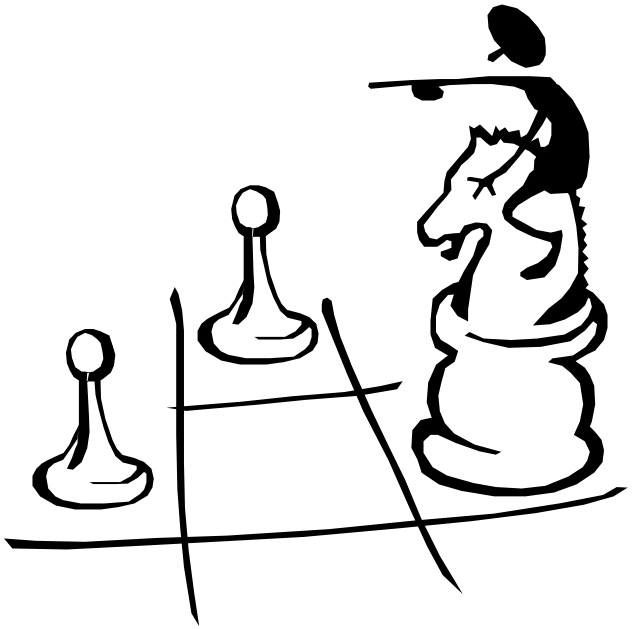
Data Warehouse: Non-Volatile

- There are only two kinds of operations that occur in the data warehouse
  - data loading and data access

- No need to be cautious of the update anomaly in the data warehouse, since update of data is not done.

Data Mining works with Warehouse Data

- Data Warehousing provides the Enterprise with a memory



- Data Mining provides the enterprise with intelligence

We want to know more, such as

- "Given a database of 100,000 names, which people are the least likely to default on their credit cards?"
- "Which types of transactions are likely to be fraudulent given the demographics and transactional history of a particular customer?"
- "If I raise the price of my product by 2%, what is the effect on my Return On Investment (ROI) ?"
- "If I offer only 2,500 airline miles as an incentive to purchase rather than 5,000, how many lost responses will result?"
- "If I emphasize ease-of-use of the product as opposed to its technical capabilities, what will be the net effect on my revenues?"
- "Which of my customers are likely to be the most loyal?"

Data Mining helps extract such information!

Data Mining in Use

- Governments use Data Mining to track fraud
- A Supermarket becomes an information broker
- Basketball teams use it to track game strategy
- Cross Selling
- Warranty Claims Routing
- Holding on to Good Customers
- Weeding out Bad Customers

Application Areas

Industry	Application
Finance	Credit Card Analysis
Insurance	Claims, Fraud Analysis
Telecommunication	Call record analysis
Transport	Logistics management
Consumer goods	Promotion analysis
Data Service providers	Value added data
Utilities	Power usage analysis

What makes data mining possible?

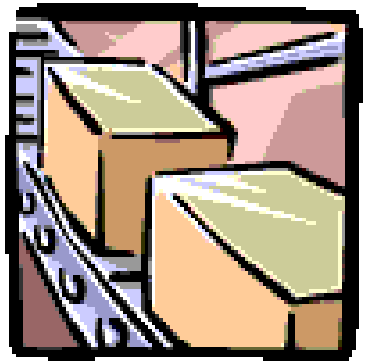
- Advances in the following areas are making data mining deployable
  - data warehousing
  - better and more data (i.e., operational, behavioral, demographic)
  - the emergence of easily deployed data mining tools
  - the advent of new data mining techniques

Why Separate Data Warehousing?

- Performance
  - Operational DBs designed & tuned for known tasks and workloads.
  - Complex OLAP queries would degrade performance for operational tasks.
  - Special data organization, access & implementation methods needed for multidimensional views & queries.
- Function
  - Missing data: Decision support requires historical data, which operational DBs do not typically maintain.
  - Data consolidation: Decision support requires consolidation (aggregation, summarization) of data from many heterogeneous sources: operational DBs and external sources.
  - Data quality: Different sources typically use inconsistent data representations, codes, and formats which have to be reconciled.

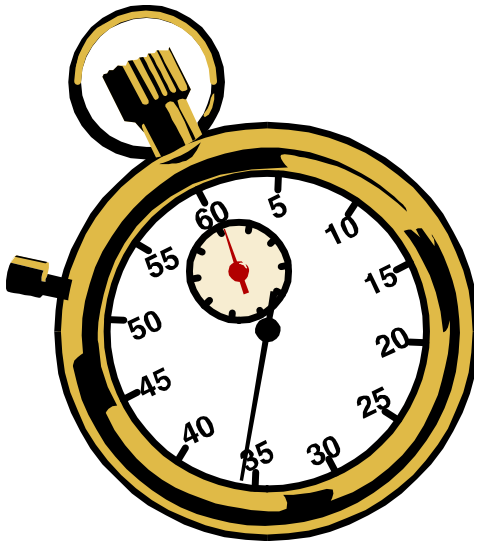
What are Operational Systems?

- They are OLTP systems
- Run mission-critical applications
- Need to work with stringent performance requirements for routine tasks
- Used to run a business!



Operational Systems

- Run business in real time
- Based on up-to-the-second data
- Optimized to handle large numbers of simple read/write transactions
- Optimized for fast response to predefined transactions
- Used by people who deal with customers, products (clerks, salespeople etc.)
- They are increasingly used by customers



Examples of Operational Data

Data	Industry	Usage	Technology	Volumes
Customer File	All	Track Customer Details	Legacy applications, flat files, mainframe	Small-medium
Account Balance	Finance	Control account activities	Legacy applications, hierarchical databases, mainframe	Large
Point-of-Sale data	Retail	Generate bills, manage stock	ERP, Client/Server, relational databases	Very Large
Call Record	Telecomm-unications	Billing	Legacy application, hierarchical database, mainframe	Very Large
Production Record	Manufact-uring	Control Production	ERP, relational databases, AS/400	Medium

RDBMS used for OLTP

- Database Systems have been used traditionally for OLTP
  - clerical data processing tasks
  - detailed, up-to-date data
  - structured repetitive tasks
  - read/update a few records
  - isolation, recovery, and integrity are critical

OLTP vs. Data Warehouse

- OLTP systems are tuned for known transactions and workloads, while workload is not known a priori in a data warehouse
- Special data organization, access methods, and implementation methods are needed to support data warehouse queries (typically multidimensional queries)
  - e.g., *average amount spent on phone calls between 9 AM-5PM in HK during the month of December*

OLTP vs Data Warehouse (*cont.*)

- OLTP
  - Application Oriented
  - Used to run business
  - Detailed data
  - Current up-to-date
  - Isolated Data
  - Repetitive access
  - Clerical User
- Data Warehouse (DSS)
  - Subject Oriented
  - Used to analyze business
  - Summarized and refined
  - Snapshot data
  - Integrated Data
  - Ad-hoc access
  - Knowledge User (Manager)

OLTP vs Data Warehouse (*cont.*)

- OLTP
  - Performance Sensitive
  - Few records accessed at a time
  - Read/Update Access
  - No data redundancy
  - Database Size:100MB -100 GB
- Data Warehouse
  - Performance relaxed
  - Large volumes accessed at a time
  - Mostly Read (Batch Update)
  - Redundancy present
  - Database Size: 100 GB -

OLTP vs Data Warehouse (*cont.*)

- OLTP
  - Transaction throughput is the performance metric
  - Thousands of users
  - Managed in entirety
- Data Warehouse
  - Query throughput is the performance metric
  - Hundreds of users
  - Managed by subsets

To summarize ...

- OLTP Systems are used to *"run"* a business

- The Data Warehouse helps to *"optimize"* the business

Why Now?

- Data is being produced
- ERP provides clean data
- The computing power is available
- The computing power is affordable
- The competitive pressures are strong
- Commercial products are available

Old Retail Paradigm

- Wal*Mart
  - Inventory Management
  - Merchandise Accounts Payable
  - Purchasing
  - Supplier Promotions: National, Region, Store Level
- Suppliers
  - Accept Orders
  - Promote Products
  - Provide special Incentives
  - Monitor and Track The Incentives
  - Bill and Collect Receivables
  - Estimate Retailer Demands

New (Just-In-Time) Retail Paradigm

- No more deals
- Shelf-Pass Through
  - One Unit Price
    - Suppliers paid once a week on ACTUAL items sold
  - Wal*Mart Manager
    - Daily Inventory Restock
    - Suppliers (sometimes SameDay) ship to Wal*Mart
- Warehouse-Pass Through
  - Stock some Large Items
    - Delivery may come from supplier
  - Distribution Center
    - Supplier's merchandise unloaded directly onto Wal*Mart Trucks

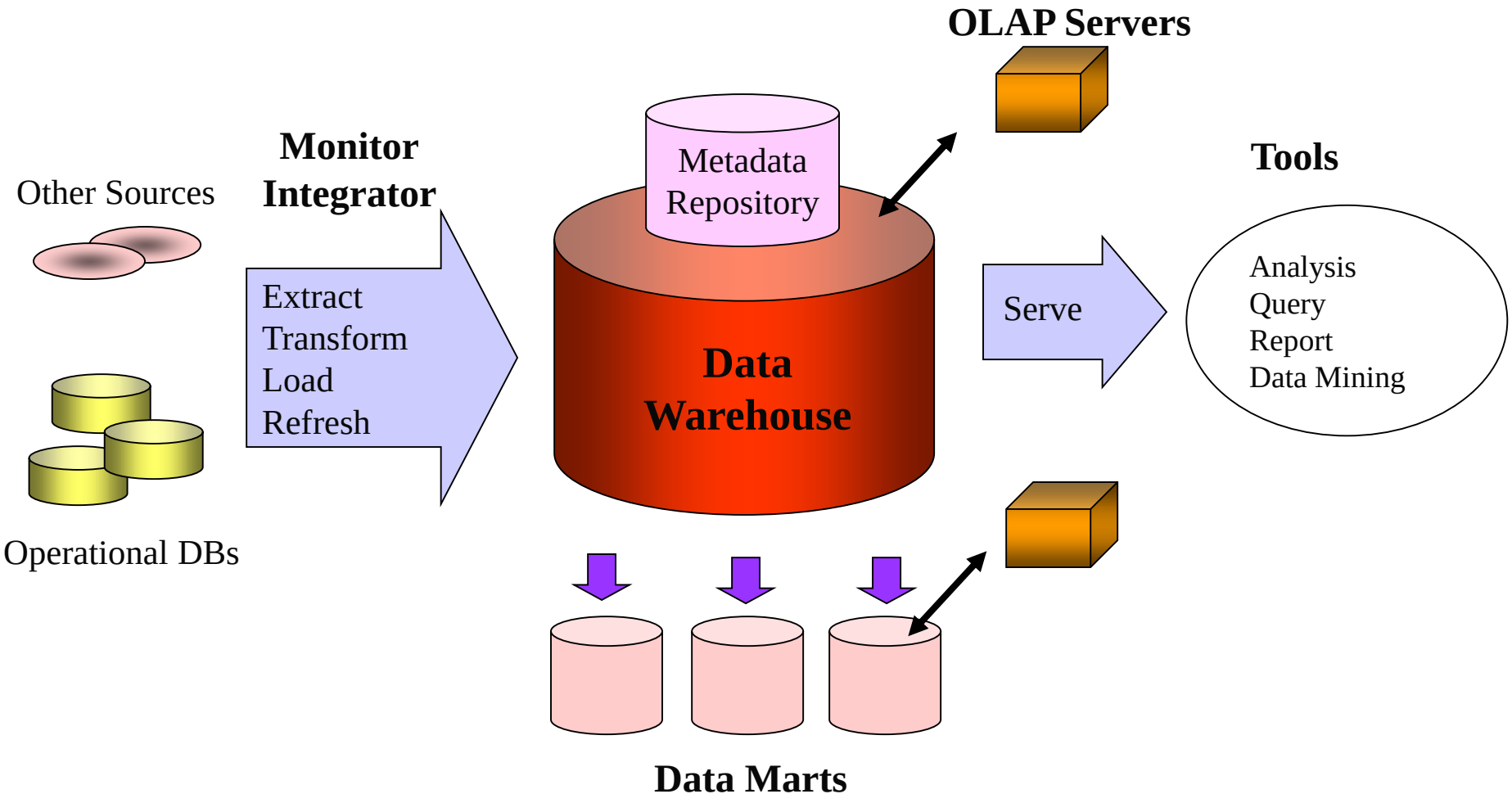
12. Fundamentals of Data Warehousing

- Introduction
- → Data warehouse

  - Architecture
  - Model
  - data marts
  - Querying
  - OLAP

- Providing Semantics to Data Warehouses
- Summary

The Reference Architecture



Back End Tools and Utilities

- Data Extraction
  - extract *foreign* sources via gateways, ODBC drivers, or other wrappers.
- Data Cleaning
  - detect and correct data anomalies
- Data Loading
  - check integrity constraints, sort, summarization, aggregation, etc.
- Refreshing
  - propagate updates on source data to the data stored in the warehouse
- Analyzing, Querying and Data Mining Tools
- Metadata

Loading the Warehouse

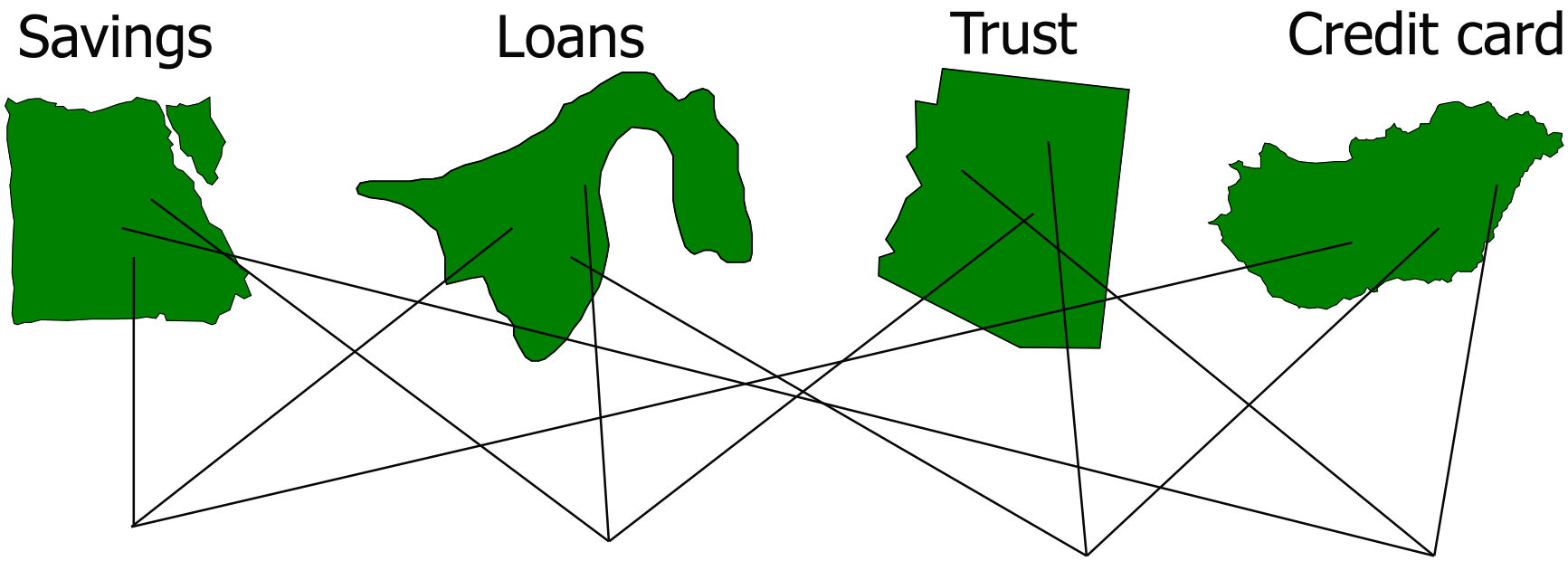
- Source Data
  - Tempting to think creating a data warehouse is simply extracting operational data and entering into a data warehouse
  - However, warehouse data comes from disparate questionable sources.

Quality of Source Data

- Legacy systems no longer documented
- Outside sources with questionable quality procedures
- Production systems with no built-in integrity checks and no integration
  - Operational systems are usually designed to solve a specific business problem and are rarely developed to a corporate plan
    - "And get it done quickly, we do not have time to worry about corporate standards..."

Clean the source data before it is loaded!

Data Integration Across Sources



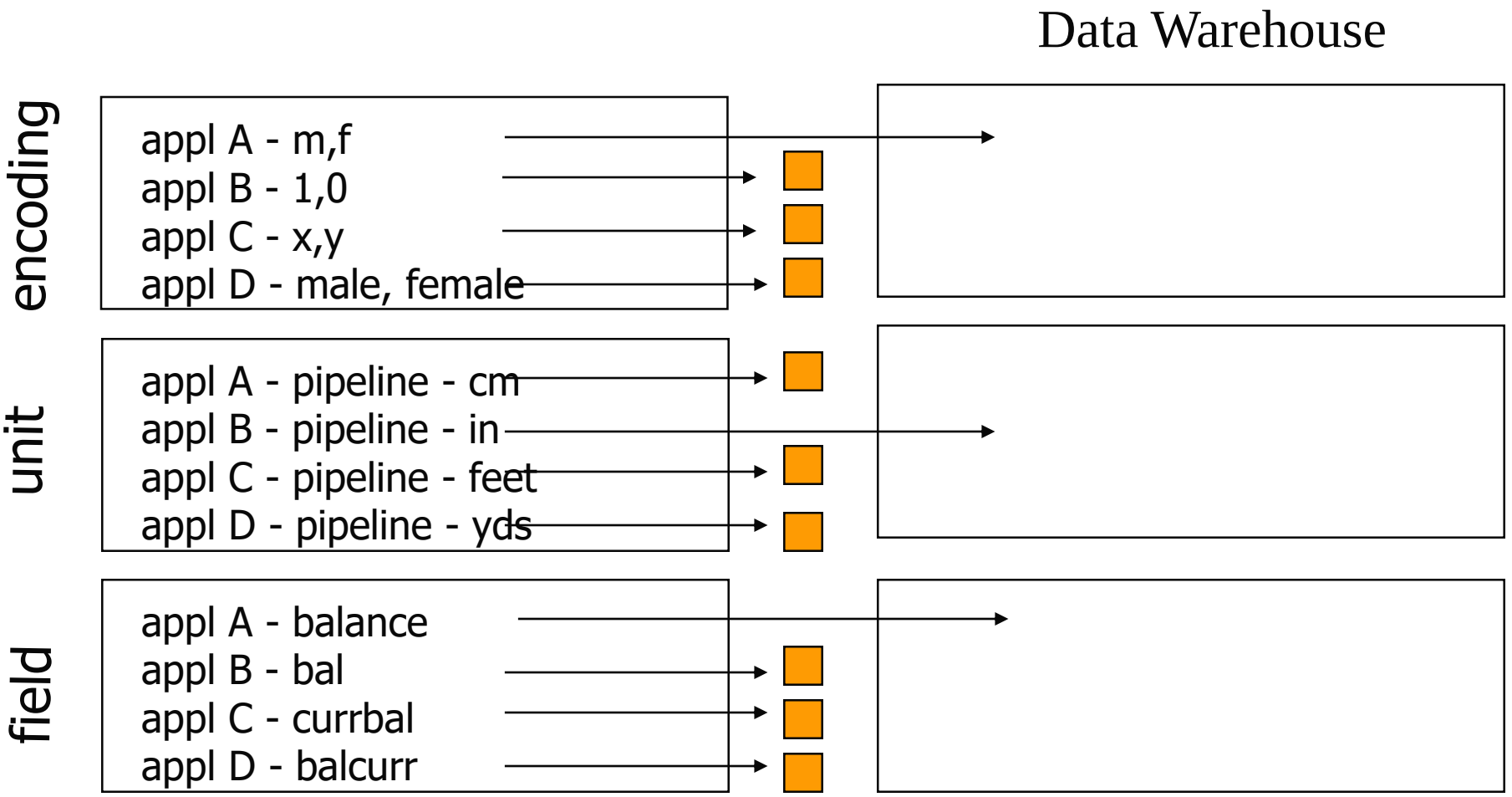
Same data different names

Different data same name

Data found here nowhere else

Different keys same data

Data Transformation Example



Data Integrity Problems

- Same person, different spellings
  - Agarwal, Agrawal, Aggarwal, etc.
- Multiple ways to denote company names
  - Persistent Systems, PSPL, Persistent Pvt. LTD.
- Use of different names
  - mumbai, bombay
- Different account numbers generated by different applications for the same customer
- Required fields left blank
- Invalid product codes collected at point of sale
  - manual entry leads to mistakes
  - “in case of a problem use 9999999”

Data Transformation Terms

- Extracting
- Conditioning
- Scrubbing
- Merging
- Householding
- Enrichment
- Scoring
- Loading
- Validating
- Delta Updating

Extracting Data

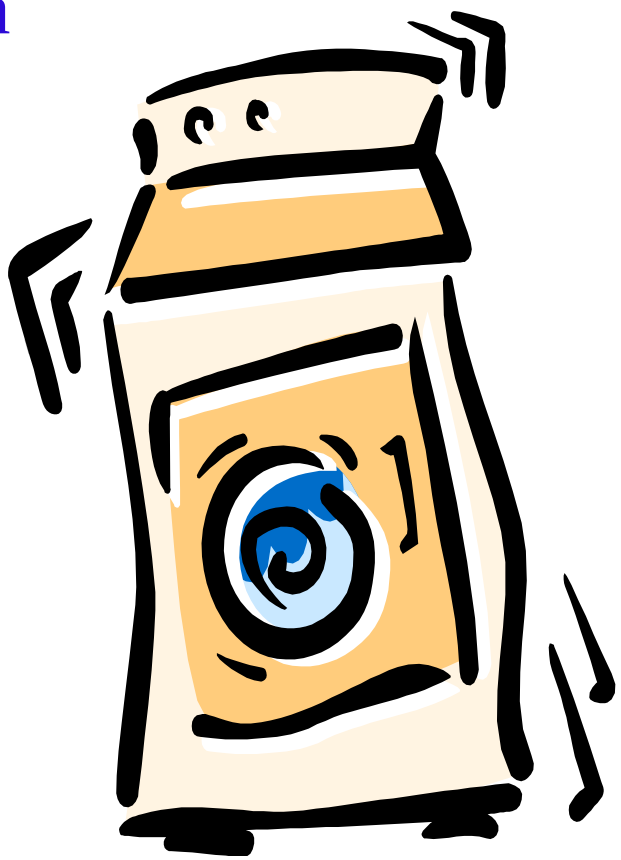
- Extract data from existing operational and legacy data
- Issues:
  - Sources of data for the warehouse
  - Data quality at the sources
  - Merging different data sources
  - Data transformation
  - How to propagate updates (on the sources) to the warehouse
  - Terabytes of data to be loaded

Extracting Data (*cont.*)

- Conditioning
  - The conversion of data types from the source to the target data store (warehouse) -- always a relational database
- Householding
  - Identify all members of a household (living at the same address)
- Enrichment
  - Bring data from external sources to augment/enrich operational data
- Scoring
  - Compute the probability of an event. e.g., chance that a customer is likely to buy a new product

Scrubbing Data

- Sophisticated transformation tools.
- Used for cleaning the quality of data
- Clean data is vital for the success of the warehouse
- Example
  - Seshadri, Sheshadri, Sesadri, Seshadri S., Srinivasan Seshadri, etc. are the same person
- Scrubbing tools
  - Apertus -- Enterprise/Integrator
  - Vality -- IPE
  - Postal Soft



Data Loading

- After extracting, scrubbing, cleaning, validating, etc., we load the data into the warehouse.
- Issues
  - huge volumes of data to be loaded
  - small time window available when warehouse can be taken off-line (usually nights)
  - when to build index and summary tables
  - allow system administrators to monitor, cancel, resume, and change load rates
  - recover gracefully -- restart after failure from where you were and without loss of data integrity

Data Loading (*cont.*)

- Use SQL to append or insert new data
  - record at a time interface
  - will lead to random disk I/O's
- Use batch load utility
- Incremental versus Full loads
- Online versus Offline loads

Data Refreshing

- Propagate updates on source data to the warehouse
- Issues:
  - **when to refresh?**
    - periodically (e.g., every night, every week) or after significant events
    - on every update: not warranted unless warehouse data require current data (up to the minute stock quotes)
    - refresh policy set by administrator based on user needs and traffic
    - possibly different policies for different sources
  - **how to refresh?** -- refresh techniques

    Full extract from base tables
    - read entire source table: too expensive
    - maybe the only choice for legacy systems

Detecting Changes

- Create a snapshot log table to record ids of updated rows of source data and timestamp
- Detect changes by:
  - defining after row triggers to update snapshot log when source table changes
  - using regular transaction log to detect changes to source data

12. Fundamentals of Data Warehousing

- Introduction
- → Data warehouse
  - Architecture
  - Model
  - Data marts
  - Querying
  - OLAP
- Providing Semantics to Data Warehouses
- Summary

Data -- Heart of the Data Warehouse

- Heart of the data warehouse is the data itself!
- Single version of the truth
- Corporate memory
- Data is organized in a way that represents business -- subject orientation

Data Warehouse Structure

- Subject Orientation
  - customer, product, policy, account, etc...
- A subject may be implemented as a set of related tables, e.g., customer may have five tables
  - base customer (1985-87)
    - custid, from date, to date, name, phone, job
  - base customer (1988-90)
    - custid, from date, to date, name, credit rating, employer
  - customer activity (1986-89) -- monthly summary
  - customer activity detail (1987-89)
    - custid, activity date, amount, clerk id, order no
  - customer activity detail (1990-91)
    - custid, activity date, amount, line item no, order no

Time is part of key of each table.

Data Granularity in Warehouse

- Summarized data stored
  - reduce storage costs
  - reduce CPU usage
  - increase performance since smaller number of records to be processed
  - design around traditional high level reporting needs
  - tradeoff with volume of data to be stored and detailed usage of data

Data Granularity in Warehouse (*cont.*)

- Cannot answer some questions with summarized data
  - *Did Anand call Seshadri last month*? Not possible to answer if total duration of calls by Anand over a month is only maintained and individual call details are not.
- Detailed data too voluminous

- Tradeoff is to have dual level of granularity
  - Store summary data on disks
    - 95% of DSS processing done against this data
  - Store details on tapes
    - 5% of DSS processing against this data

Vertical Partitioning

Acct. No	Name	Balance	Date Opened	Interest Rate	Address

Frequently accessed

Acct. No	Balance

Rarely accessed

Acct. No	Name	Date Opened	Interest Rate	Address

Smaller table, so less I/O!

Derived Data

- Introduction of derived (calculated data) may often help
- Have seen this in the context of dual levels of granularity
- Can keep auxiliary views and indexes to speed up query processing

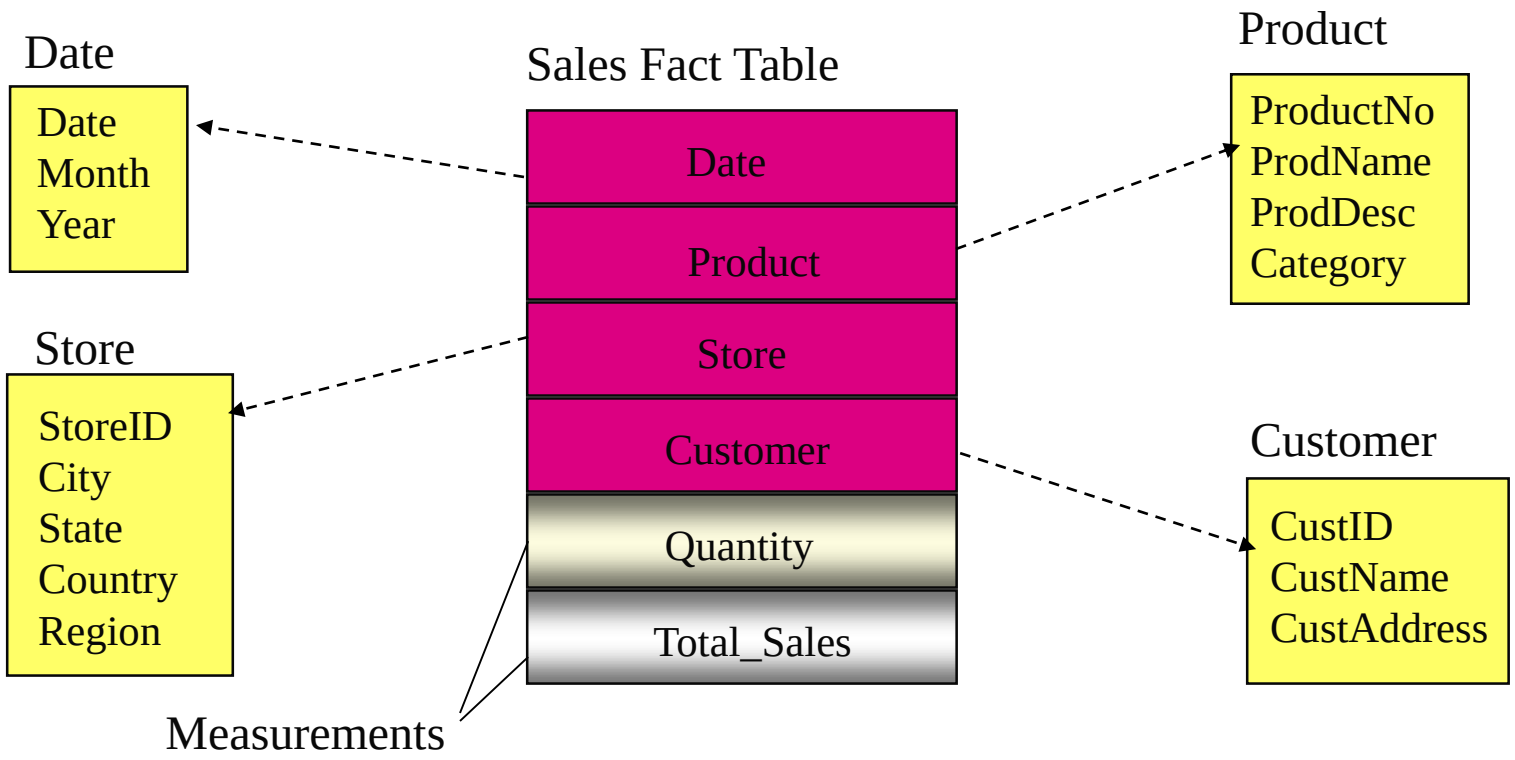
Schema Design

- Database organization
  - must look like business
  - must be recognizable by business users
  - must be approachable by business users
  - must be simple

Fact Table and Dimension Tables

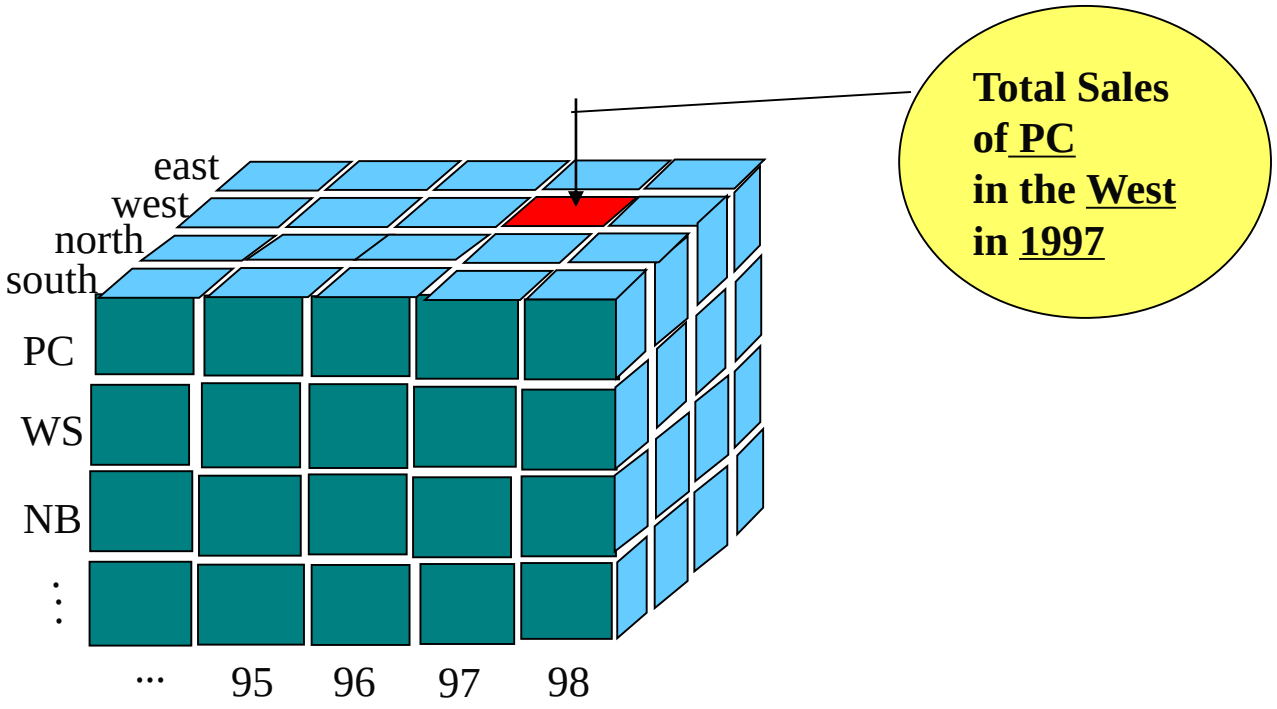
- Fact table
  - mostly raw numeric items
  - narrow rows, a few columns at most
  - large number of rows (millions to billions)
  - access via dimensions
- Dimension tables
  - Define business in terms already familiar to users
  - Wide rows with lots of descriptive text
  - Small number of rows (about a million rows)
  - Joined to fact table by a foreign key
  - heavily indexed
  - typical dimensions
    - time periods, geographic regions (markets, cities), products, customers, salesperson, etc.

- A single fact table in the middle connected to a number of dimension tables.



Multidimensional Data

- Data warehouse may store some selected summary data.
- *Total_Sales* as a function of *product*, *year*, and *region*.

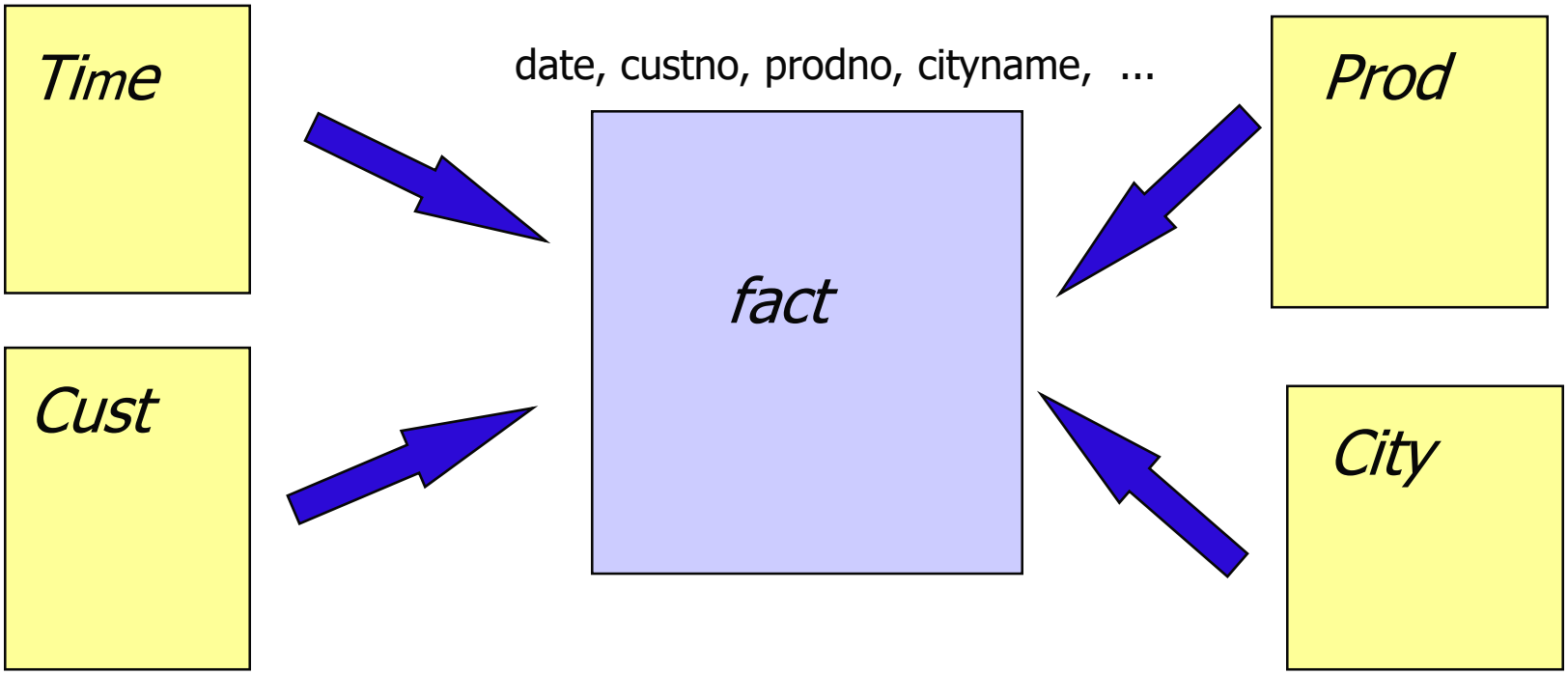


Schema Types

- Star Schema
- Snowflake schema
- Fact Constellation Schema

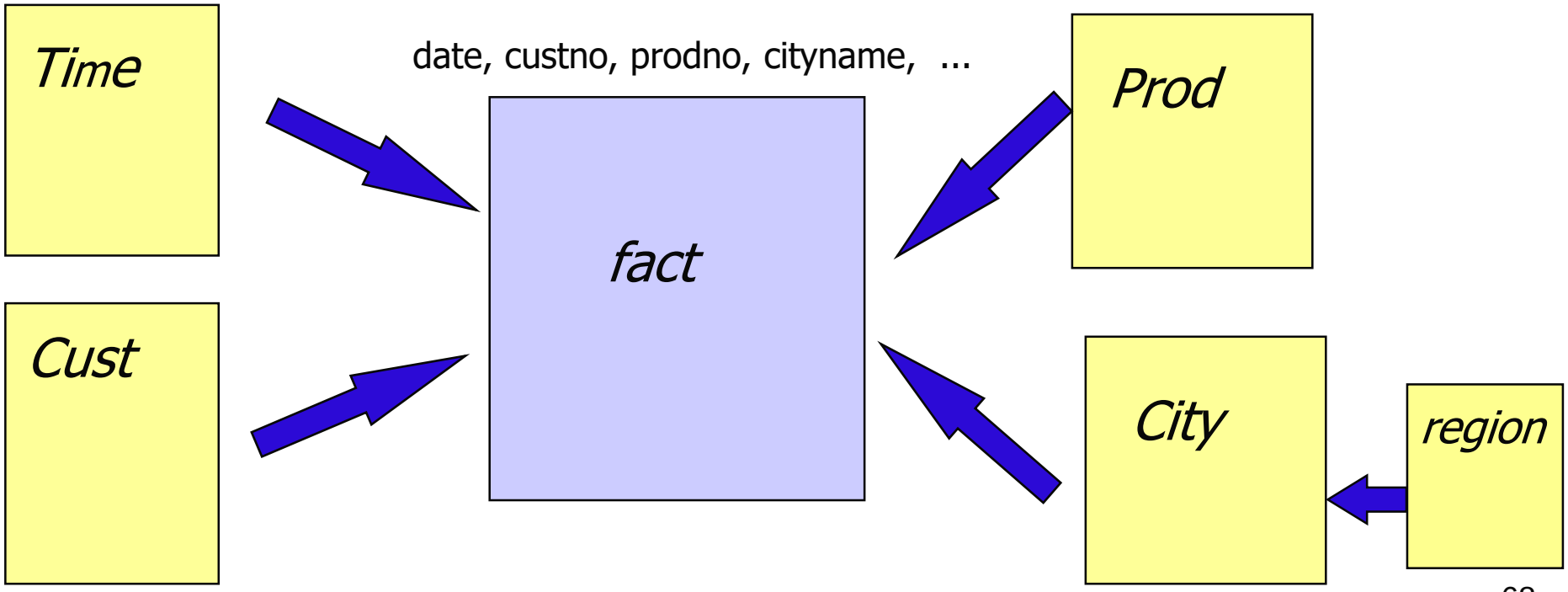
Star Schema

- A single fact table and for each dimension one dimension table
- Does not capture hierarchies directly



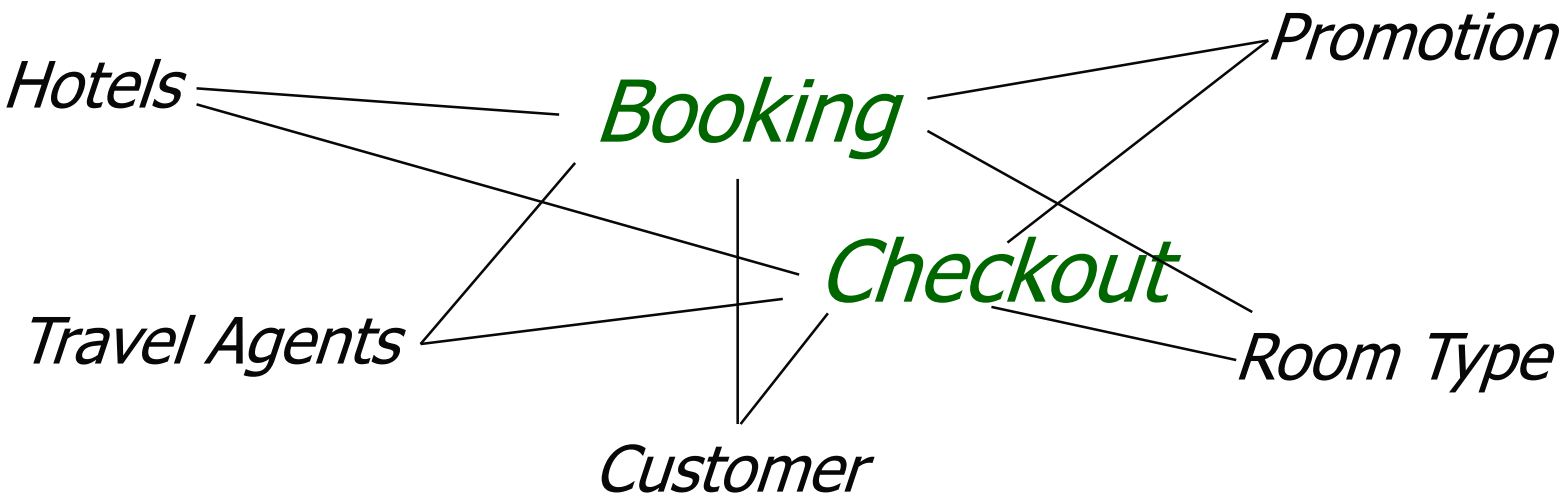
Snowflake schema

- Represent dimensional hierarchy directly by normalizing tables
- Easy to maintain and saves storage



Fact Constellation

- Multiple fact tables that share many dimension tables
  - Booking and Checkout may share many dimension tables in the hotel industry



De-normalization

- Normalization in a data warehouse may lead to lots of small tables
- Can lead to excessive I/O's since many tables have to be accessed
- De-normalization is the answer especially since updates are rare

Creating Arrays

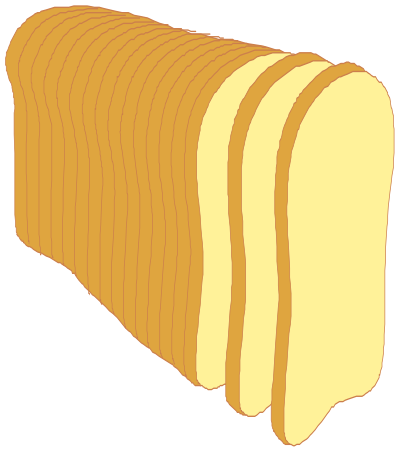
- Many times each occurrence of a sequence of data is in a different physical location
- Beneficial to collect all occurrences together and store as an array in a single row
- Make sense only if there are a stable number of occurrences which are accessed together
- In a data warehouse, such situations arise naturally due to time based orientation
  - can create an array by month

Selective Redundancy

- Description of an item can be stored redundantly with order table -- most often item description is also accessed with order table
- Updates have to be careful

Partitioning

- Breaking data into several physical units that can be handled separately
- Not a question of whether to do it in data warehouses but how to do it
- Granularity and partitioning are key to effective implementation of a warehouse



Why Partition?

- Flexibility in managing data
- Smaller physical units allow
  - easy restructuring
  - free indexing
  - sequential scans if needed
  - easy reorganization
  - easy recovery
  - easy monitoring

Criterion for Partitioning

- Typically partitioned by
  - date
  - line of business
  - geography
  - organizational unit
  - any combination of above

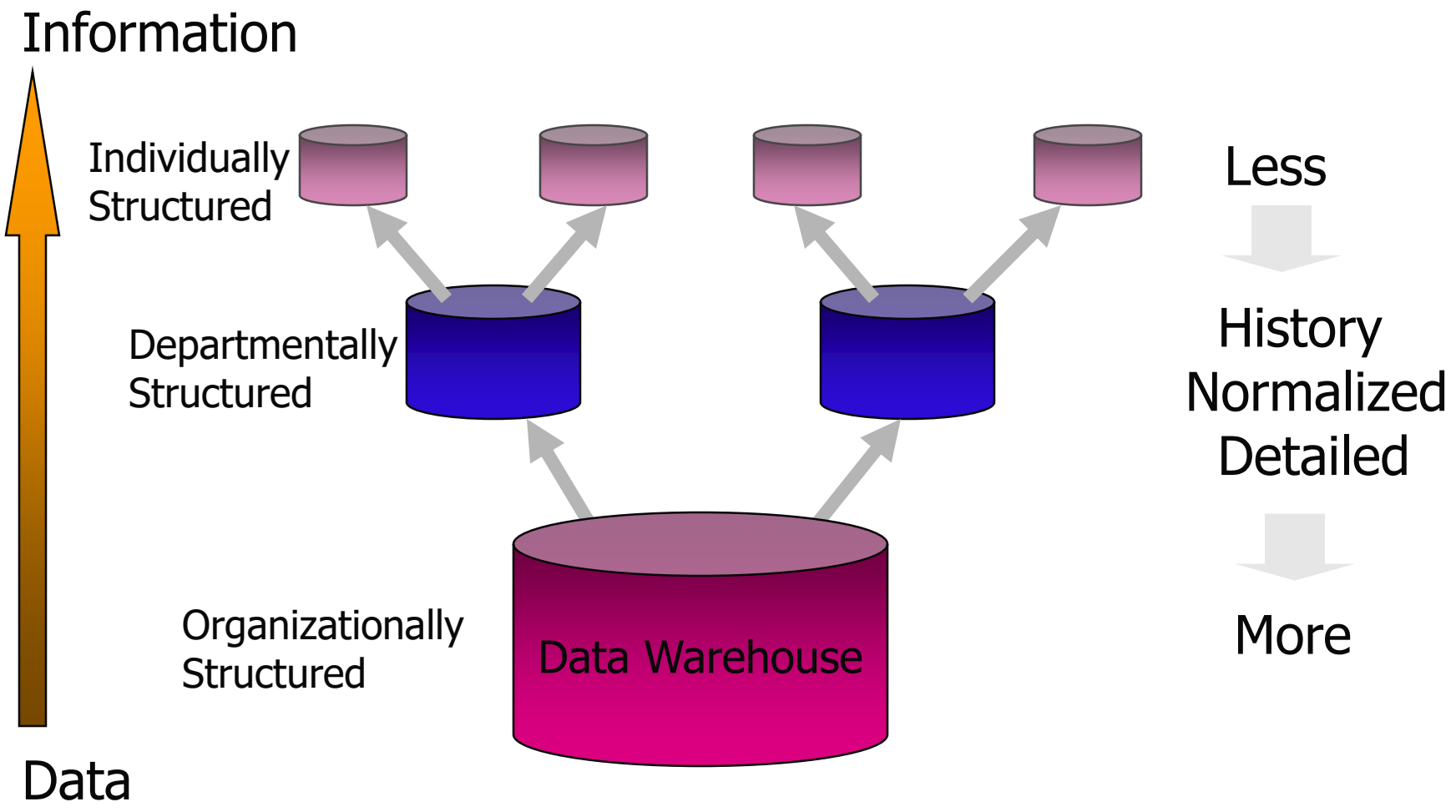
Where to Partition?

- Application level or DBMS level
- Make sense to partition at application level
  - Allow different definitions at different granularities for each year
    - Important since warehouse spans many years and as business evolves, definition changes
  - Allow data to be moved between processing complexes easily

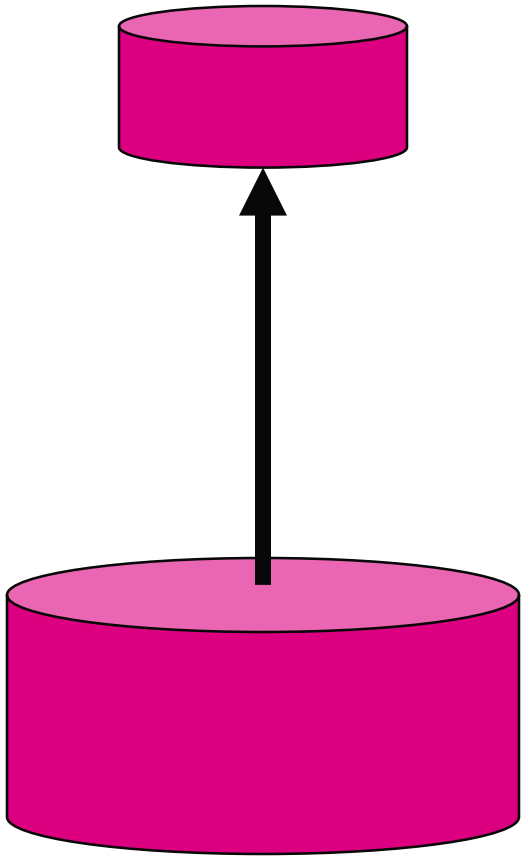
12. Fundamentals of Data Warehousing

- Introduction
- → Data warehouse
  - Architecture
  - Model
  - Data marts
  - Querying
  - OLAP
- Providing Semantics to Data Warehouses
- Summary

From Data Warehouse to Data Marts



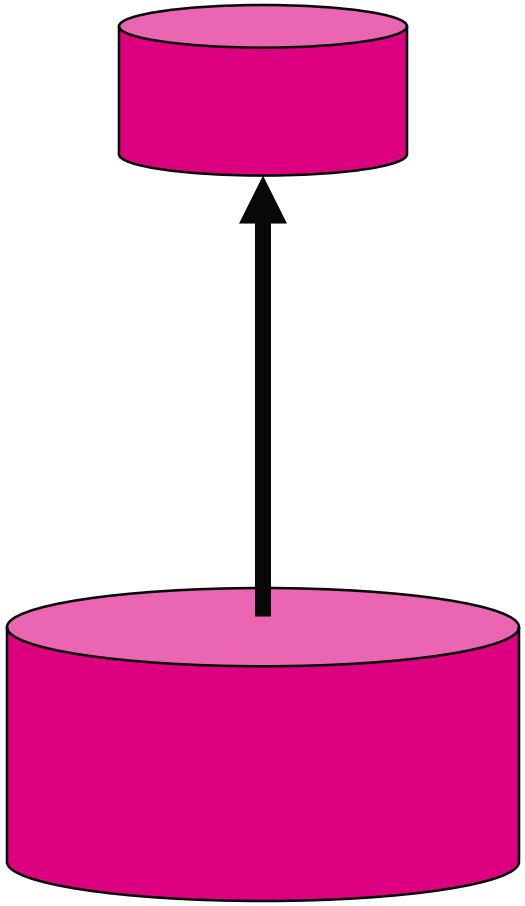
Data Warehouse and Data Marts



OLAP
Data Mart
Lightly summarized
Departmentally structured

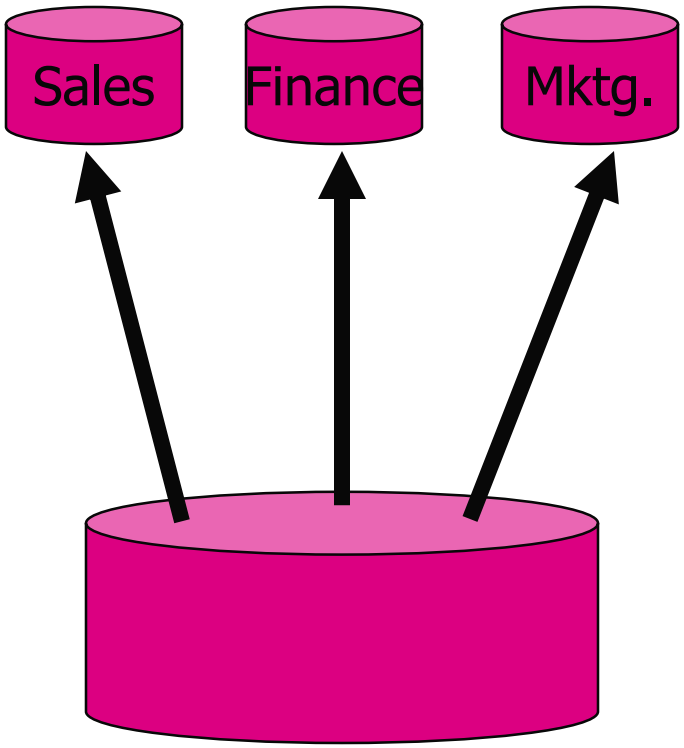
Organizationally structured
Atomic
Detailed Data Warehouse Data

Characteristics of Departmental Data Marts



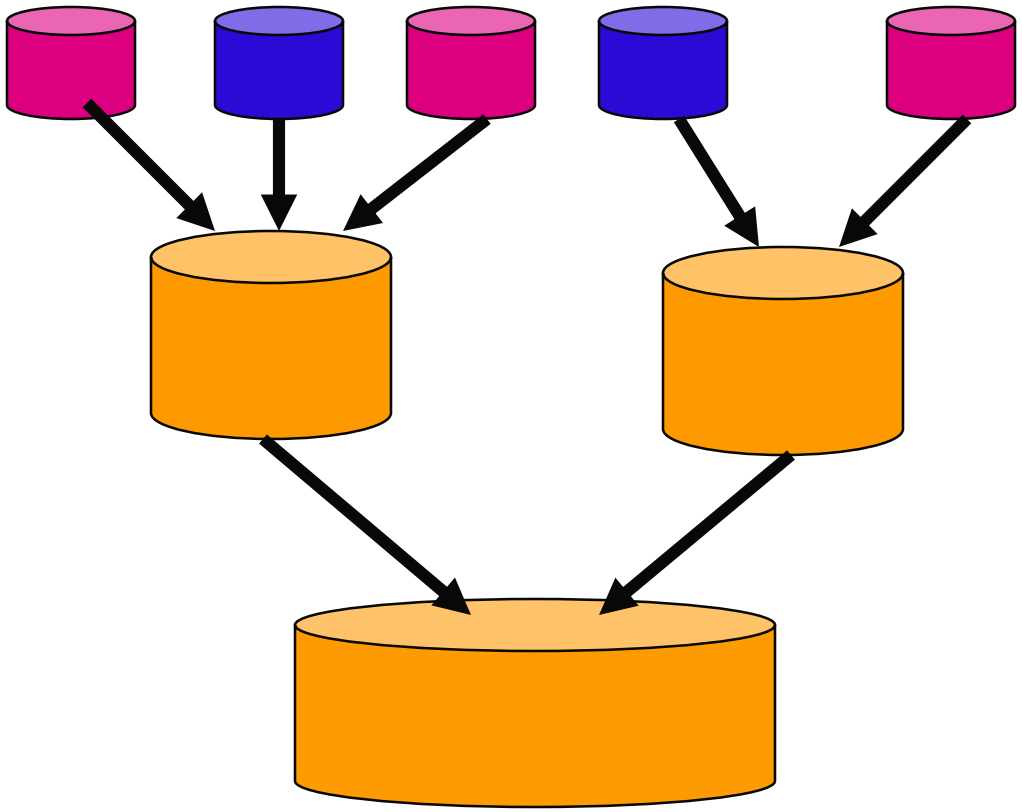
- OLAP
- Small
- Flexible
- Customized by department
- Source is departmentally structured data warehouse

Creating Departmental Data Marts

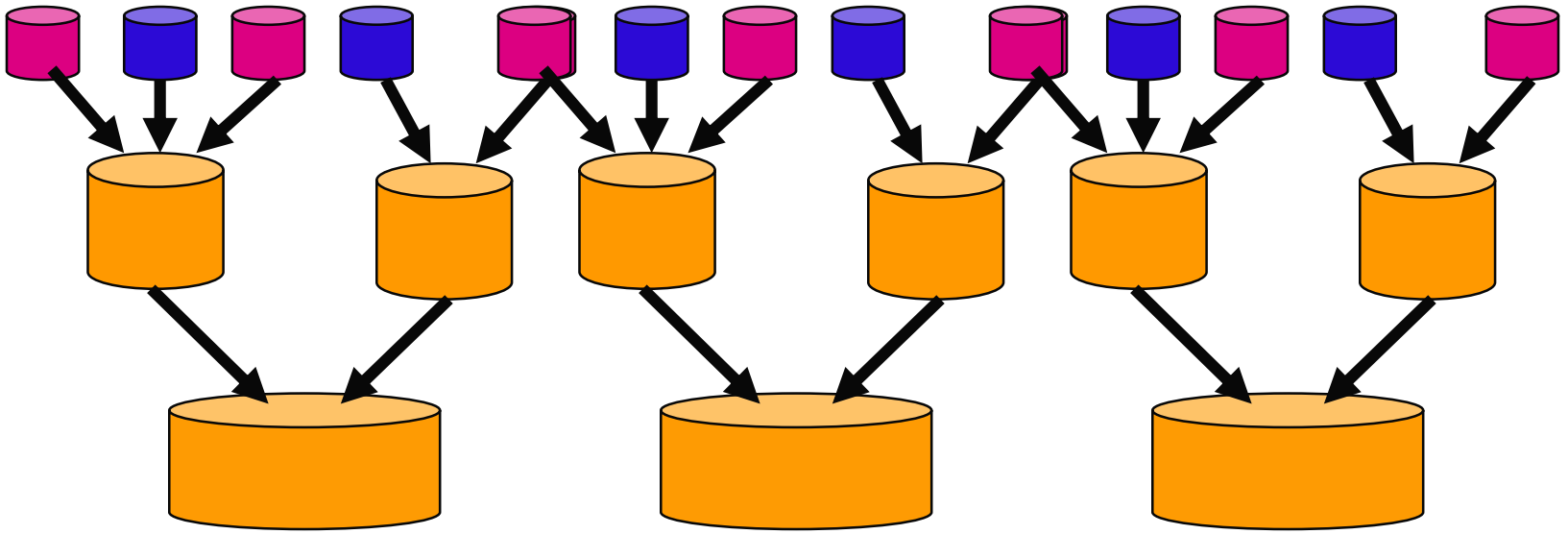


- OLAP
- Subset
- Summarize
- Superset
- Index
- Array

Data Mart Centric Solution

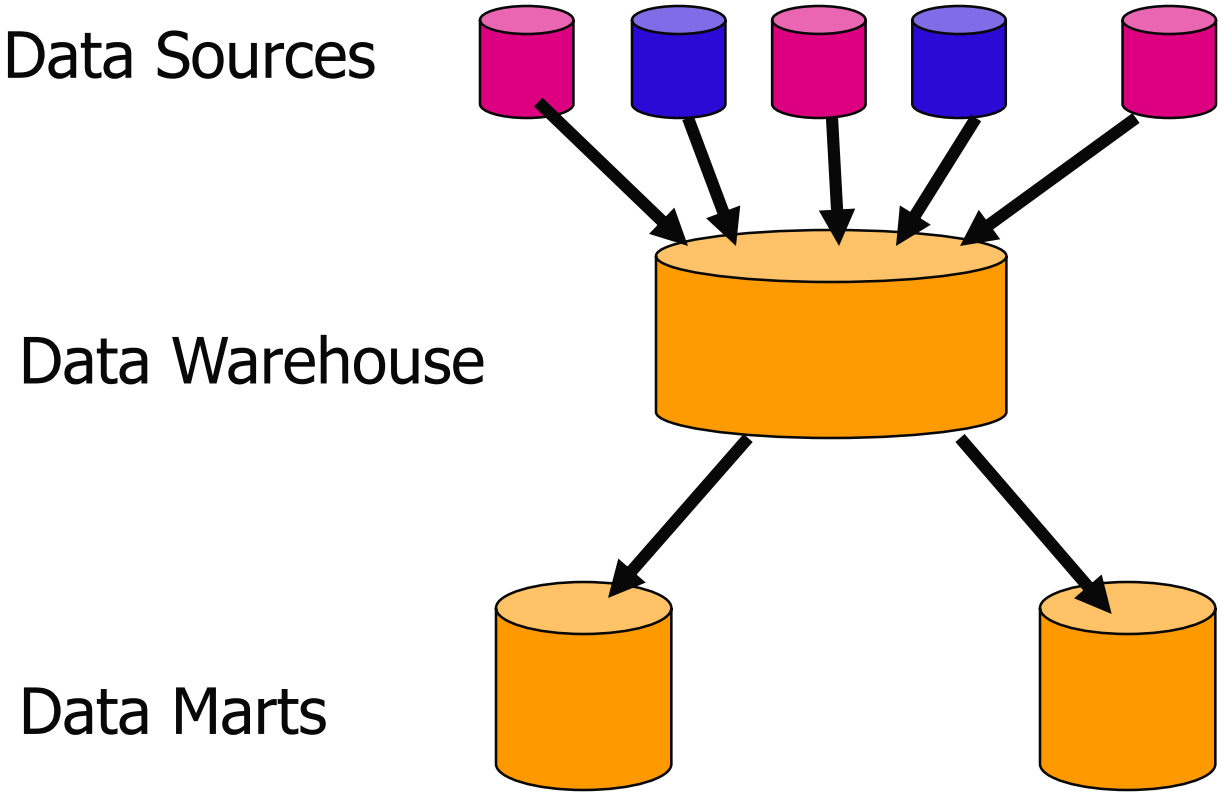


Problems with Data Mart Centric Solution



If creating multiple warehouses, integrating them is a problem

True Warehouse Solution



12. Fundamentals of Data Warehousing

- Introduction
- → Data warehouse
  - Architecture
  - Model
  - Data marts
  - Querying
  - OLAP
- Providing Semantics to Data Warehouses
- Summary

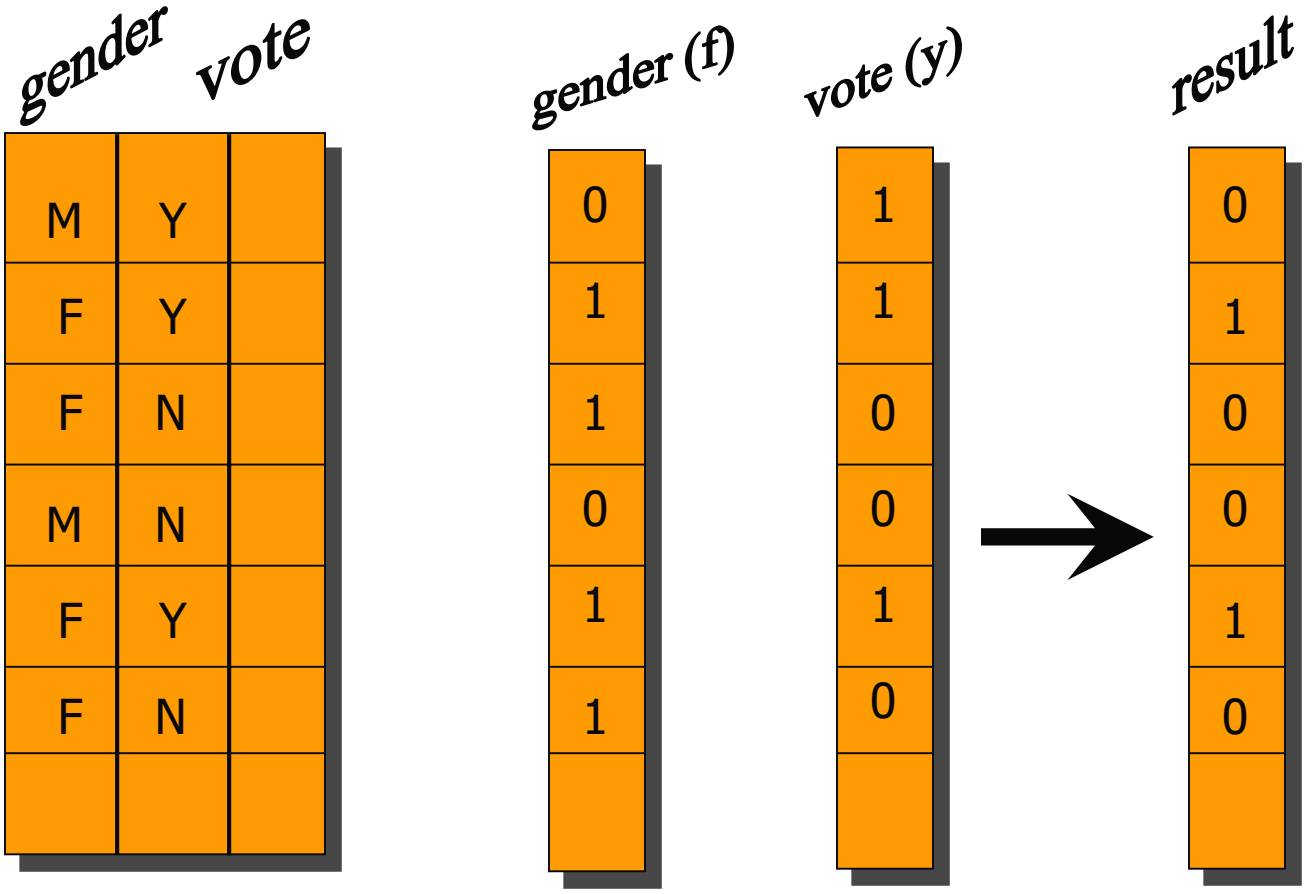
Query Processing

- Indexing
- Pre-computed views/aggregates
- SQL extensions

Indexing Techniques

- Exploiting indexes to reduce scanning of data is of crucial importance
- Bitmap Indexes
- Join Indexes
- Other Issues
  - Text indexing
  - Parallelizing and sequencing of index builds and incremental updates

BitMap Indexes: Examples



Customer

Query : select * from customer where gender = 'F' and vote = 'Y'

Bitmap Indexing

- A collection of bitmaps -- one for each distinct value of the column
- Each bitmap has N bits where N is the number of rows in the table
- A bit corresponding to a value *v* for a row *r* is set if and only if *r* has the value for the indexed attribute

BitMap Indexes: Examples

Base Table

Cust	Region	Rating
C1	N	H
C2	S	M
C3	W	L
C4	W	H
C5	S	L
C6	W	L
C7	N	H

Region Index

Row ID	N	S	E	W
1	1	0	0	0
2	0	1	0	0
3	0	0	0	1
4	0	0	0	1
5	0	1	0	0
6	0	0	0	1
7	1	0	0	0

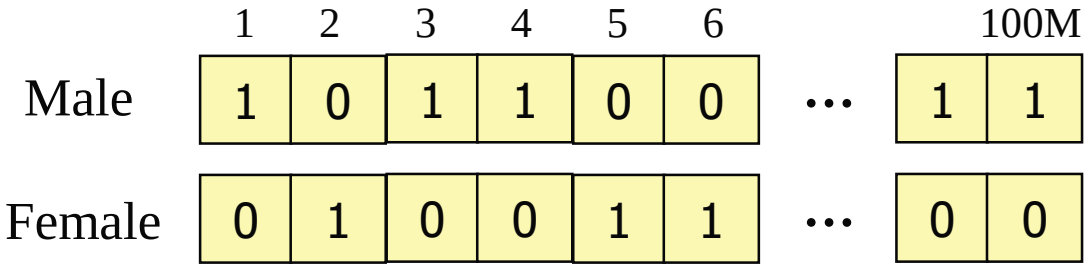
Rating Index

Row ID	H	M	L
1	1	0	0
2	0	1	0
3	0	0	1
4	1	0	0
5	0	0	1
6	0	0	1
7	1	0	0

Customers *where* **Region = W** ***and*** **Rating = M**

Alternative Representation of BitMap Indexes

- An alternative representation of RID-list
- Specially advantageous for low-cardinality domains
- Represent each row of a table by a bit and the table as a bit vector
- There is a distinct bit vector Bv for each value v for the domain
- Example: the attribute sex has values M and F. A table of 100 million people needs 2 lists of 100 million bits



BitMap Indexes

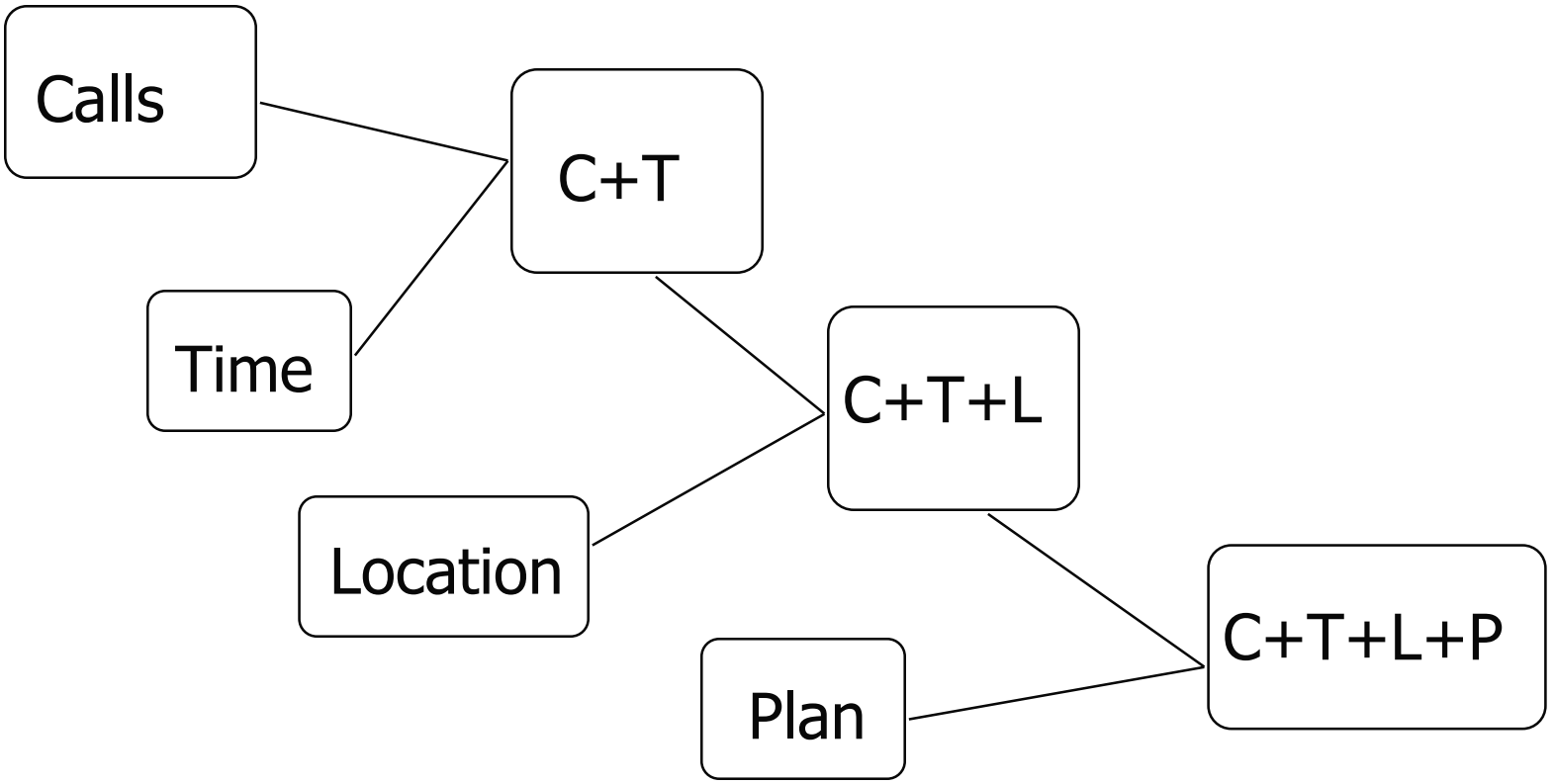
- Comparison, join and aggregation operations are reduced to bit arithmetic with dramatic improvement in processing time
- Significant reduction in space and I/O (30:1)
- Adapted for higher cardinality domains as well
- Compression (e.g., run-length encoding) exploited

Join Indexes

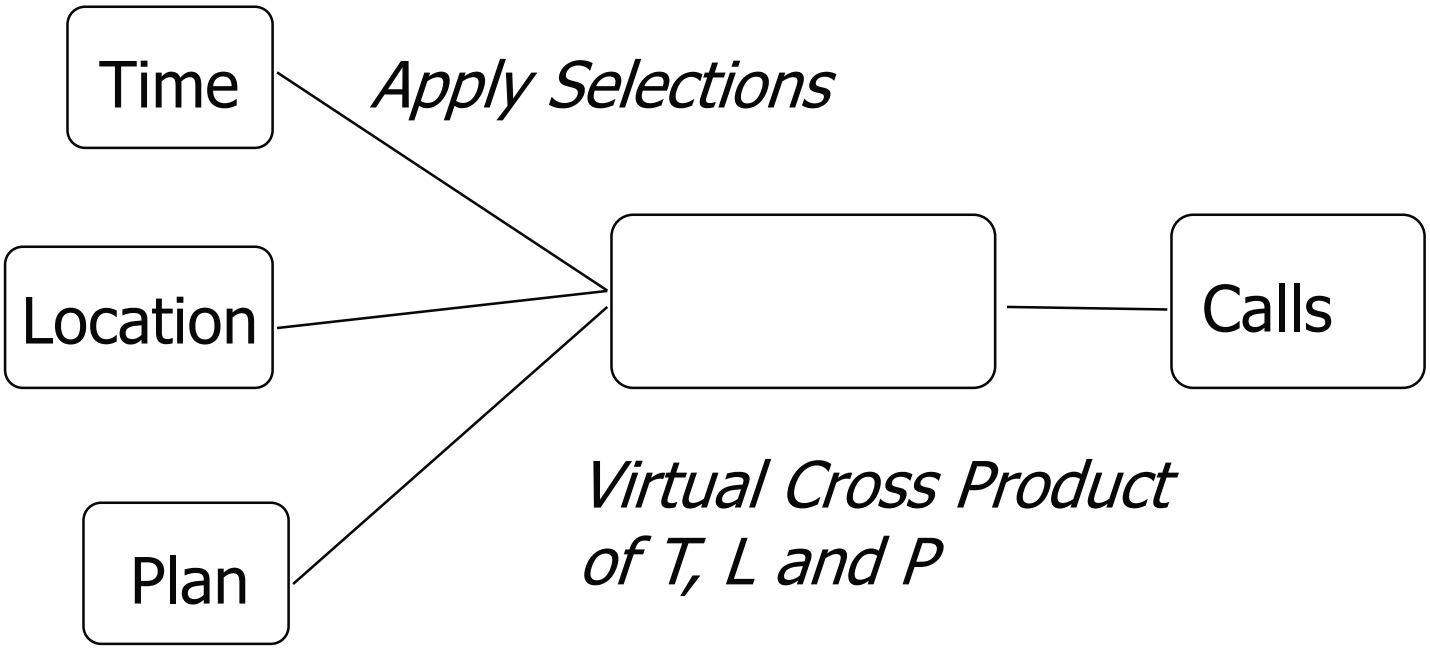
- Pre-computed joins
- A join index between a fact table and a dimension table correlates a dimension tuple with the fact tuples that have the same value on the common dimensional attribute
  - e.g., a join index on *city* dimension of *calls* fact table
  - correlates for each city the calls (in the *calls* table) from that city
- Join indexes can also span multiple dimension tables
  - e.g., a join index on *city* and *time* dimension of *calls* fact table

Star Join Processing

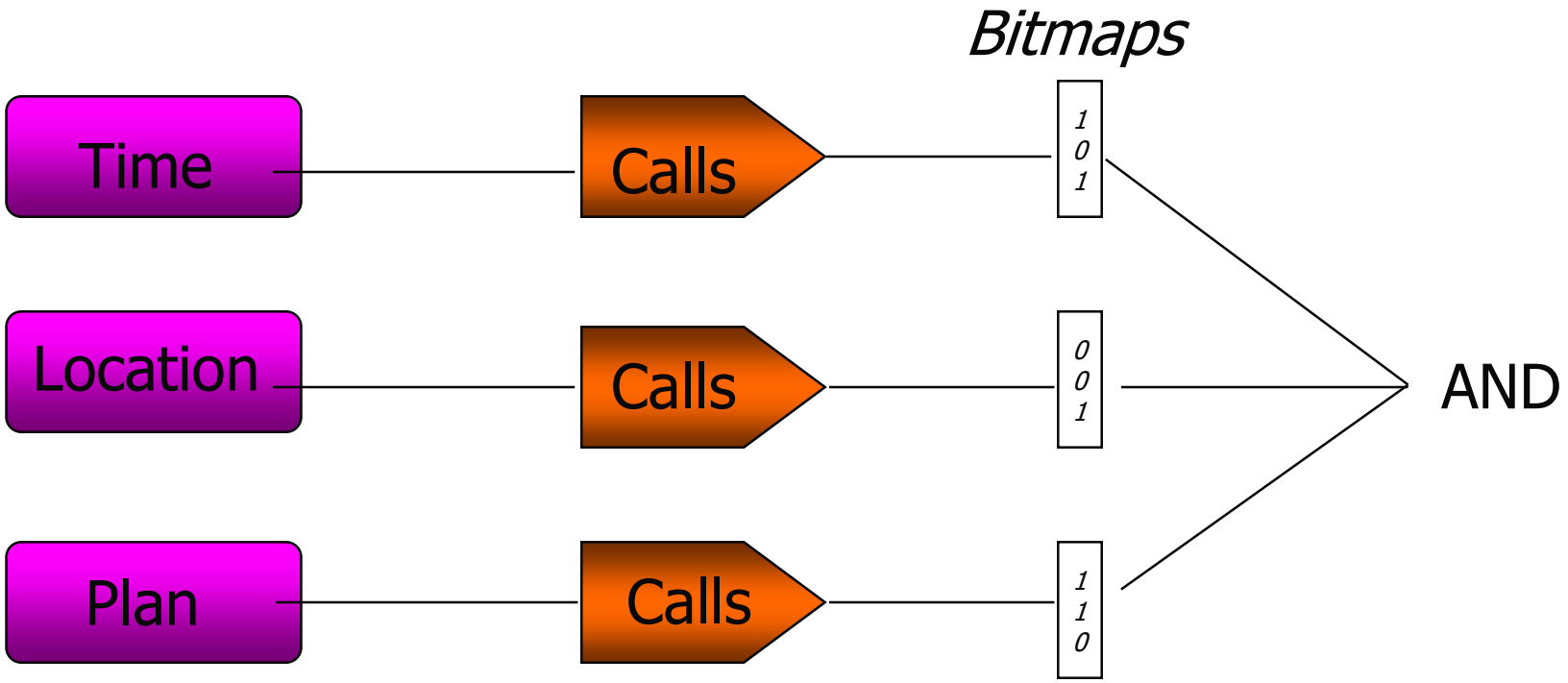
- Use join indexes to join dimension and fact table



Optimized Star Join Processing



Bitmapped Join Processing



Parallel Query Processing

- Three forms of parallelism
  - Independent
  - Pipelined
  - Partitioned and “partition and replicate”
- Deterrents to parallelism
  - startup
  - communication

Parallel Query Processing (*cont.*)

- Partitioned Data
  - Parallel scans
  - Yields I/O parallelism
- Parallel algorithms for relational operators
  - Joins, Aggregates, Sort
- Parallel Utilities
  - Load, Archive, Update, Parse, Checkpoint, Recovery
- Parallel Query Optimization

Pre-computed Aggregates

- Keep aggregated data for efficiency (pre-computed queries)
- Questions
  - Which aggregates to compute?
  - How to update aggregates?
  - How to use pre-computed aggregates in queries?

Pre-computed Aggregates (*cont.*)

- Aggregated table can be maintained by the
  - warehouse server
  - middle tier
  - client applications
- Pre-computed aggregates -- special case of materialized views -- same questions and issues remain

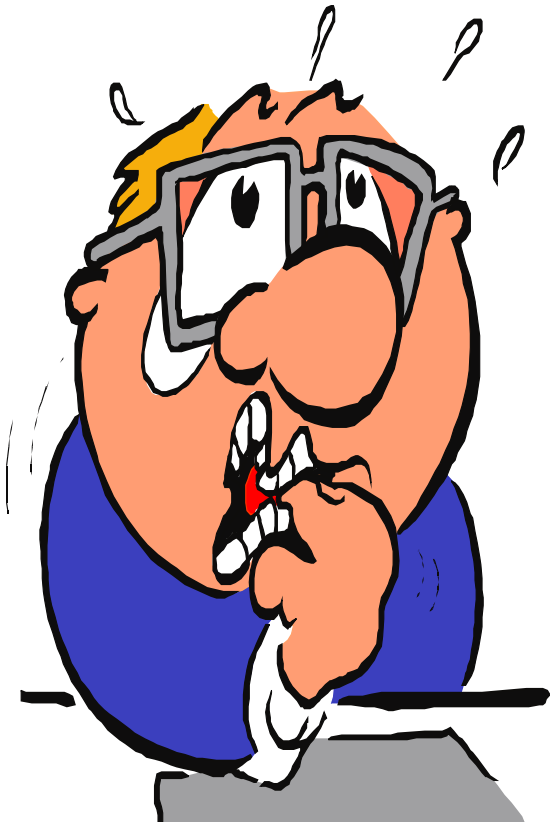
12. Fundamentals of Data Warehousing

- Introduction
- → Data warehouse
  - Architecture
  - Model
  - data marts
  - Querying
  - OLAP
- Providing Semantics to Data Warehouses
- Summary

On-Line Analytical Processing (OLAP)

Making Decision
Support Possible

Limitations of SQL



"A Freshman in Business needs a Ph.D. in SQL"

-- Ralph Kimball

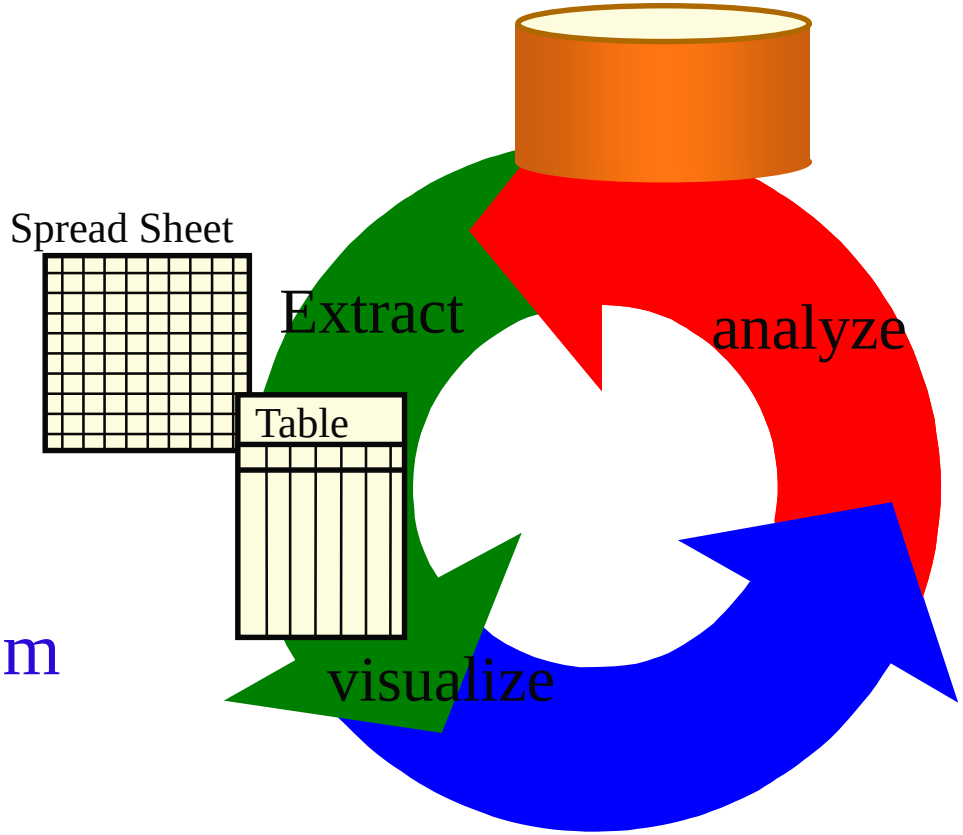
Typical OLAP Queries

- Write a multi-table join to compare sales for each product line this year vs. last year.
- Repeat the above process to find the top 5 product contributors to margin.
- Repeat the above process to find the sales of a product line to new vs. existing customers.
- Repeat the above process to find the customers that have had negative sales growth.

Nature of OLAP Analysis

- Aggregation -- (total sales, percent-to-total)
- Comparison -- Budget vs. Expenses
- Ranking -- Top 10, quartile analysis
- Access to detailed and aggregate data
- Visualization

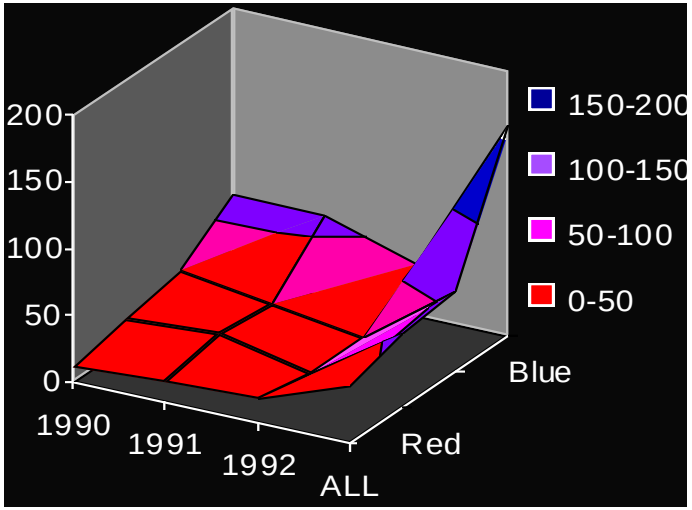
The Data Analysis Cycle

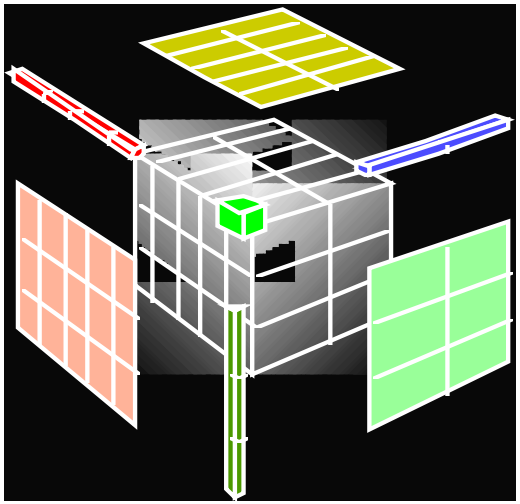


- User extracts data from database with query
- Then visualizes, analyzes data with tools

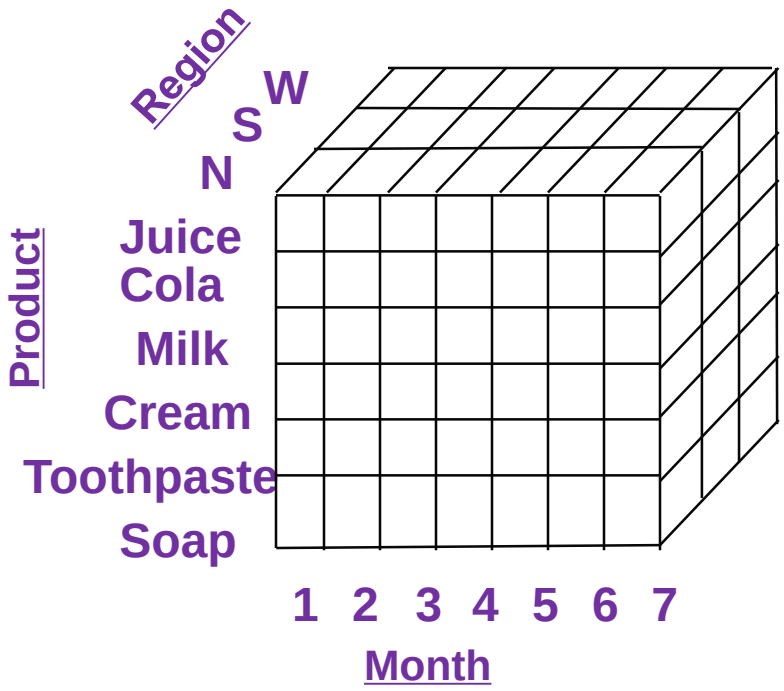
Organizationally Structured Data

- Different departments look at the same detailed data in different ways. Without the detailed organizationally structured data as a foundation, there is no reconcilability of data.
- Visualization System displays/explores the cube

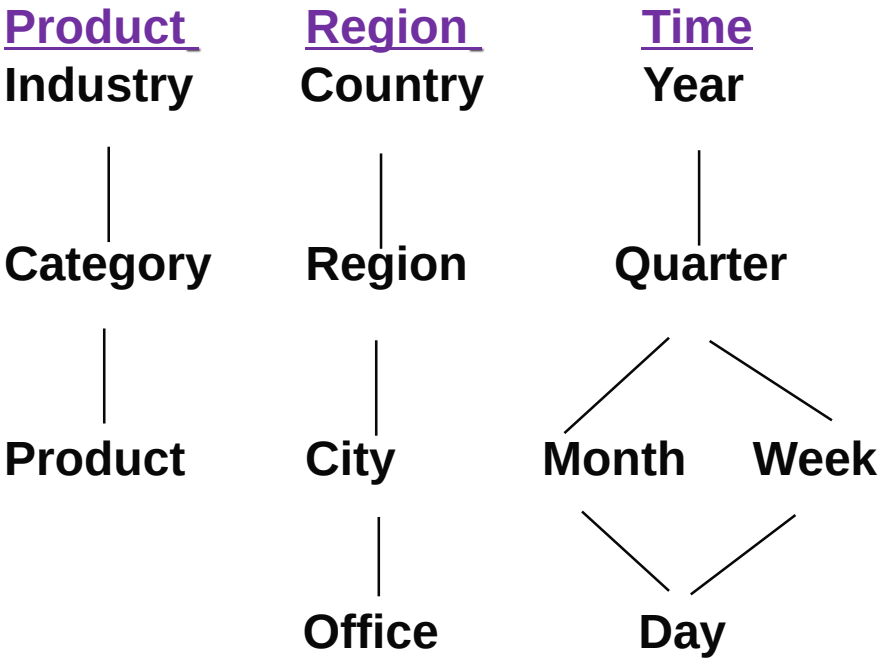




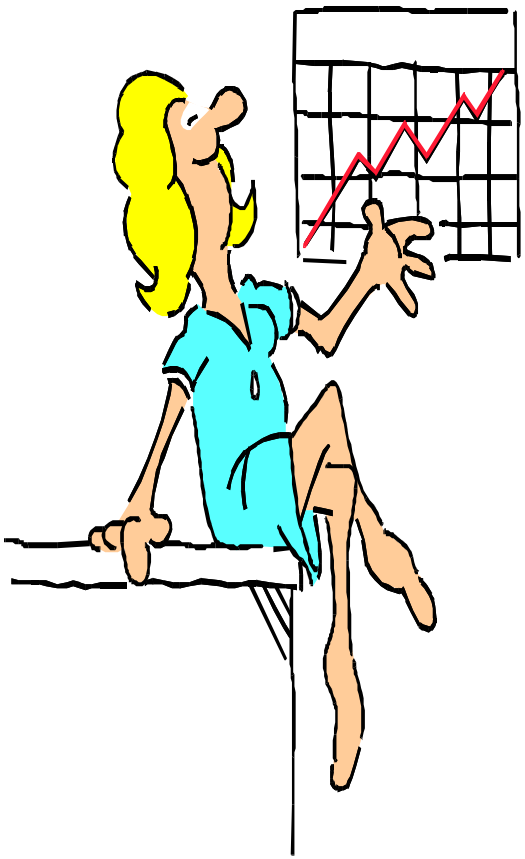
Multi-dimensional Data



Dimensions: Product, Region, Time
Hierarchical summarization paths

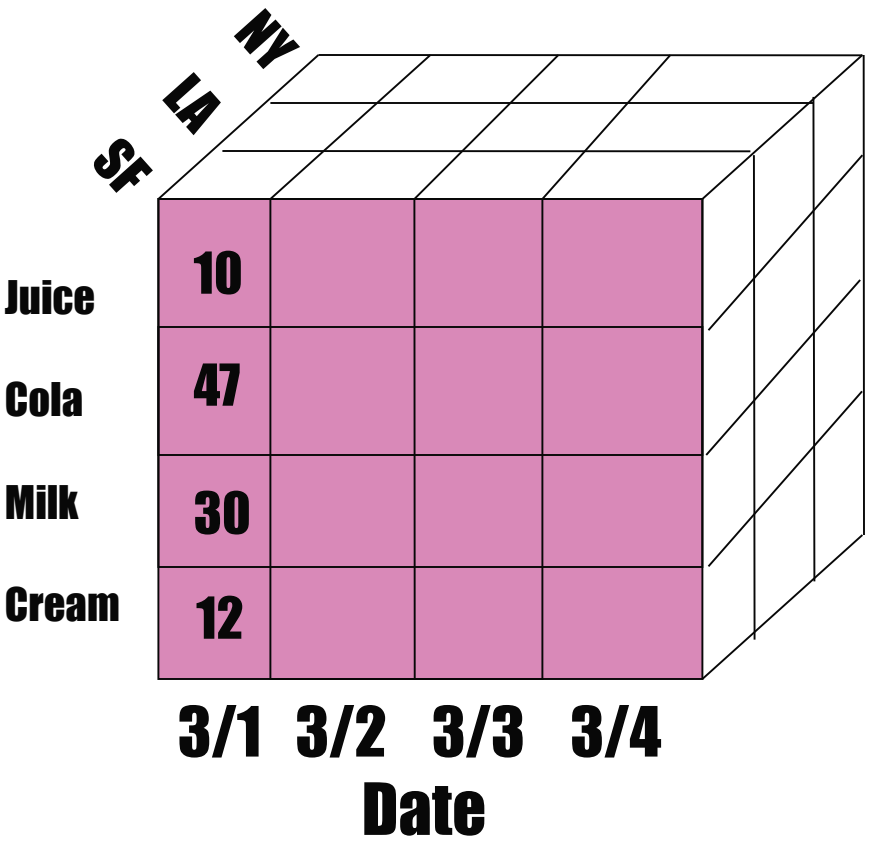


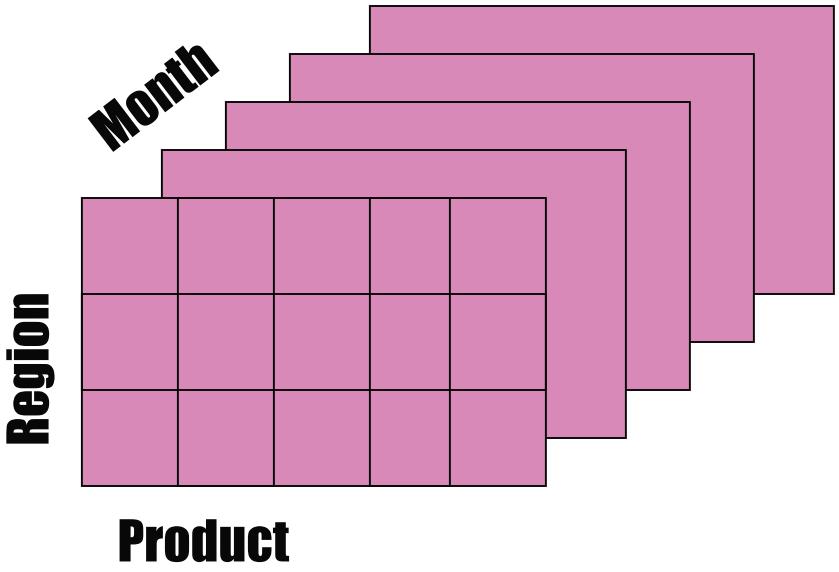
Multidimensional Analysis



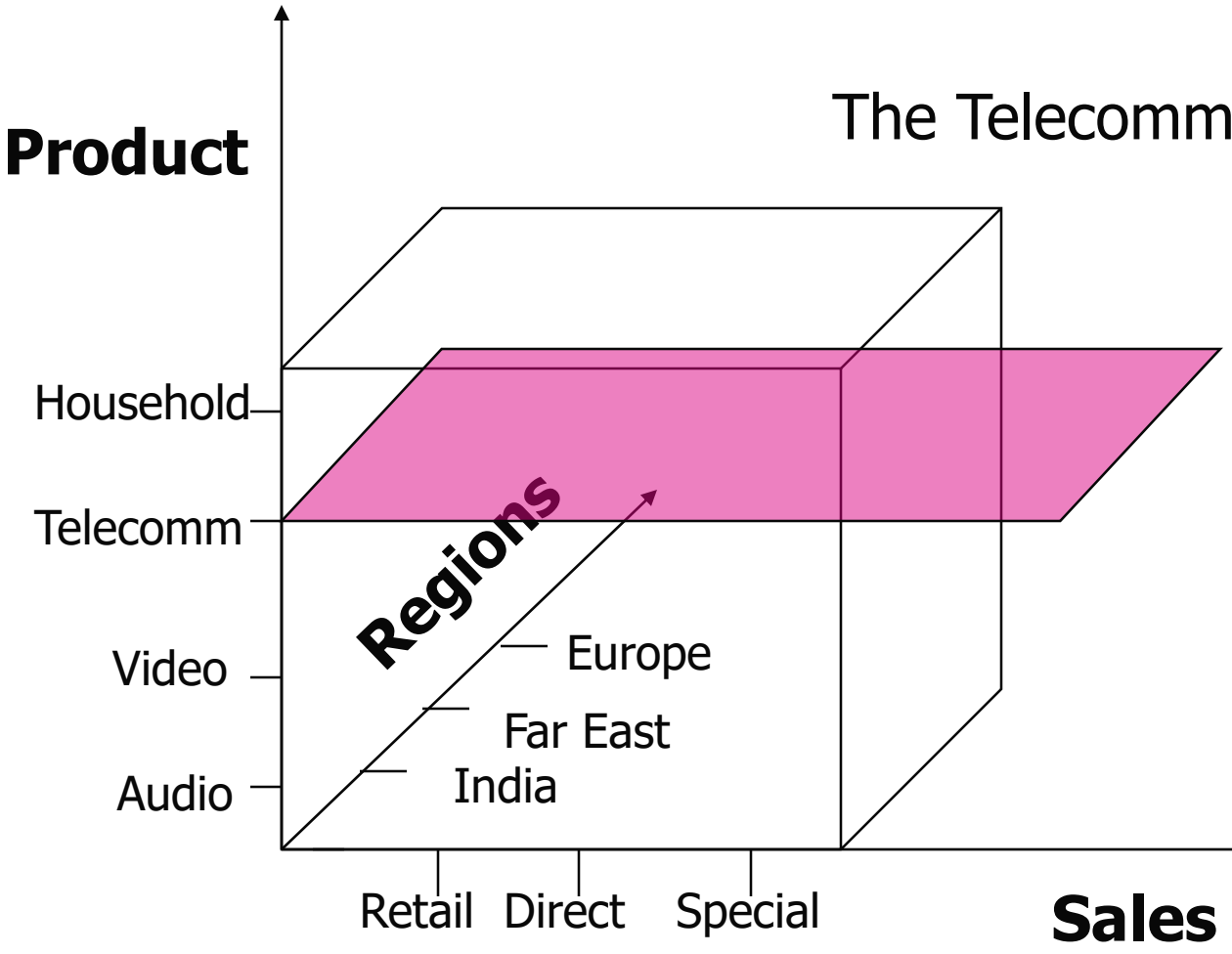
- Analysts need spreadsheets that support
  - pivot tables (cross-tabs)
  - drill-down and roll-up
  - slice and dice
  - sort
  - selections
  - derived attributes
- Popular in retail domain

A Visual Operation: Pivot (Rotate)

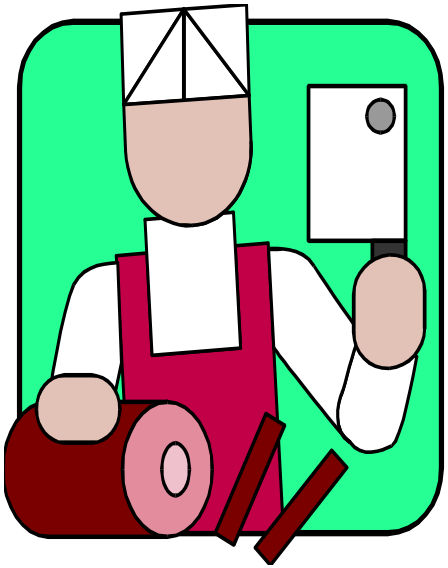




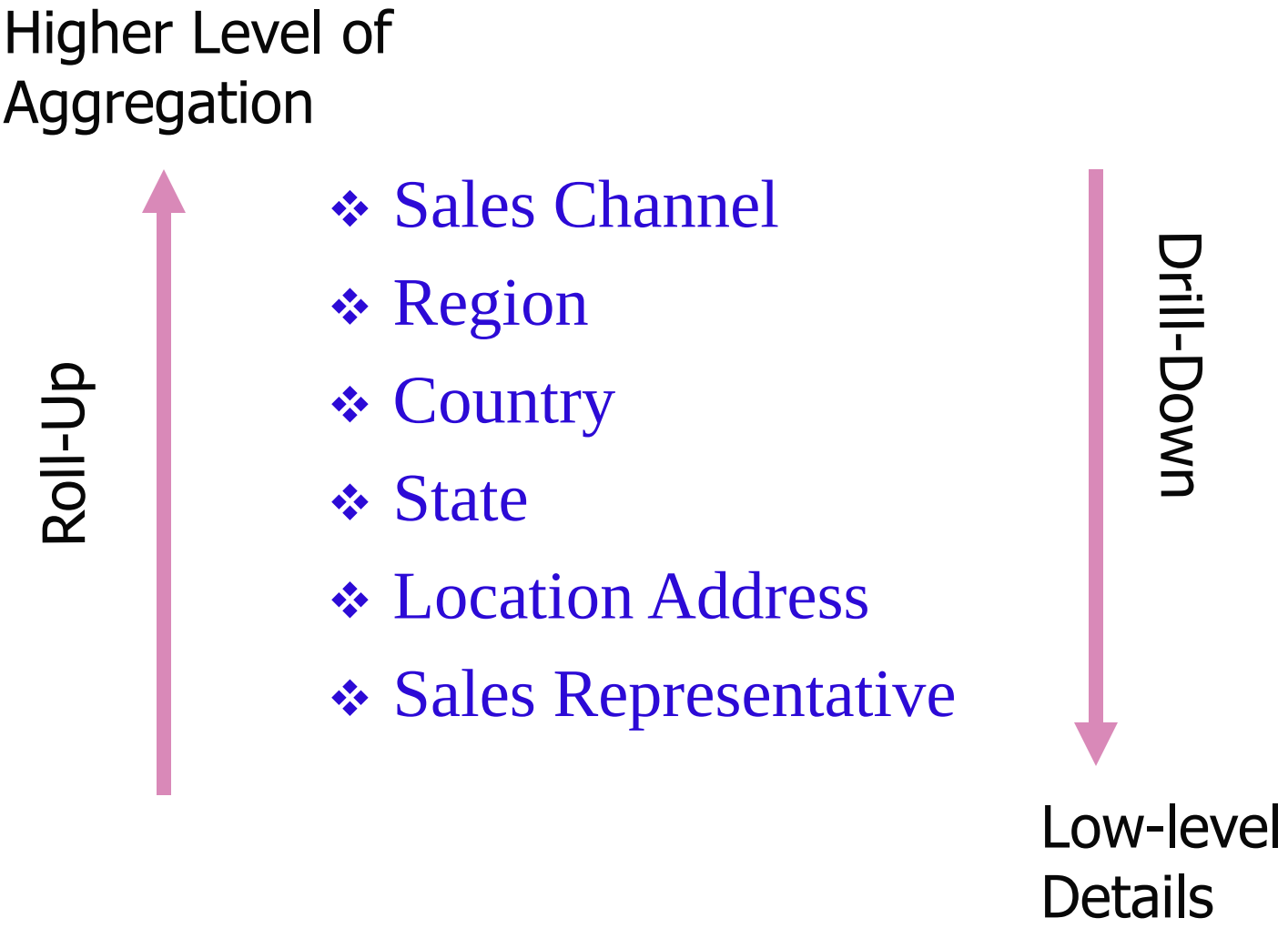
Slicing and Dicing



The Telecomm Slice



Roll-Up and Drill-Down



Visualizing Neighbors is simpler

	S1	S2	S3	S4	S5	S6	S7	S8
Apr								
May								
Jun								
Jul								
Aug								
Sep								
Oct								
Nov								
Dec								
Jan								
Feb								
Mar								

Month	Store	Sales
Apr	S1	
Apr	S2	
Apr	S3	
Apr	S4	
Apr	S5	
Apr	S6	
Apr	S7	
Apr	S8	
May	S1	
May	S2	
May	S3	
May	S4	
May	S5	
May	S6	
May	S7	
May	S8	
Jun	S1	
Jun	S2	

What Is OLAP?

- Online Analytical Processing - coined by EF Codd in 1994 paper
- Generally synonymous with earlier terms such as Decisions Support, Business Intelligence, Executive Information System
- OLAP = Multidimensional Database
- MOLAP: Multidimensional OLAP
- ROLAP: Relational OLAP

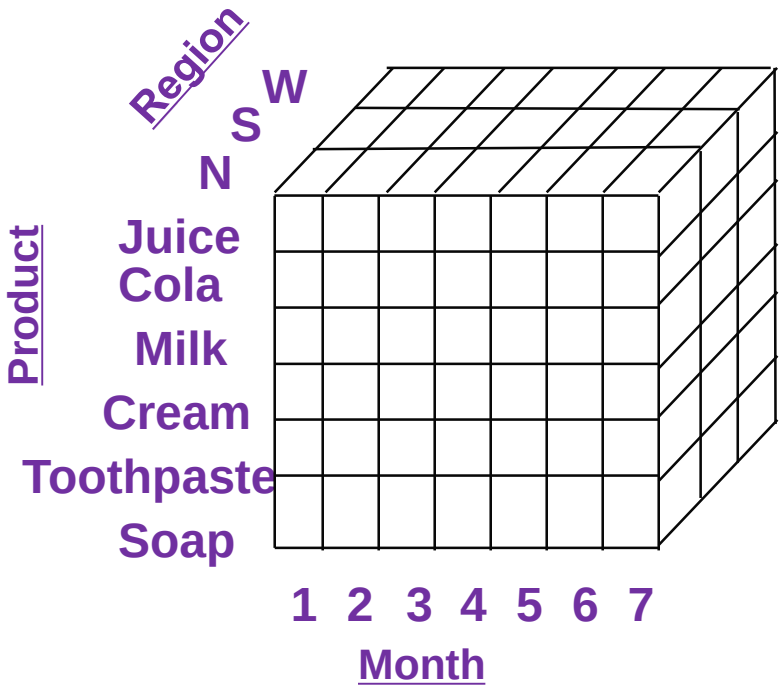
Strengths of OLAP

- It is a powerful visualization paradigm
- It provides fast, interactive response time
- It is good for analyzing time series
- It can be useful to find some clusters and outliers
- Many vendors offer OLAP tools

OLAP Is FASMI

- Fast
- Analysis
- Shared
- Multidimensional
- Information

Data Cube Lattice



- Can materialize some groupbys, compute others on demand
- Question: which groupbys to materialze?
- Question: what indices to create
- Question: how to organize data (chunks, etc.)

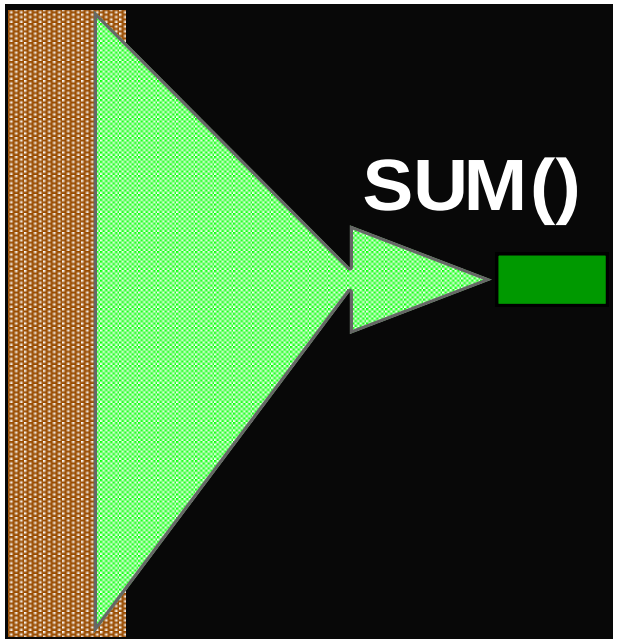
OLAP - Data Cube

- Analysts need to group data in many different ways
  - e.g. Sales (region, product, prodtype, prodstyle, date, saleamount)
  - saleamount is a measure attribute, rest are dimension attributes
  - groupby every subset of the other attributes
    - materialize (precompute and store) groupbys to give online response
  - Also: hierarchies on attributes: date - weekday, date - month - quarter - year

Relational Aggregate Operators

- SQL has several aggregate operators:
  - **sum( ), min( ), max( ), count( ), avg( )**
- Other systems extend this with many others:
  - **stat functions, financial functions, ...**
- The basic idea is:
  - **Combine all values in a column into a single scalar value.**
- Syntax

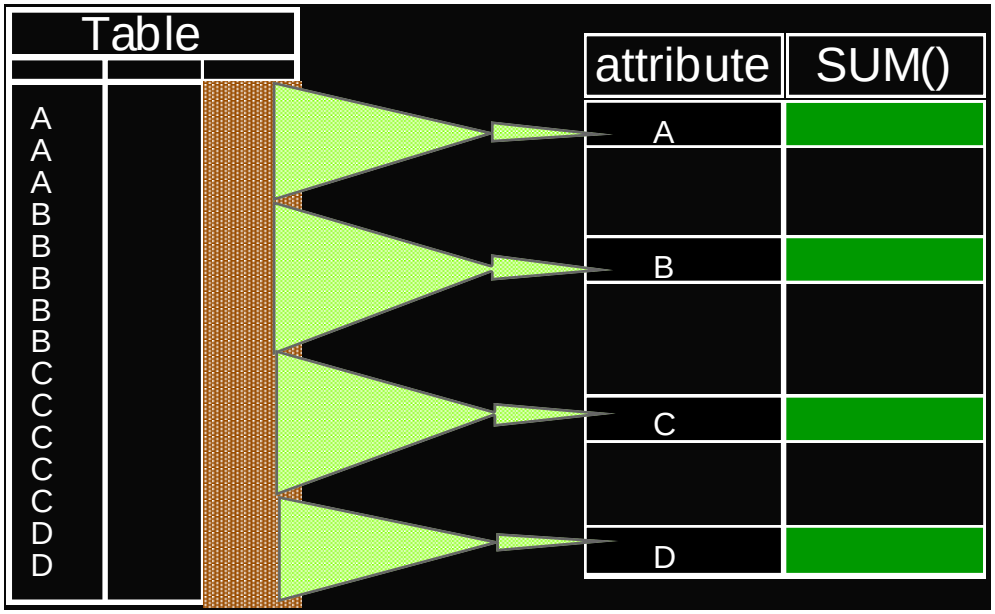
```
select  sum(units)
from    inventory;
```



Relational Group By Operator

- Group By allows aggregates over table sub-groups
- Result is a new table
- Syntax:

```
select      location, sum(units)
from        inventory
group by    location
having      nation = "USA";
```



Limitations With This Design

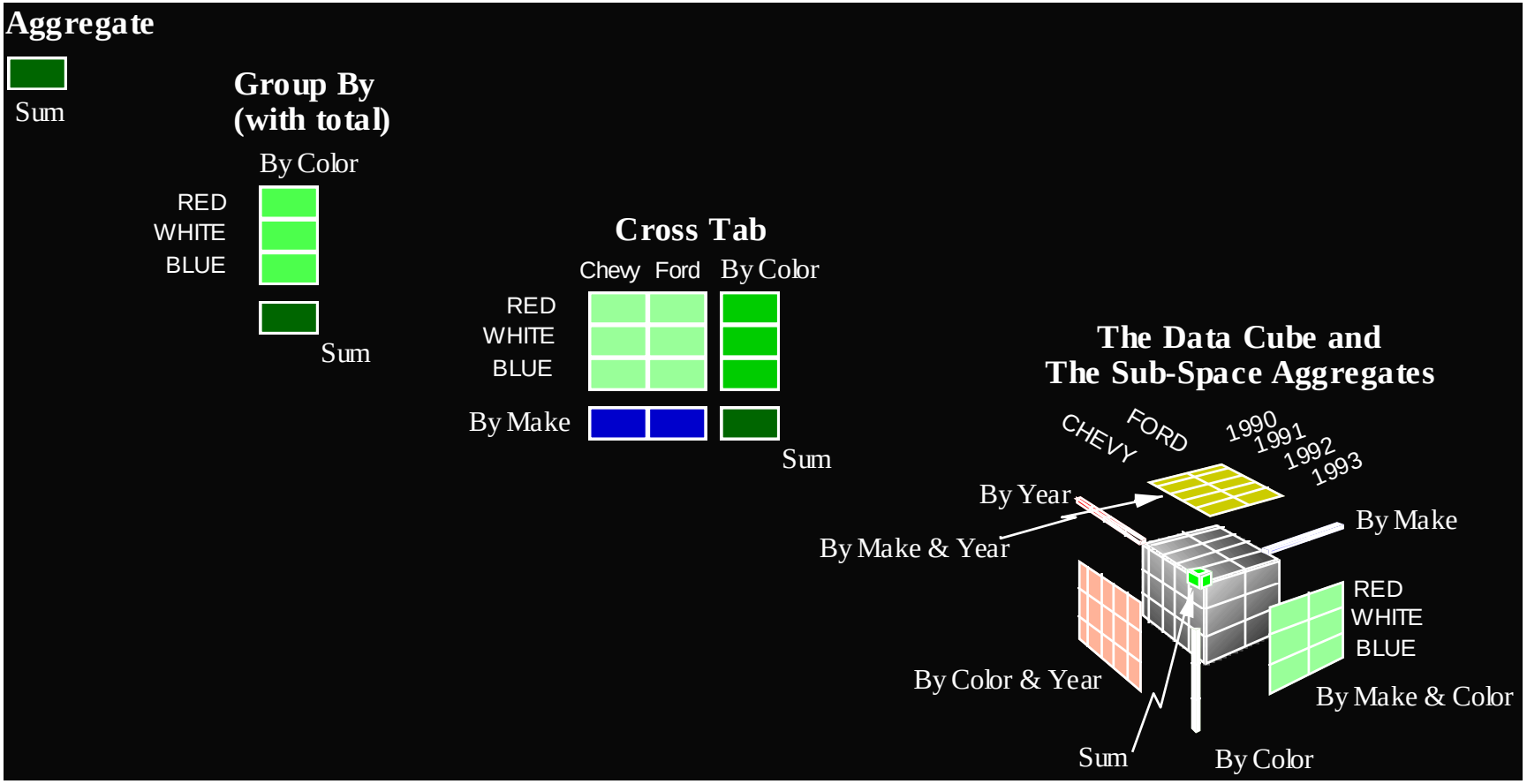
- Users Want Histograms

- Users want sub-totals and totals
  - drill-down & roll-up reports

- Users want CrossTabs

- Conventional wisdom
  - These are not relational operators
  - They are in many report writers and query engines

The Data CUBE Relational Operator Generalizes GroupBy and Aggregates

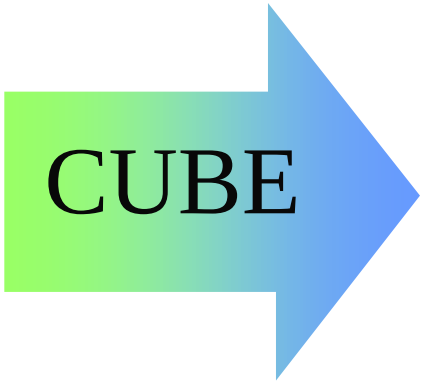


Think of the N-dimensional Cube Each Attribute is a Dimension

- N-Dimensional Aggregate (sum( ), max( ),...)
  - **fits relational model exactly:**
    - **a_1, a_2, ..., a_N, f( )**
- Super-aggregate over N-1 Dimensional sub-cubes
    - **ALL, a_2, ..., a_N, f( )**
    - **a_3 , ALL, a_3, ..., a_N, f( )**
    - **...**
    - **a_1, a_2, ..., ALL, f( )**
  - **this is the N-1 Dimensional cross-tab.**
- Super-aggregate over N-2 Dimensional sub-cubes
    - **ALL, ALL, a_3, ..., a_N , f( )**
    - **...**
    - **a_1, a_2 ,...., ALL, ALL, f( )**

An Example

SALES			
Model	Year	Color	Sales
Chevy	1990	red	5
Chevy	1990	white	87
Chevy	1990	blue	62
Chevy	1991	red	54
Chevy	1991	white	95
Chevy	1991	blue	49
Chevy	1992	red	31
Chevy	1992	white	54
Chevy	1992	blue	71
Ford	1990	red	64
Ford	1990	white	62
Ford	1990	blue	63
Ford	1991	red	52
Ford	1991	white	9
Ford	1991	blue	55
Ford	1992	red	27
Ford	1992	white	62
Ford	1992	blue	39



DATA CUBE			
Model	Year	Color	Sales
ALL	ALL	ALL	942
chevy	ALL	ALL	510
ford	ALL	ALL	432
ALL	1990	ALL	343
ALL	1991	ALL	314
ALL	1992	ALL	285
ALL	ALL	red	165
ALL	ALL	white	273
ALL	ALL	blue	339
chevy	1990	ALL	154
chevy	1991	ALL	199
chevy	1992	ALL	157
ford	1990	ALL	189
ford	1991	ALL	116
ford	1992	ALL	128
chevy	ALL	red	91
chevy	ALL	white	236
chevy	ALL	blue	183
ford	ALL	red	144
ford	ALL	white	133
ford	ALL	blue	156
ALL	1990	red	69
ALL	1990	white	149
ALL	1990	blue	125
ALL	1991	red	107
ALL	1991	white	104
ALL	1991	blue	104
ALL	1992	red	59
ALL	1992	white	116
ALL	1992	blue	110

Why the "ALL" Value?

- Need a new "Null" value (overloads the null indicator)
- Value must not already be in the aggregated domain
- Can't use NULL since may aggregate on it
- Think of ALL as a token representing the set
  - **{red, white, blue}, {1990, 1991, 1992}, {Chevy, Ford}**
- Rules for "ALL" in other areas not explored
  - **assertions**
  - **insertion / deletion / ...**
  - **referential integrity**
- Follow "set of values" semantics

CUBE operator: Syntax

```
select      model, make, year, sum(units)
from         car_sales
where       model in {"chevy", "ford"} and
             year between 1990 and 1994
group by  model, make, year with cube
having     sum(units) > 0;
```

- Note: GroupBy operator repeats aggregate list
  - **in select list**
  - **in group by list**

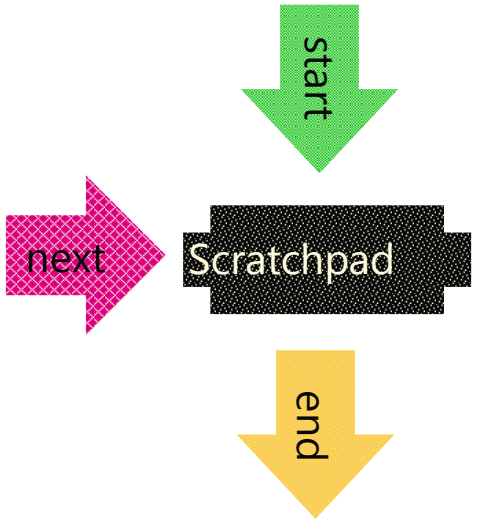
Interesting Aggregate Functions

- From RedBrick systems
  - **Rank (in sorted order)**
  - **N-Tile (histograms)**
  - **Running average (cumulative functions)**
  - **Windowed running average**
  - **Percent of total**
- Users want to define their own aggregate functions
  - **statistics**
  - **domain specific**

User Defined Aggregates

- Idea:
  - User function is called at start of each group
  - Each function instance has scratchpad
  - Function is called at end of group

- Example: SUM
  - START: allocates a cell and sets it to zero
  - NEXT: adds next value to cell
  - END: deallocates cell and returns value
  - Simple example: MAX ( )



- Need extension for rollup and cube

User Defined Aggregate Function Generalized For Cubes

- Aggregates have graduated difficulty
  - **Distributive: can compute cube from next lower dimension values (count, min, max,...)**
  - **Algebraic: can compute cube from next lower scratchpads (average, ...)**
  - **Holistic: need base data (Median, Rank, …)**
- Distributive and Algebraic have simple and efficient algorithm: build higher dimensions from core
- Holistic computation seems to require multiple passes
  - real systems use sampling to estimate them
    - e.g., sample to find median, quartile boundaries

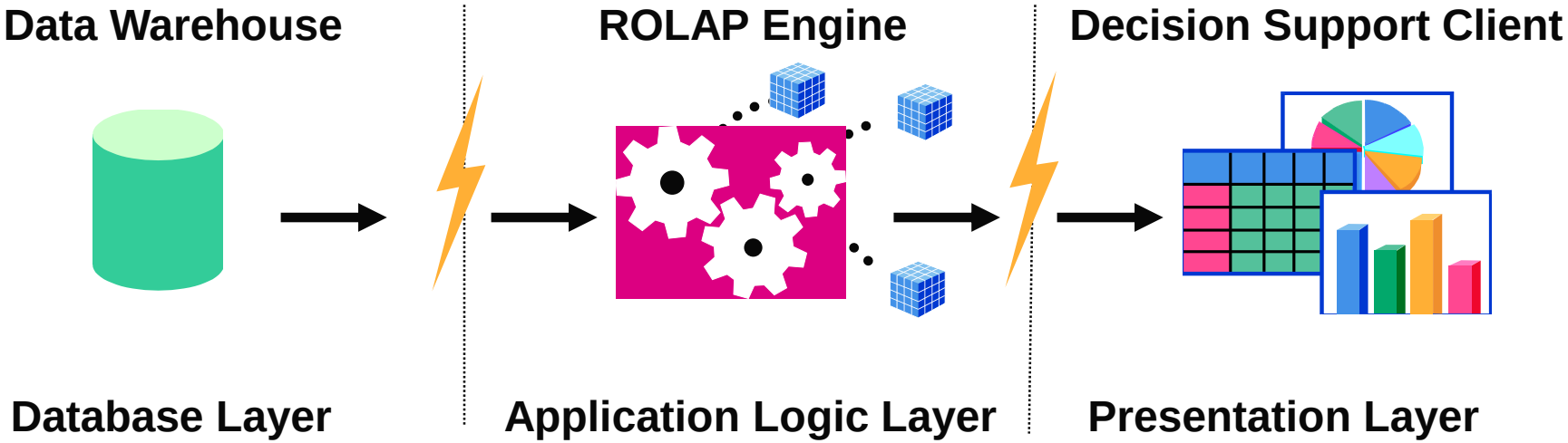
SQL Extensions

- Front-end tools require
  - Extended Family of Aggregate Functions
    - rank, median
  - Reporting Features
    - running totals, cumulative totals
  - Results of multiple group by
    - total sales by month, by product, …
  - Data Cube
    - group by on all subsets of a set of attributes (month, city)
    - redundant scan and sorting of data can be avoided

SQL Extensions (*cont.*)

- Extended family of aggregate functions
  - rank (top 10 customers)
  - percentile (top 30% of customers)
  - median
  - object relational systems allow addition of new aggregate functions

Relational OLAP: 3 Tier DSS

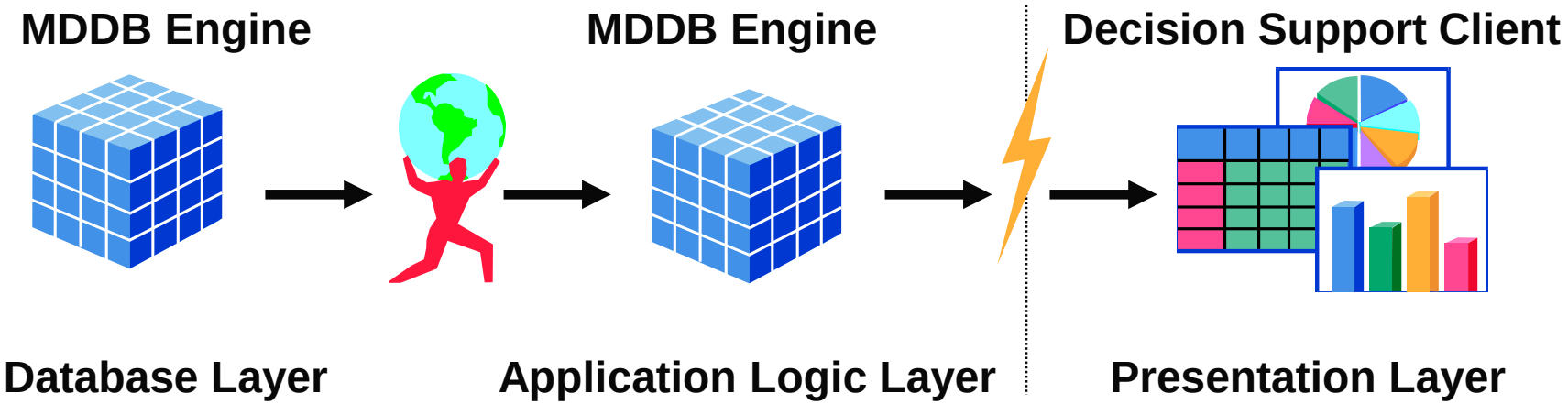


Store atomic data in industry standard RDBMS.

Generate SQL execution plans in the ROLAP engine to obtain OLAP functionality.

Obtain multi-dimensional reports from the DSS Client.

MD-OLAP: 2 Tier DSS



Store atomic data in a proprietary data structure (MDDB), pre-calculate as many outcomes as possible, obtain OLAP functionality via proprietary algorithms, running against this data.

Obtain multi-dimensional reports from the DSS Client.

Summary

- CUBE operator generalizes relational aggregates
- Needs ALL value to denote sub-cubes
  - **ALL values represent aggregation sets**
- Needs generalization of user-defined aggregates
- Decorations and abstractions are interesting
- Computation has interesting optimizations

Research Topics

- Generalize Spreadsheet Pivot operator to RDBs
- Characterize Algebraic/Distributive/Holistic functions for update

Metadata Repository

- Administrative metadata
  - source databases and their contents
  - gateway descriptions
  - warehouse schema, view & derived data definitions
  - dimensions, hierarchies
  - pre-defined queries and reports
  - data mart locations and contents
  - data partitions
  - data extraction, cleansing, transformation rules, defaults
  - data refresh and purging rules
  - user profiles, user groups
  - security: user authorization, access control

Metadata Repository (*cont.*)

- Business data
  - business terms and definitions
  - ownership of data
  - charging policies
- Operational metadata
  - data lineage: history of migrated data and sequence of transformations applied
  - currency of data: active, archived, purged
  - monitoring information: warehouse usage statistics, error reports, audit trails.

Recipe for a Successful Warehouse

From Larry Greenfield, http://pwp.starnetinc.com/larryg/index.html

From day one,

- establish that warehousing is a joint user/builder project
- establish that maintaining data quality will be an ongoing joint user/builder responsibility
- Train the users one step at a time
- Consider doing a high level corporate data model in no more than three weeks

For a Successful Warehouse

- Look closely at the data extracting, cleaning, and loading tools
- Implement a user accessible automated directory to information stored in the warehouse
- Determine a plan to test the integrity of the data in the warehouse
- From the start get warehouse users in the habit of 'testing' complex queries

For a Successful Warehouse (*cont*.)

- Coordinate system roll-out with network administration personnel
- When in a bind, ask others who have done the same thing for advice
- Be on the lookout for small, but strategic, projects
- Market and sell your data warehousing systems

Data Warehouse Pitfalls

- You are going to spend much time extracting, cleaning, and loading data
- Despite best efforts at project management, data warehousing project scope will increase
- You are going to find problems with systems feeding the data warehouse
- You will find the need to store data not being captured by any existing system
- You will need to validate data not being validated by transaction processing systems

Data Warehouse Pitfalls (*cont.*)

- Some transaction processing systems feeding the warehousing system will not contain detail
- Many warehouse end users will be trained and never or seldom apply their training
- After end users receive query and report tools, requests for IS written reports may increase
- Your warehouse users will develop conflicting business rules
- Large scale data warehousing can become an exercise in data homogenizing

Data Warehouse Pitfalls (*cont*.)

- 'Overhead' can eat up great amounts of disk space
- The time it takes to load the warehouse will expand to the amount of the time in the available window
- Assigning security cannot be done with a transaction processing system mindset
- You are building a high maintenance system
- You will fail if you concentrate on resource optimization to the neglect of project, data, and customer management issues and an understanding of what adds value to the customer

DW and OLAP Research Issues

- Data cleaning
  - focus on data inconsistencies, not schema differences
  - data mining techniques
- Physical Design
  - design of summary tables, partitions, indexes
  - tradeoffs in use of different indexes
- Query processing
  - selecting appropriate summary tables
  - dynamic optimization with feedback
  - acid test for query optimization: cost estimation, use of transformations, search strategies
  - partitioning query processing between OLAP server and backend server.

DW and OLAP Research Issues (*cont*.)

- Warehouse Management
  - detecting runaway queries
  - resource management
  - incremental refresh techniques
  - computing summary tables during load
  - failure recovery during load and refresh
  - process management: scheduling queries, load and refresh
  - query processing, caching
  - use of workflow technology for process management

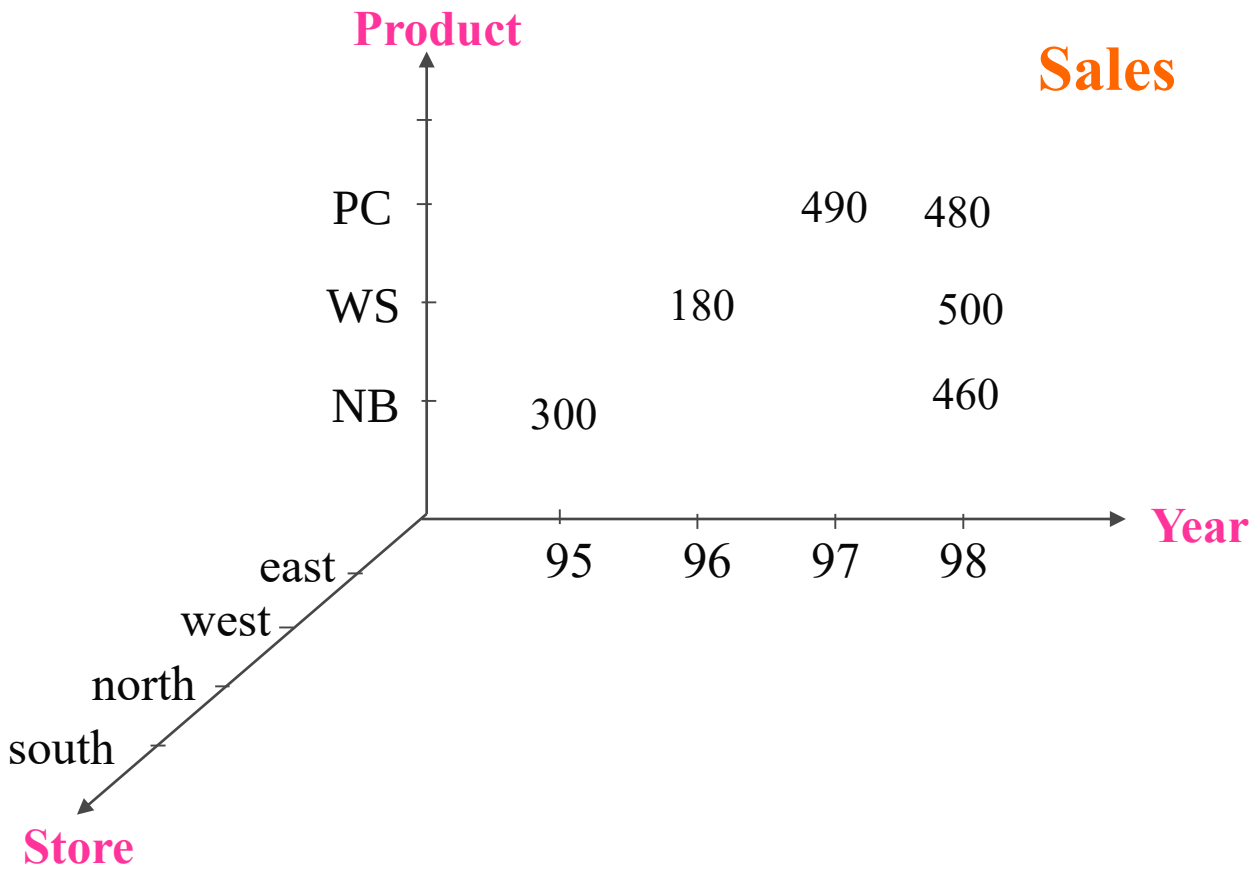
12. Fundamentals of Data Warehousing

- Introduction
- Data warehouse
  - Architecture
  - Model
  - Data marts
  - Querying
  - OLAP
- → Providing Semantics to Data Warehouse
- Summary

Providing Semantics to Data Warehouses

- Most analysis that current data warehousing systems provided is **numerically** oriented.
  - e.g., count the total sales of products within a certain period, calculate the average sales of products in a certain area, etc.

A 3-Dimensional View of a Chain Store



Quantitative Summary Data of a Chain Store

Product	*Year*	*Store*	*Sales*
PC	1997	East	490
PC	1997	West	200
PC	1998	South	480
PC	1998	West	450
PC	1998	East	300
PC	1998	North	50
Workstation	1998	East	500
Notebook	1998	East	460
..	..	..	..

Queries on the Summary Data

- The headquarters makes business decisions through a number of analysis-oriented queries like:
  - **Q1:** "For each member store, find the total sales amount in 1998."
  - **Q2:** "For each member store, find products whose total sales in 1998 were above 220."
  - **Q3:** "Find products whose sales difference in the past two years was less than 50."

The Inadequacy

- Most analysis that current data warehousing systems provided is **numerically** oriented (e.g., total amount).
- The real **meaning** of the data has been ignored.
  - *Whether a total sales amount **1000** items indicates a **good** or **bad** sales performance* is still unclear.
  - From the decision makers' point of view -
    - the **semantics** rather than raw **numbers** are more natural, meaningful, and understandable.
    - it is not easy to decide query criteria with crisp numbers; to reach the desired number of answers, a sequence of queries need to be formulated and tested.

Queries for Decision Support

- Traditional queries are like:
  - **Q1:** "*For each member store, find the total sales amount in 2000.*"
- How about the following queries?
  - **Q2:** "*Tell me the sales performance **(good/medium/bad)** of each member store in 2000.*"

linguistic term

Q3: "*Tell me the years in which **most/average/few** of products had **very** **good(medium/bad)** sales performance.*"

linguistic hedge

linguistic quantifier

Queries by Linguistic Terms

- Instead of querying detailed sales amounts via Q1, Q2, Q3, users would prefer to know sales performance via:
  - **Q1':** "How was the sales performance (***good*** or ***medium*** or ***bad***) of each member store in 1998."
  - **Q2':** "For each member store, find products whose sales performance in 1998 was ***good***."
  - **Q3':** "Find products whose sales in the past two years were ***very, very similar***."

Queries by Linguistic Quantifies

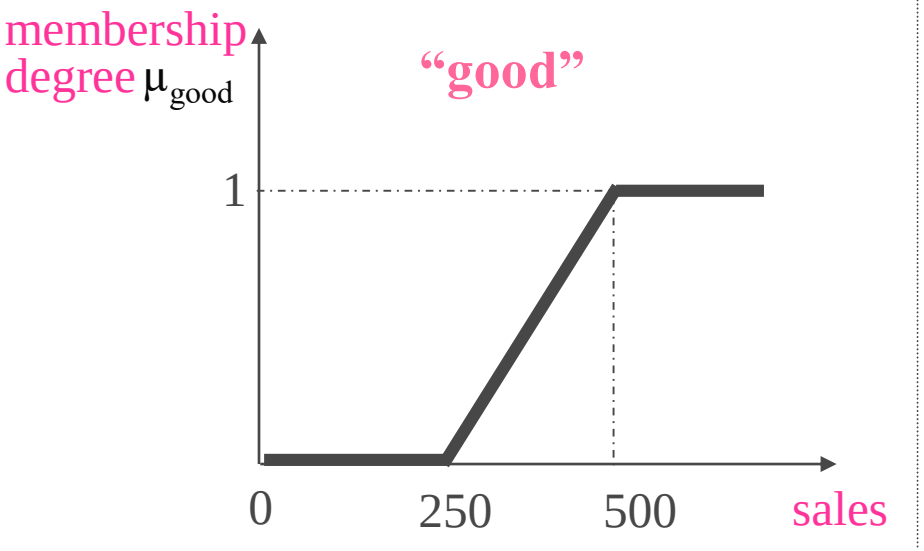
- Besides querying *individual* records, managers often want to have some insight into the overall performance of a *group* of records, like:
  - **Q1":** "How did ***most*** of products behave last year?"
  - **Q2":** "What was the ***average*** performance of products in the southern area?"
  - **Q3":** "***At least*** what percent of products' behavior was ***good*** recently?"

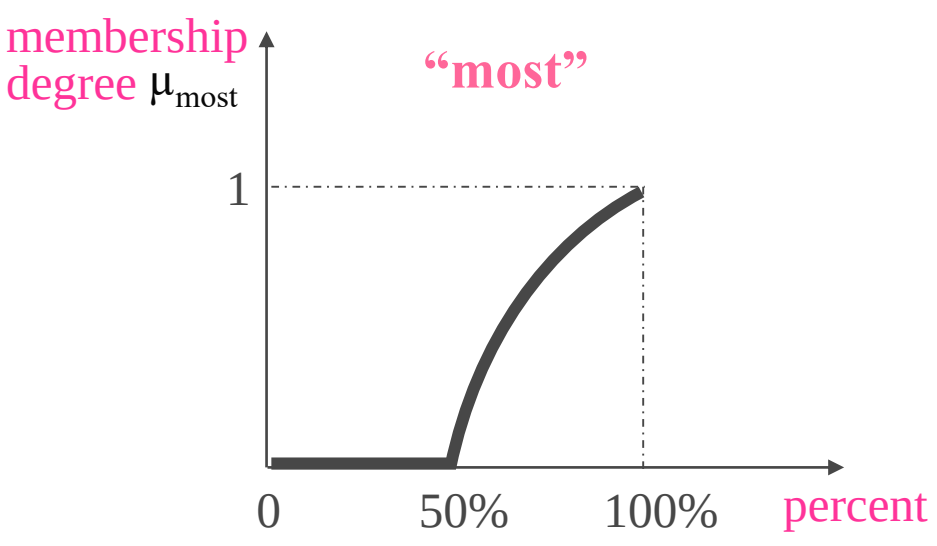
Membership Functions and Fuzzy Sets

- Fuzzy technology provides a framework for modeling the interface between human conceptual categories and data.

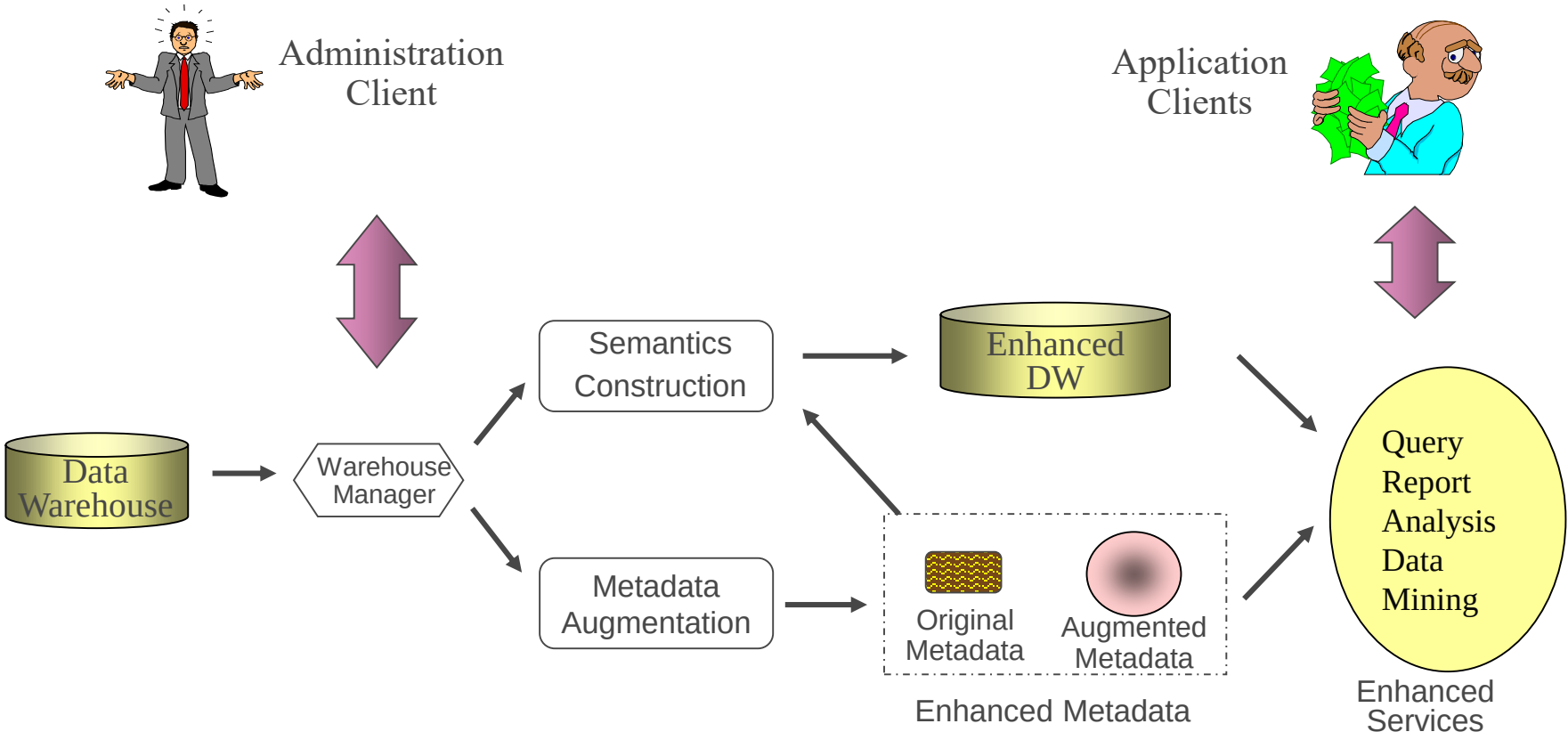
- **Membership function** & **Fuzzy set**
  - Expressing classes (*good*, *median*, *bad*, etc.) with ill-defined boundaries.
  - Modeling a gradual rather than sharp transition between non-membership and full membership.
  - The membership assigning function is called a membership function; $\mu_A : X \rightarrow [0,1]$
  - The set defined by a membership function is a fuzzy set.

Membership Function - Examples





Providing Explanatory Semantics to a DW

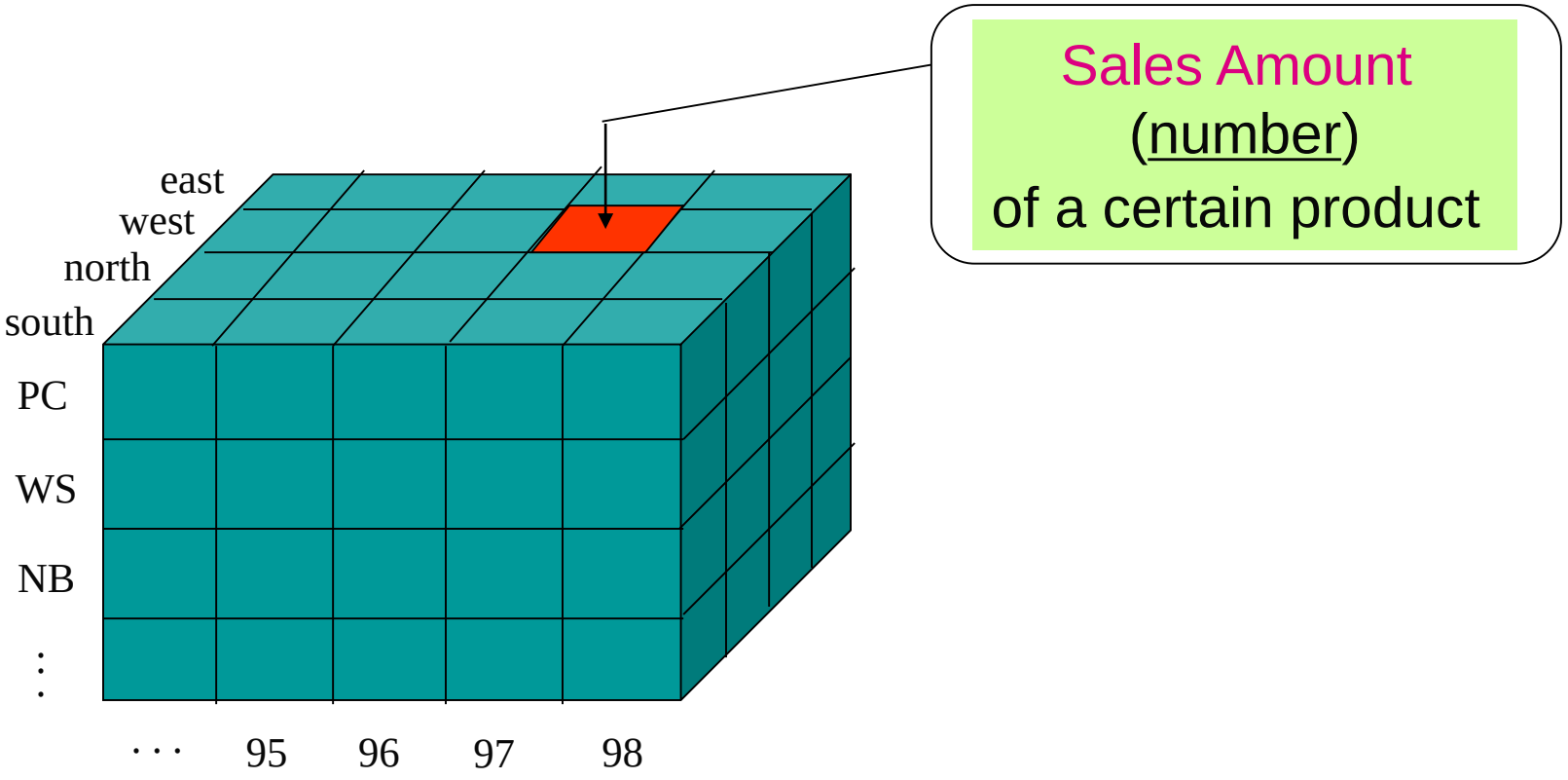


A framework for enhancing the semantics of a DW

A Three-Layered DW Semantic Model

- **Level-1:** Quantitative (Numerical) Summarization
  - Traditional aggregate views of data warehouses, e.g.,
    - Total sales amount, average sales, etc.
- **Level-2:** Qualitative (Categorical) Summarization
  - Generalization on Level-1 in **linguistics labels**, e.g.,
    - Products are categorized by *good* (*medium* or *bad*) sales, etc.
- **Level-3:** Quantifier Summarization
  - Generalization on Level-2 in **linguistics quantifies**, e.g.,
    - M*ost* (*half* or *few*) of product's behavior is *good* (*medium* or *bad*).

Level-1: Quantitative Summarization



Level-1: Quantitative Summarization (cont.)

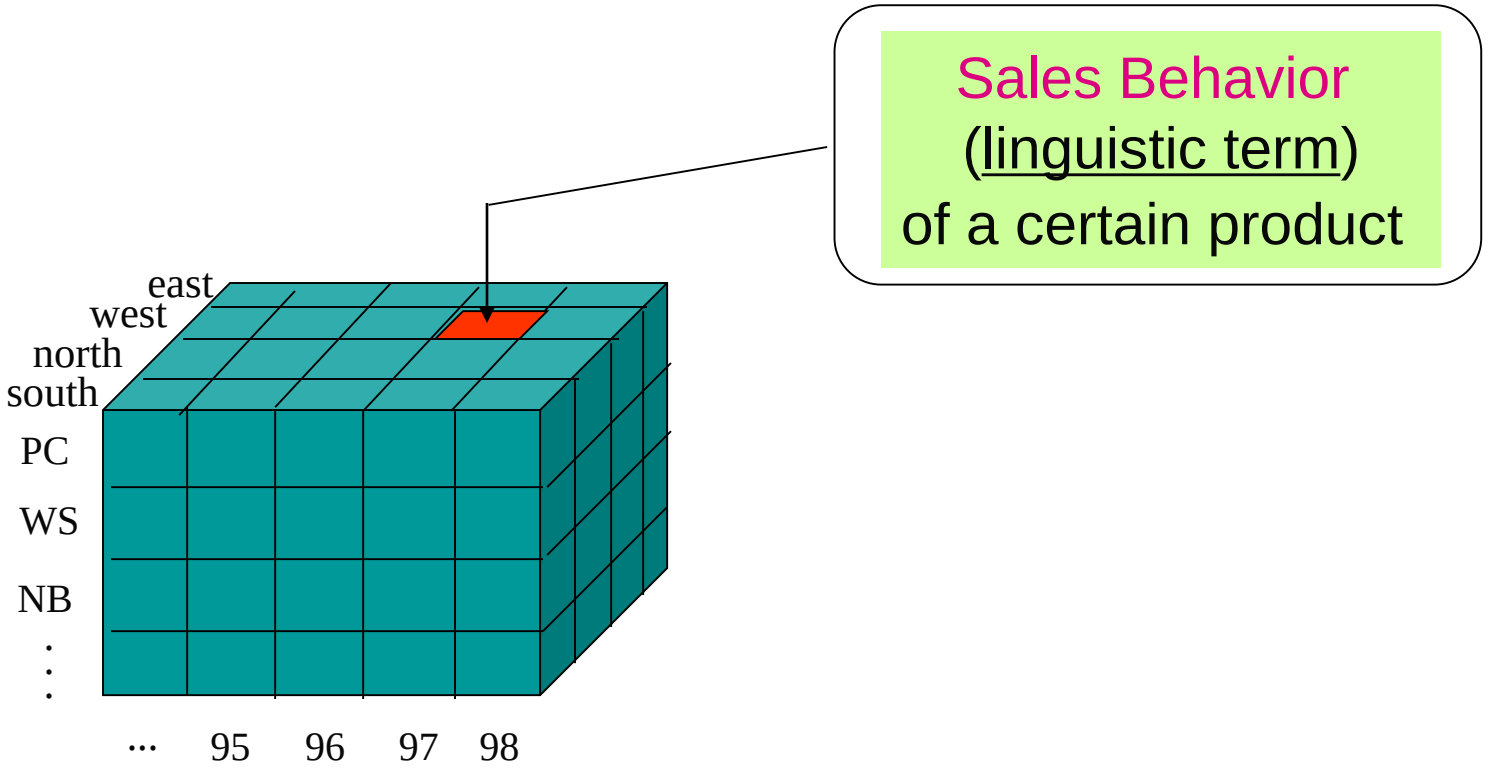
- **Definition**
  - **An *n*-dimensional quantitative data cube C_q^n** can be defined as a mapping function:

    $$F_q(C_q^n) : dom(D_1) \times \ldots \times dom(D_n) \rightarrow V$$

    which maps a set of dimensional values to a set of scalar values V, including a *NULL* value.

- **Example**
  - $F_q(C_q^3)(PC, 1997, east) = 490$
    $F_q(C_q^3)(PC, 2000, east) = NULL$

Level-2 : Qualitative Summarization



Level-2: Qualitative Summarization (cont.)

- **Definition**
  - **An *n*-dimensional qualitative data cube C_c^n** can be defined as a mapping function:

$$F_c(C_c^n): dom(D_1) \times \ldots \times dom(D_n) \rightarrow \{< T, [0,1] >\}$$

which maps a set of dimensional values to a set of 2-tuples, each indicating a linguistic term and a membership grade.

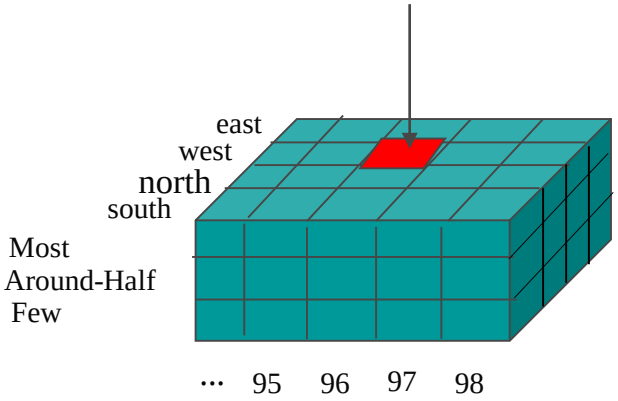
- **Example**
  - $v = F_q(C_q^3)(PC, 1997, east) = 490$ -- measurement at Level-1
    $\mu_{good}(490) = 0.96, \quad \mu_{medium}(490) = 0, \quad \mu_{bad}(490) = 0.$
  - $F_c(C_c^3)(PC, 1997, east) = \{< t, \mu_t(v) >\} = \{< good, 0.96 >\}$
    -- measurement at Level-2

Qualitative Summary Data of a Chain Store

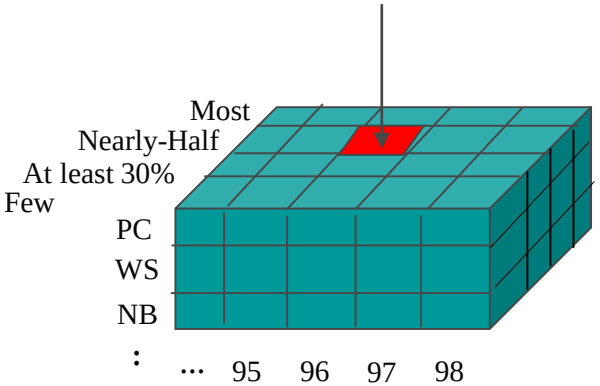
Product	*Year*	*Store*	*Sales*
PC	1997	East	<good, 0.96>
PC	1997	West	<medium, 0.75>
PC	1998	South	<good, 0.92>
PC	1998	West	<good, 0.8>
PC	1998	East	<medium, 0.75>, <good, 0.2>
PC	1998	North	<bad, 0.5>
Workstation	1998	East	<good, 0.1>
Notebook	1998	East	<good, 0.84>
..	..	..	..

Level-3: Quantifier Summarization

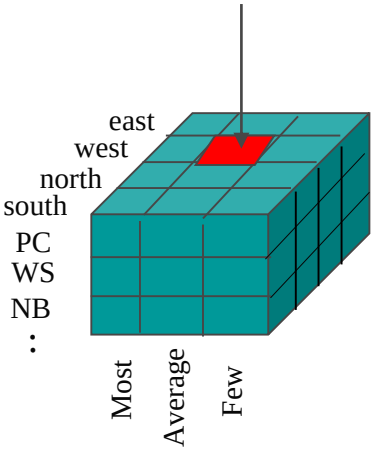
Sales Behavior (linguistic quantifier) of overall products



Sales Behavior (linguistic quantifier) of overall stores



Sales Behavior (linguistic quantifier) of overall years



Level-3: Quantifier Summarization (cont.)

- **Definition**
  - **An *n*-dimensional quantifier data cube C_f^n** can be defined as a mapping function:

$$F_f(C_f^n): dom(D_1) \times \ldots \times dom(D_n) \rightarrow \ < T, \ [0,1] >$$

which maps a set of dimensional values to a set of 2-tuples, each indicating a linguistic term and a membership grade of a quantifier.

Let $v_t = \boldsymbol{f}(\boldsymbol{t}, \boldsymbol{G})$ $(0 \leq v_t \leq 1)$, and $\mu_q(v_t)$ be the membership grade of the quantifier $q \in Q$. Let $\mu_q(v_{t'}) = \text{Max}_{t \in T}(\mu_q(v_t))$, we have

$$F_f(C_f^n)(d_1, \ldots, d_{i-1}, q, d_{i+1}, \ldots, d_n) = <t', \ \mu_q(v_{t'})>$$

Level-3: Quantifier Summarization (cont.)

- **Example**
  - A group $G = \{ (PC, 1998, east), (WS, 1998, east), (NB, 1998, east) \}$
    $F_c(C_c^3)(PC, 1998, east) = \{ <medium, 0.75>, <good, 0.2> \}$
    $F_c(C_c^3)(NB, 1998, east) = \{ <good, 1.0> \}$
    $F_c(C_c^3)(WS, 1998, east) = \{ <good, 0.84> \}$

    -- measurement at Level-2
  - $f(good, G) = (0.2+1.0+0.84)/3 = 0.68$
    $f(medium, G) = (0.75+0+0)/3 = 0.25$
    $f(bad, G) = (0+0+0)/3 = 0$

    $\mu_{most}(0.68) = 0.6$, $\mu_{most}(0.25) = 0$, $\mu_{most}(0) = 0$, hence $t' = good$
  - $\boldsymbol{F_f(C_f^3)(most, 1998, east) = < t', \mu_q(t') > = < good, 0.6 >}$

    -- measurement at Level-3

Quantifier Summary Data of the Chain Store

Product	***Year***	***Store***	***Sales***
most	1998	East	<good, 0.6>
few	1998	West	<bad, 1.0>
Around half	1998	South	<good, 0.5>
..	..	..	..

Algebraic Operators

- Pivot-Oriented Operators
  - Slice: $\Pi\,(C_s^{\,n}, D_{k_1}, \ldots, D_{k_m}) = C_s^{\,'m} \quad (\, s \in \{q,c,f\}\,)$
  - Dice: $\sigma\,(C_s^{\,n}, P) = C_s^{\,'n}$
  - Sift: $\varsigma\,(C_s^{\,n}, r, [t], l) = C_s^{\,'n}$
  - Roll-up: $\Uparrow (C_s^{\,n}, D_{k_1}, \ldots, D_{k_{n-1}}, f_{combine}) = C_s^{\,'n-1}$
  - Drill-down: $\Downarrow (C_s^{\,n}, D_{n+1}, f_{split}) = C_s^{\,'n+1}$

- Measurement-Oriented Operators - on ***compatible*** data cubes
  - Union: $\cup\,(C_s^{\,n}, C_s^{\,'n}, \oplus) = C_s^{\,''m}$
  - Intersect: $\cap\,(C_s^{\,n}, C_s^{\,'n}, \otimes) = C_s^{\,''m}$
  - Difference: $-\,(C_s^{\,n}, C_s^{\,'n}, \Theta) = C_s^{\,''m}$

Querying the Enhanced Data Warehouses

SELECT [n] [distinct] <select-list>

FROM <table-list>

[**WHERE** <predicate-list>]

[**GROUP BY** <grouping-attribute list>]

[**HAVING** <predicate-list>]

[**ORDER BY** <attribute-list>] [**DESC** |**ASC**]

<attribute> ::= <normal attribute> | <fuzzified attribute> | <quantified attribute> | <membership attribute>

<predicate> ::= <normal predicate> | <enhanced predicate>

<enhanced predicate >::= (<attribute> [mod]* <op_e> [mod]* <linguistic constant>| <attribute> [mod]* < op_e> [mod]* <attribute>) [(SATISFY = v)]

<linguistic constant> ::= <linguistic term> | <linguistic quantifier>

<op_e> ::= <normal comparison operator> | <fuzzy comparison operator>

Example Queries

R1 (Product, Year, Store, F-Sales, F-SalesDegree)
R2 (Q-Product, Year, Store, Q-F-Sales, Q-SalesDegree)

- **Q1**: *Find 3 top most products whose sales in the east were good in 1998.*

```
SELECT      3  Product
FROM        R1
WHERE       Store="east"  and Year=1998  and F-Sales="good"
ORDER BY    F-SalesDegree  DESC
```

- **Q2**: *Find years in which most products' sales were bad.*

```
SELECT     Year
FROM       R2
WHERE      Q-Product="most"  and  F-Sales="bad"
```

Example Queries (cont.)

- **Q3**: *Find products which had similar sales performance in the past two years (with satisfaction degree 0.8).*

  SELECT Product
  FROM R1 r_1, r_2
  WHERE r_1.Product= r_2.Product and r_1.Year=1997 and r_2.Year=1998
  and r_1.Product ***close-to*** r_2.Product (**SATISFY** = 0.8)

- **Q4**: *Find years in which PC sales was very very bad.*

  SELECT Year
  FROM R1
  WHERE Product="PC" and F-Sales = "very very" "bad"

- Use of fuzzy technology to provide semantics to data warehouses
- Proposed a three-layered data summarization levels
- Extended SQL to facilitate users' queries.

Summary

- Data warehousing systems provide efficient and effective business decision support.
- The work conducted includes
  - materialized view selection, warehouse maintenance, query language and processing, multidimensional models, physical warehouse design, etc.
- New aspects have to be considered to build next generation systems
  - dynamic warehouse design, object-orientation, multimedia technology, semantics, etc.

Question & Answer