



Data-Warehouse-, Data-Mining- und OLAP-Technologien

Chapter 1: Introduction

Bernhard Mitschang
Universität Stuttgart

Winter Term 2015/2016

Overview

- Motivation
 - Retail Scenario
 - Heterogeneous Data Sources
 - The Basic Idea
- Operational vs. Analytical Systems
- Fields of Application
- Information Services
- Other Approaches
- Overview Chapters 2-7

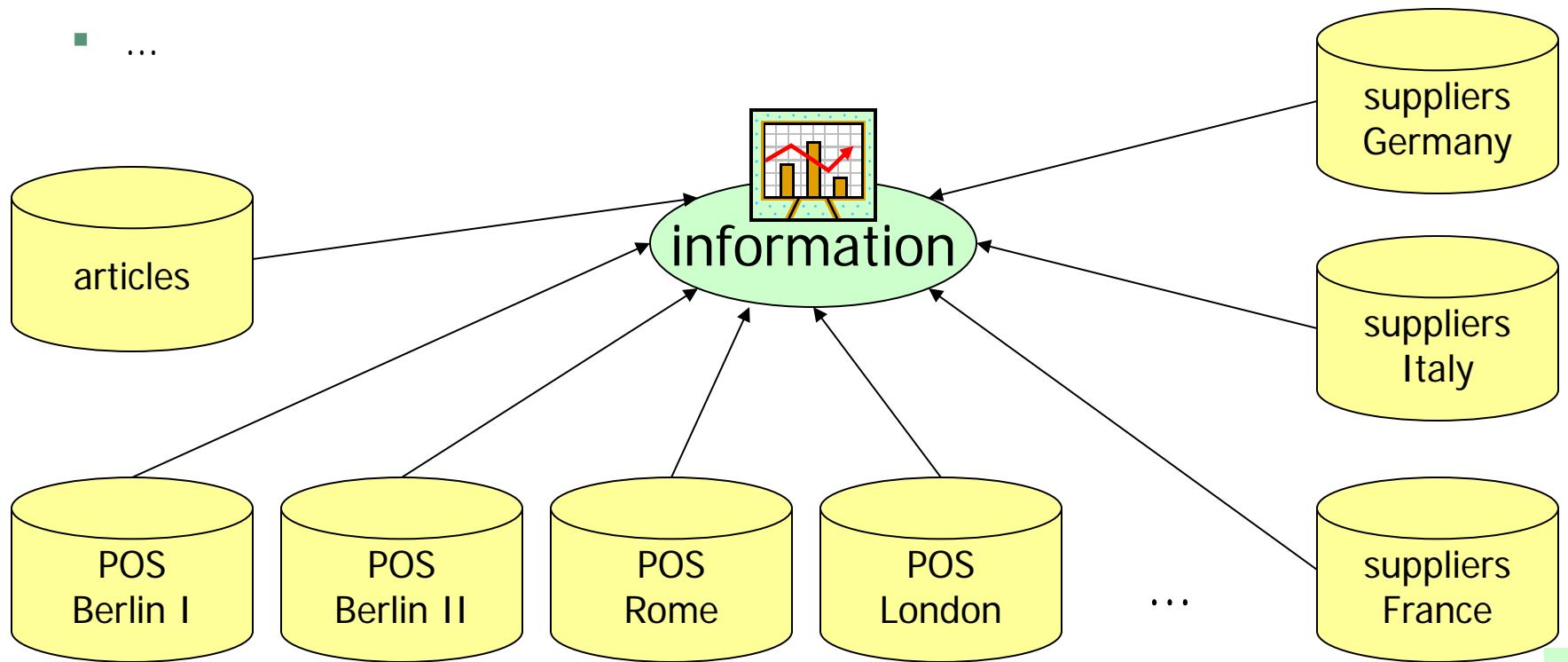
Retail Scenario

- Retail company runs many department stores in Europe (Berlin, London, Paris, Rome, ...).
- The goods come from global suppliers.
- Business people (marketing, inventory management, ...) need to know:
 - What are the gross sales for an article in each country?
 - What are the gross sales for an article per month?
 - What are the gross sales for an article per quarter?
 - What is the sales-per-square-meter of an article?
 - Did our last sales promotion push the gross sales of an article?
 - Which articles are typically sold together?
 - Which customer groups are most likely interested in a special sales promotion?
 - ...



Heterogeneous Data Sources

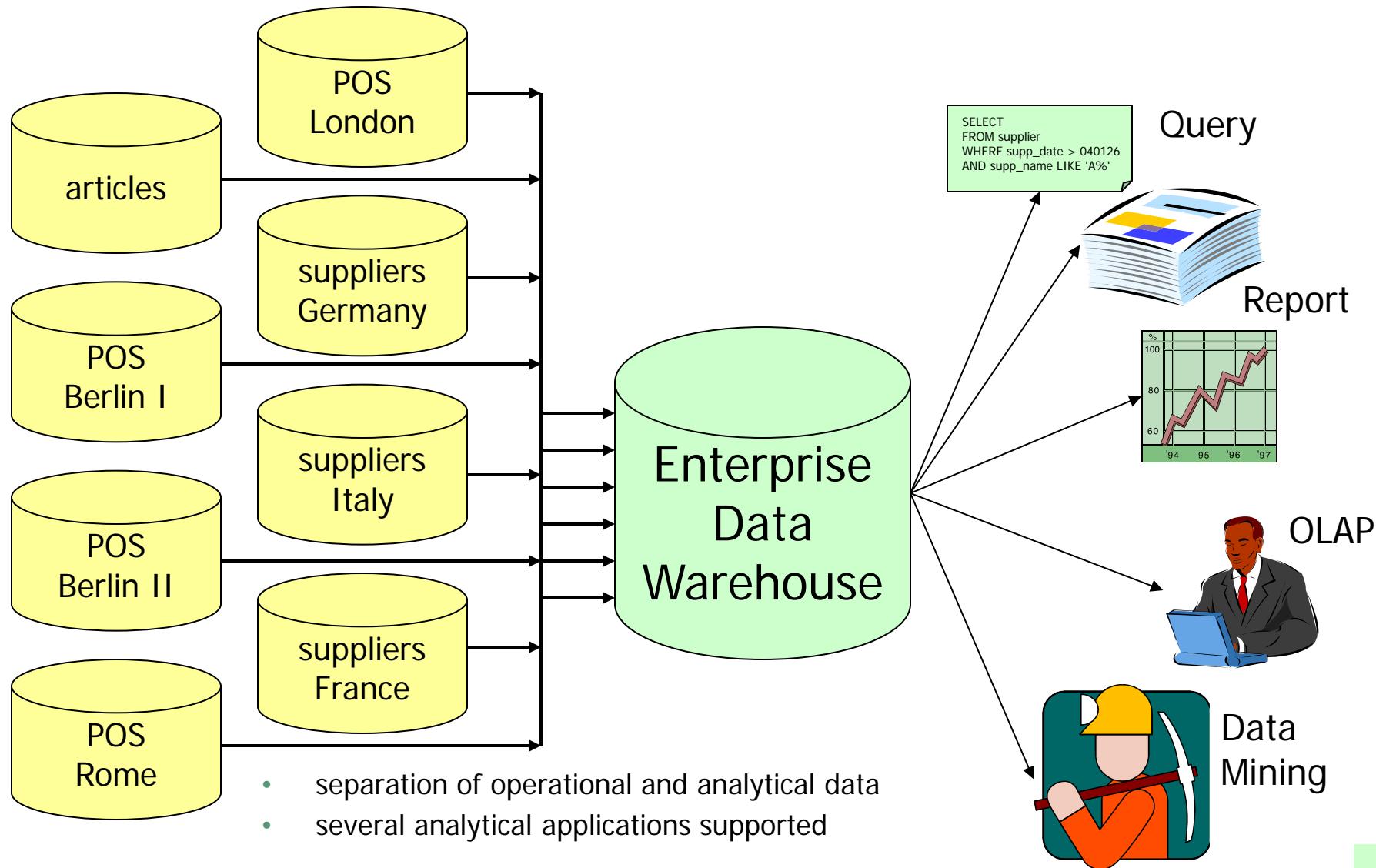
- The retail company runs several database systems that capture all the operational data:
 - one point-of-sale (POS) database per department store
 - one supplier database per country
 - one central article database
 - ...



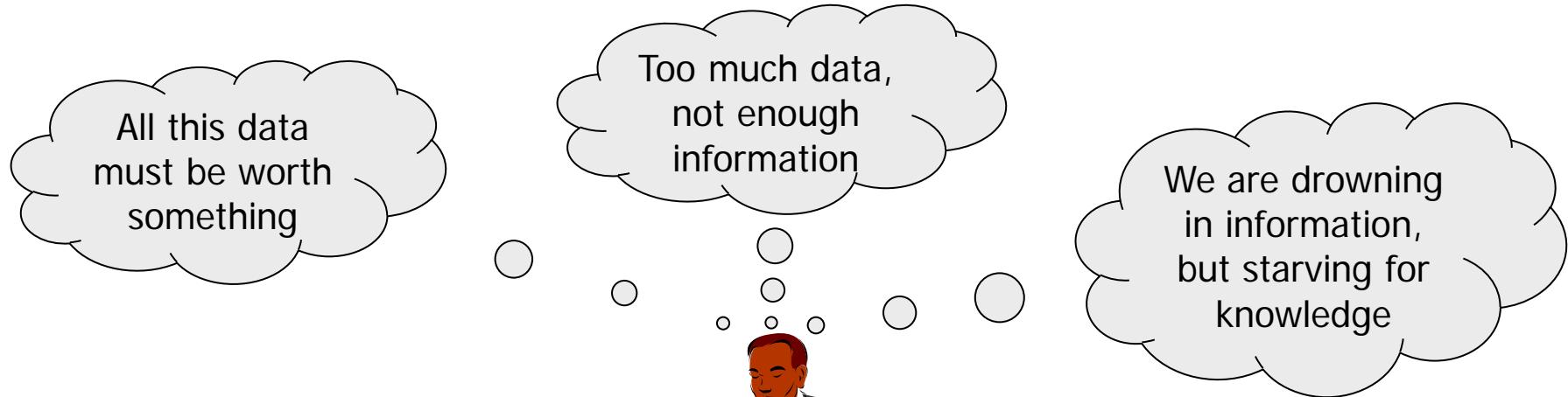
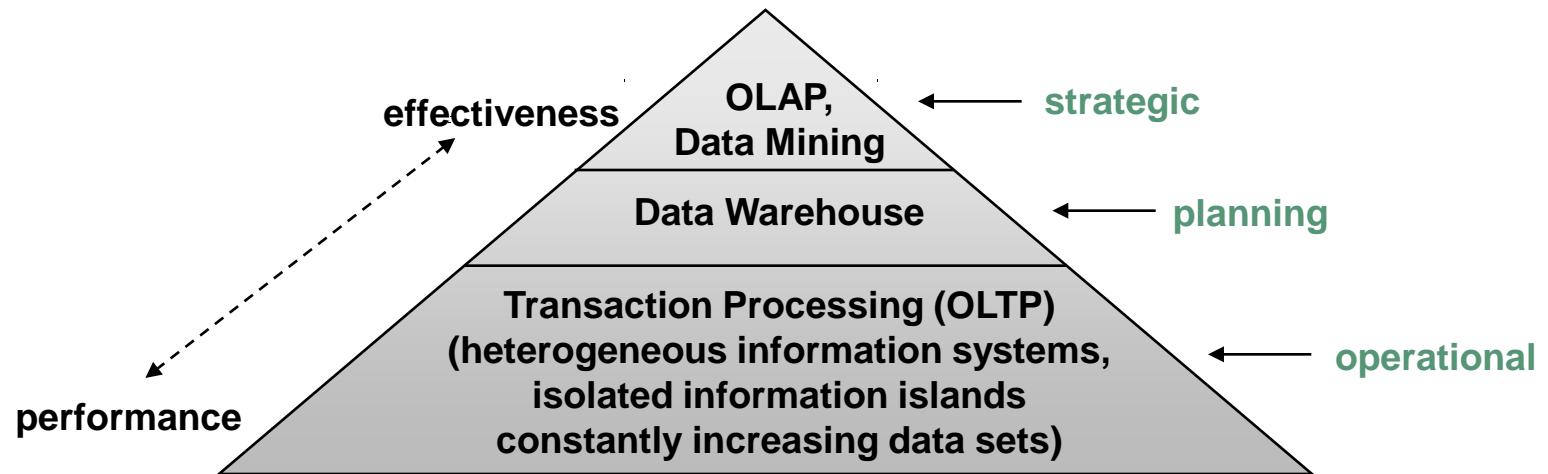
Heterogeneous Data Sources

- Broad range of data sources:
 - Database systems (relational, object-relational, hierarchical, XML, ...)
 - Enterprise Resource Planning Systems
 - External information sources (other companies, surveys, ...)
 - Files of standard applications (Excel, ...)
 - Other documents (Word, WWW, ...)
- Data Sources may differ in:
 - schema
 - coding schemes
 - time frame
 - treatment of history
 - aggregation level
 - database management system (data model, vendor, version, ...)

Data Warehouse: The Basic Idea



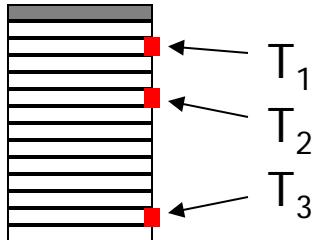
Information Pyramid



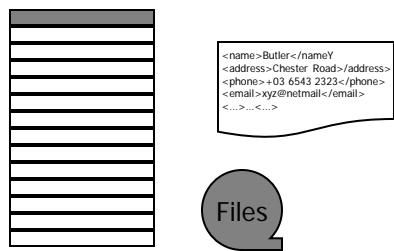
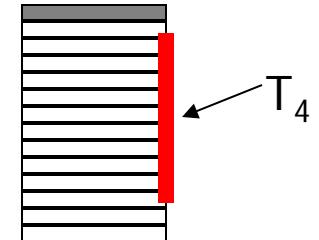
Operational vs. Analytical Systems

	operational	analytical
queries/transactions	focus	read, write, update, delete
	transaction	short read/write transaction
	queries	simple
	records per transaction	tens
	data model	flat relation
data	data sources	mainly one
	properties	current, up-to-date, detailed, isolated
	volume	MB - GB
	data access	single rows
users	user type	clerk, IT professional
	number of	$> 10^3$
	response time	ms - s
	metric	transaction throughput
		knowledge worker, manager
		> 100
		s - min
		query throughput, response time

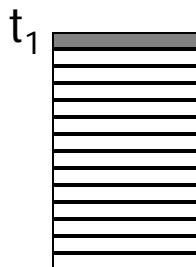
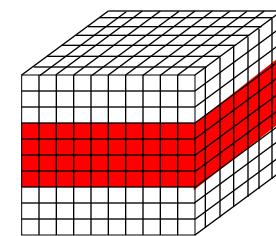
Operational vs. Analytical Systems



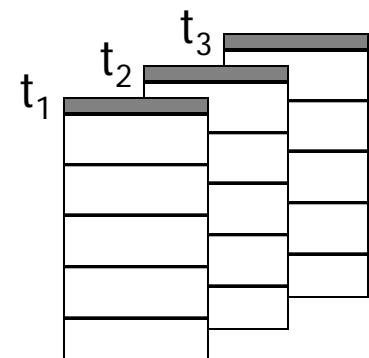
performance and safety considerations



logical interoperability problems



temporal and granularity mismatch



Fields of Application

- Business Management
 - Marketing Management
 - Enterprise Information Portals
 - One-to-one-Marketing, Electronic Customer Care, Customer Relationship Management (CRM)
 - Electronic Procurement, Supply Chain Management
- Scientific Applications
 - empirical studies, e.g. in earth observation, environmental studies
 - Statistical and Scientific Databases (SSDB)
- Engineering Applications
 - analyzing the reliability of products
 - analyzing substances and material

Introduction to Services

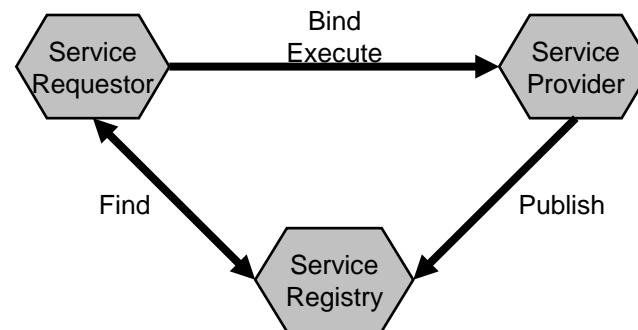
Definitions of 'Service'

- A **repeatable business task** – e.g., check customer credit; open new account
- Discretely defined **set of** contiguous and autonomous business or technical **functionality**
- A **function** provided at a network address; available via **various transports, formats, and QoS**; “always on”
- A mechanism to enable **access** to capabilities, which is provided using a prescribed **interface** and is exercised consistent with **constraints and policies** as specified by the **service description**

Introduction to SOA

Definitions of 'Service-oriented Architecture'

- An **IT architectural style** that supports integrating your **business as linked services**
- A **method for systems development and integration** where functionality is grouped around **business processes** and packaged as **interoperable services**
- An **architectural style** to realize service-oriented computing
- A **paradigm** for organizing and utilizing **distributed capabilities** that may be under the control of **different ownership domains**



Overview of Information Services

- **Service-oriented Architecture:**
→ Integrierter Ansatz zur Erneuerung der Geschäftsprozesse
(Software AG)
- "*You will waste your investment in SOA unless you have enterprise information that SOA can exploit.*" (Gartner, 2005)
→ **Information is a Vital Component of SOA Strategy**
- **Information Service:**
Equals: Data Storage + Retrieval (and crunching)
Definition: Every capability needed to understand, cleanse, integrate and deliver information across heterogeneous systems
Goal: services that provide accurate, consistent, integrated information to business processes and people
Starting point: existing legacy, inconsistent & diverse data
Approach: Information as a Service
 - Exposing application logic as services
 - Data is only accessible through the corresponding application logic
 - If needed: Mediation (brokering) and orchestration (workflow) of application logic

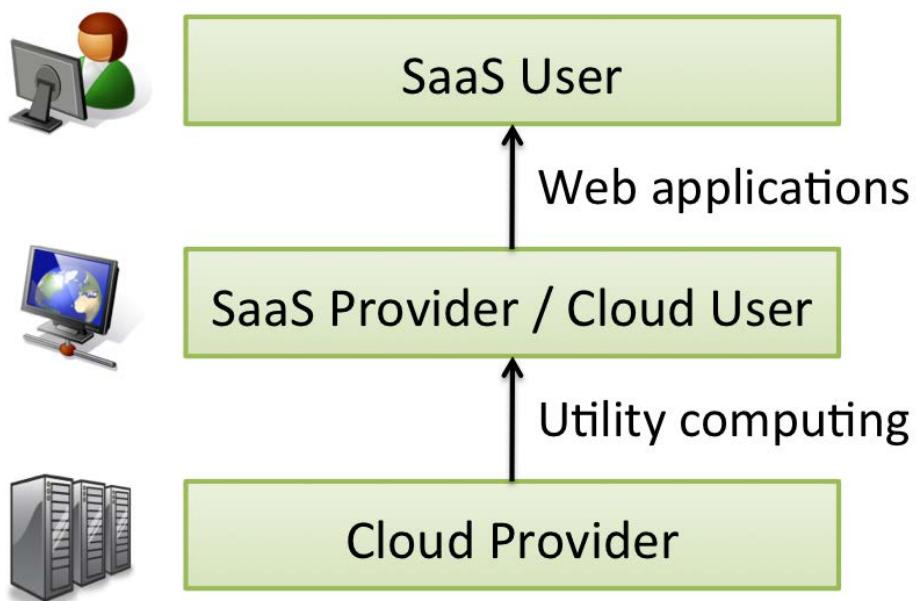
Classification of Information Services

- Classification Criteria:
 - Level of abstraction:
 - Plain source data (SQL, ...) or meta data
 - Integrated data (data/information integration, ETL, ..., MDM)
 - Interpreted data (BI, ..., text analytics, content extraction)
 - Application data (CRM, ...)
 - Range
 - Intranet
 - Internet
 - Source
 - Single/multiple
 - Homogeneous/heterogeneous
 - Provisioning of the service
 - Proprietary service
 - Web/SOA service
 - ...

Cloud Computing: One Big Picture

Gartner: "Cloud Computing is a style of computing where **massive scalable IT-enabled capabilities** are delivered **as a service** to external **customers** using Internet technologies."

... others ...: Cloud Computing isn't new. We have hosted apps for years. But now also custom-developed applications can be hosted in the cloud.



■ A Service

- Business Properties
 - No startup cost
 - Pay per use (pay as you go): costs grow linearly
- Technical Properties
 - Simple and easy to use programming model
 - Scalability and reliability as well as administration (for free) → Cloud
 - Minimum TCO → Autonomic/Cloud, Multi-tenancy

What is Missing?

1. Good and experienced Service Engineers
2. Good and experienced Cloud Engineers
3. Best Practices experiences

To 1.: International Master in Service Engineering (IMSE)

To 2.: International Master in Service Engineering (IMSE)

To 3.: International Master in Service Engineering (IMSE)

Other Approaches

1960s	1970s	1980s
Management Information Systems (MIS)		Decision Support Systems (DSS)
Executive Information Systems (EIS)		Management Decision Systems
Executive Support Systems		Strategic Planning Systems

- Goals
 - Focused on providing managers with structured, periodic reports.
 - Integrated man/machine system for providing information to support operations, management, and decision-making functions in an organization.
 - Interactive information systems that use data and models to help managers analyze semi-structured problems.
- Why they failed:
 - lacking fast communication networks
 - lacking large, cheap, and fast data storage
 - lacking cheap, high-performance CPUs
 - lacking flexibility
 - lacking productivity
 - lacking trustworthiness

Defining 'Data Warehouse'

- A data warehouse is a subject oriented, integrated, non-volatile, and time variant collection of data in support of management's decisions.
(Source: [Inm05])
- A data warehouse is a copy of transaction data specifically structured for query and analysis.
(Source: [Kim96])
- Ein Data Warehouse ist ein physischer Datenbestand, der eine integrierte Sicht auf die zugrunde liegenden Datenquellen ermöglicht.
(Source: [Zeh03])

Other Important Terms

- **Data Warehousing:**
 - Process of integrating data from several source systems, storing it in the data warehouse database, and using this integrated data source.
- **Business Intelligence:**
 - General term that covers technology and tools to turn the volume of data an organization collects and stores into meaningful information in order to achieve better and timelier business decisions.
- **Decision Support:**
 - More general term referring to all kinds of analyses of existing data in order to make better decisions - like data mining, online analytic processing (OLAP), simulation, scenario analyses, ...

Issues in Data Warehousing

Architecture

Data Model

Data Cleaning

Database Support

Data Warehouse Design

Data Extraction

Data Transformation

Data Aggregation

Query Optimization

Project Management

Metadata

Data Mining

Data Warehouse Refreshment

Phases

Data Warehouse Quality

Online Analytic Processing

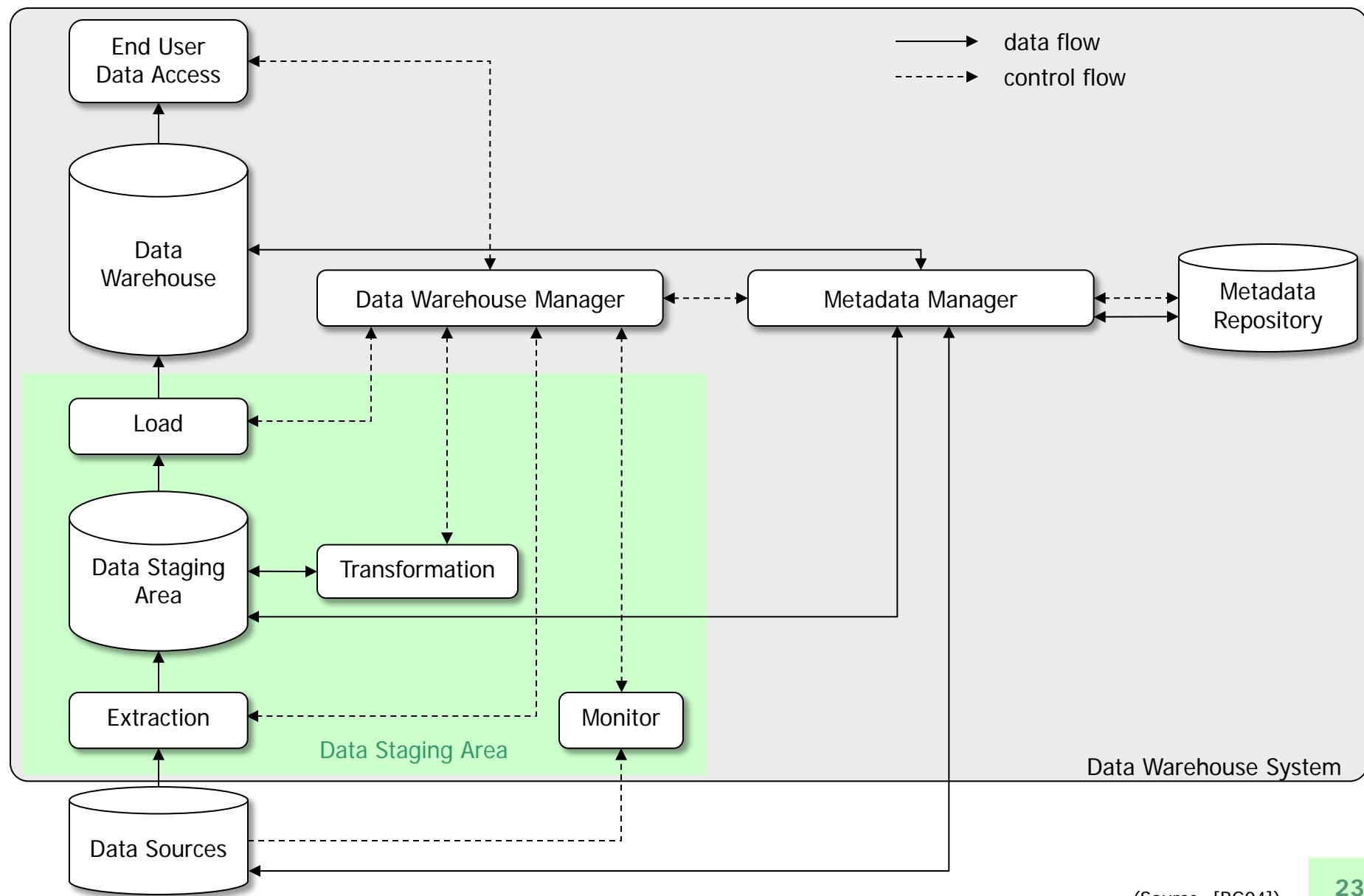
Goals of the Lecture

- Know what a data warehouse is and how it differs from other information systems
- Know the main components of a data warehouse
- Know the main processes in a data warehouse system
- Know how to design the schema of a data warehouse
- Know how OLAP and data mining is used to gain information from data warehouse data
- Know database technology that is adjusted to the needs of a data warehouse system

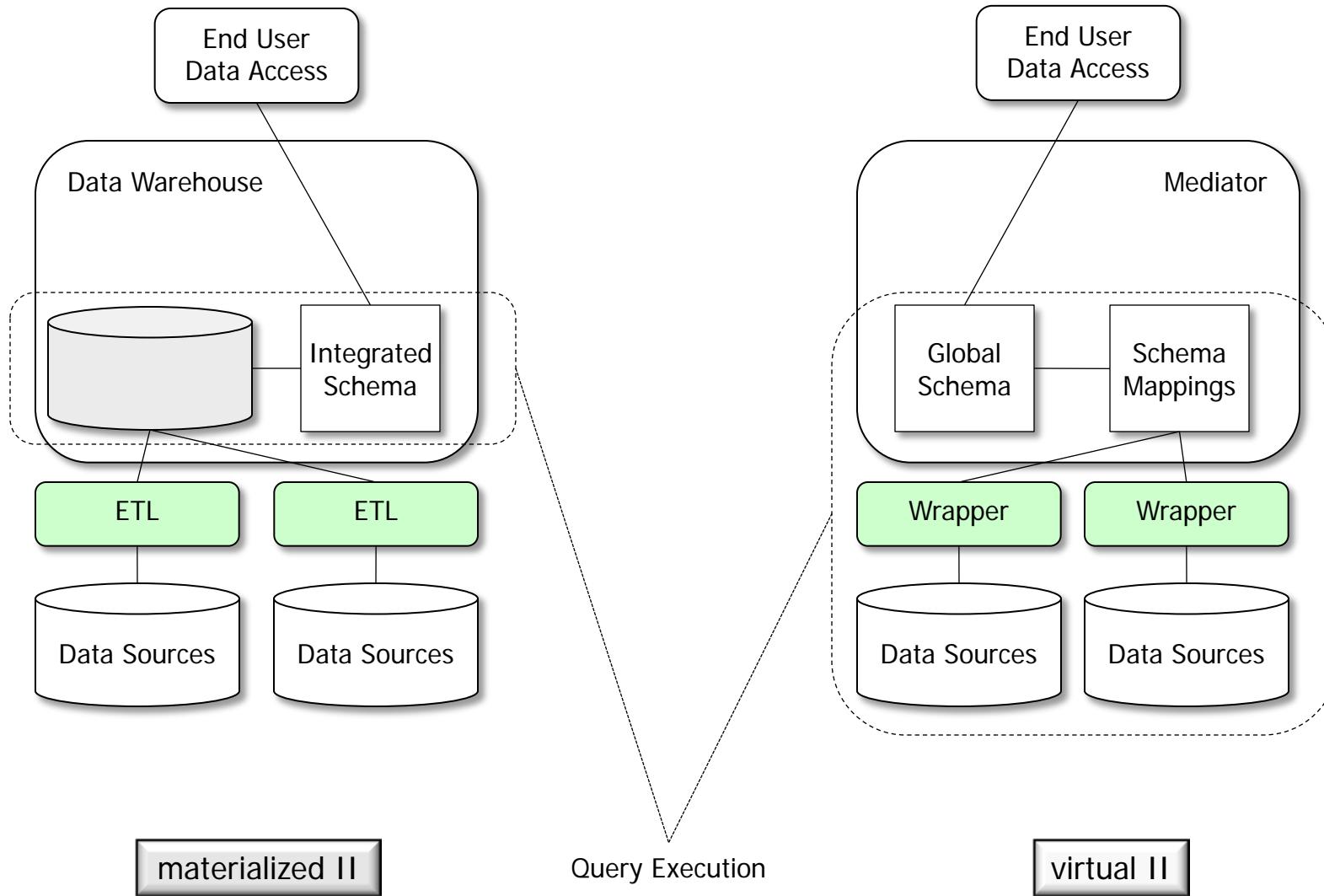
Overview

- Motivation
 - Retail Scenario
 - Heterogeneous Data Sources
 - The Basic Idea
- Operational vs. Analytical Systems
- Fields of Application
- Other Approaches
- Overview Chapters 2-7
 - Data Warehouse Architecture
 - Data Warehouse Design
 - Extraction, Transformation, Load
 - OLAP
 - Data Mining
 - Database Support

Architecture



Information Integration



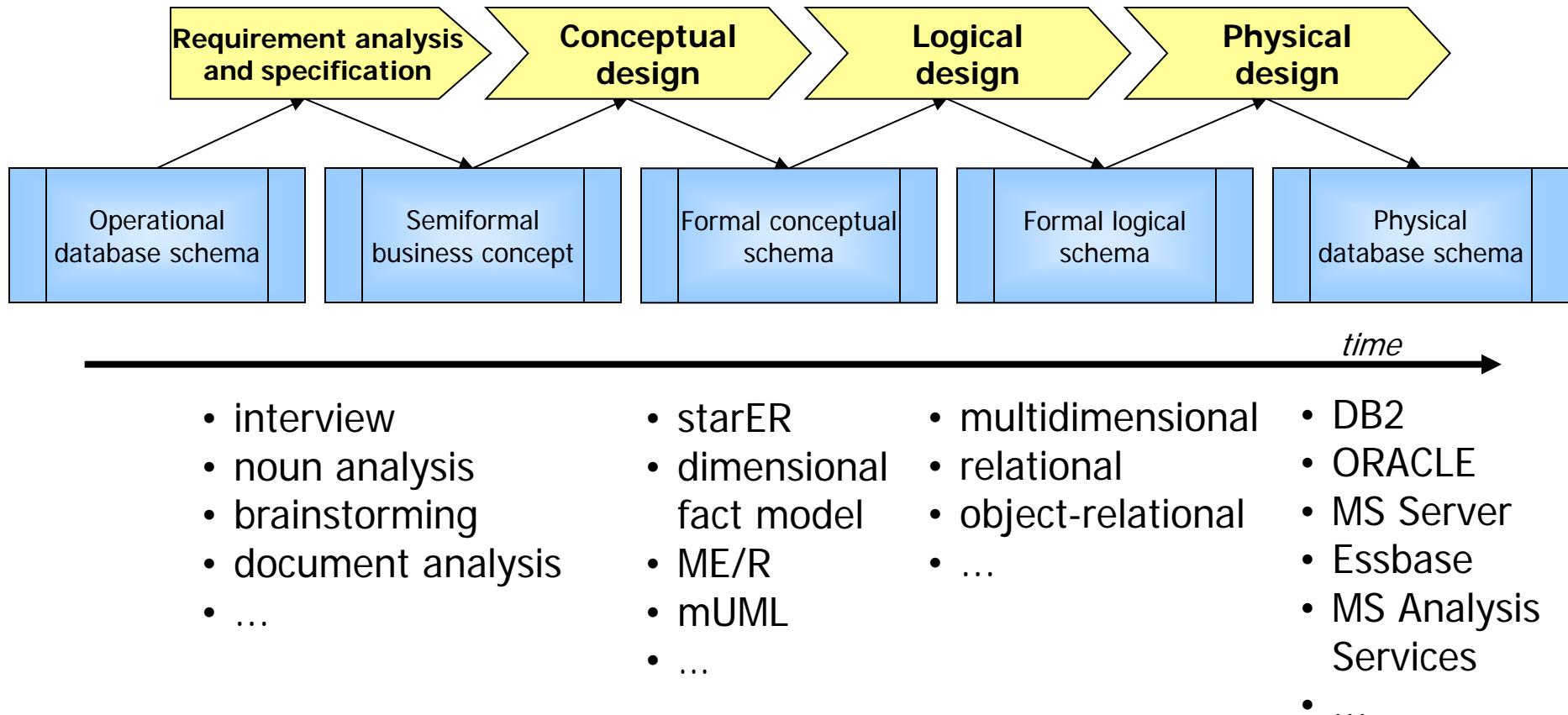
Metadata in Data Warehousing

- What data is available in the warehouse and where is the data located?
- **Data dictionary**: Definitions of the databases and relationship between data elements
- **Data flow**: Direction and frequency of data feed
- **Data transformation**: Transformations required when data is moved
- **Version control**: Changes to metadata are stored
- **Data usage statistics**: A profile of data in the warehouse
- **Alias information**: Alias names for a field
- **Security**: Who is allowed to access the data

Chapter 2: Data Warehouse Architecture

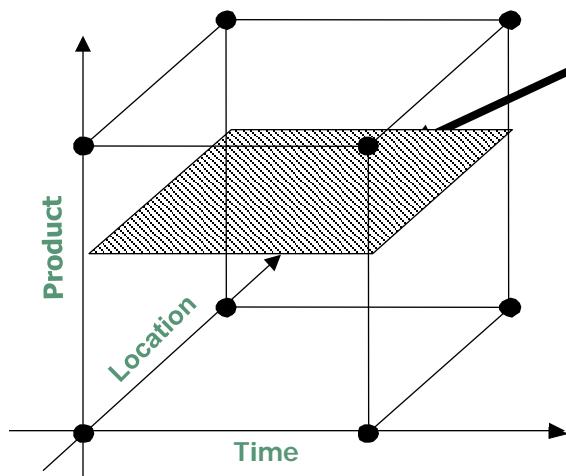
- Data Warehouse Architecture
 - Data Sources and Data Quality
 - Data Mart
 - Operational Data Store
- Information Integration
 - materialized vs. virtual
 - Federated Information Systems
- Metadata
 - Metadata Repository
 - Metadata in Data Warehousing
 - CWM Metamodel

Data Warehouse Design Process

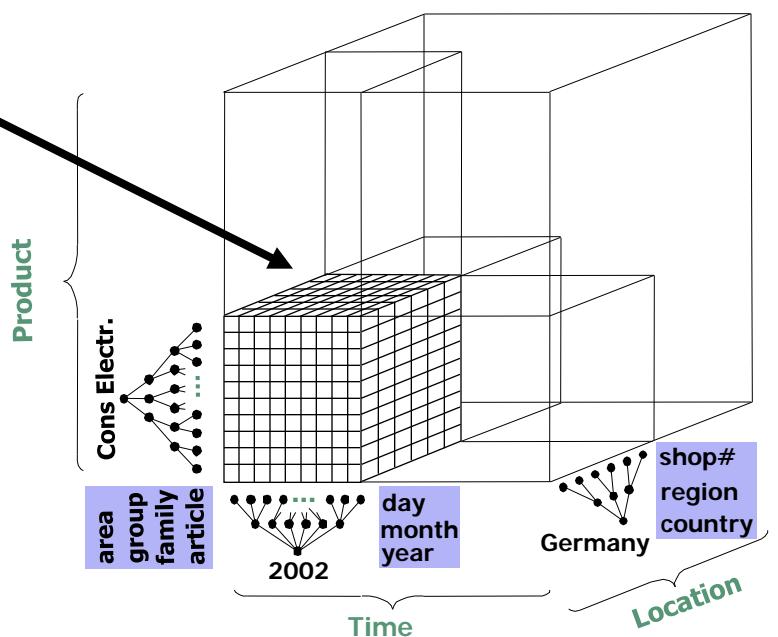


Data Warehouse Design

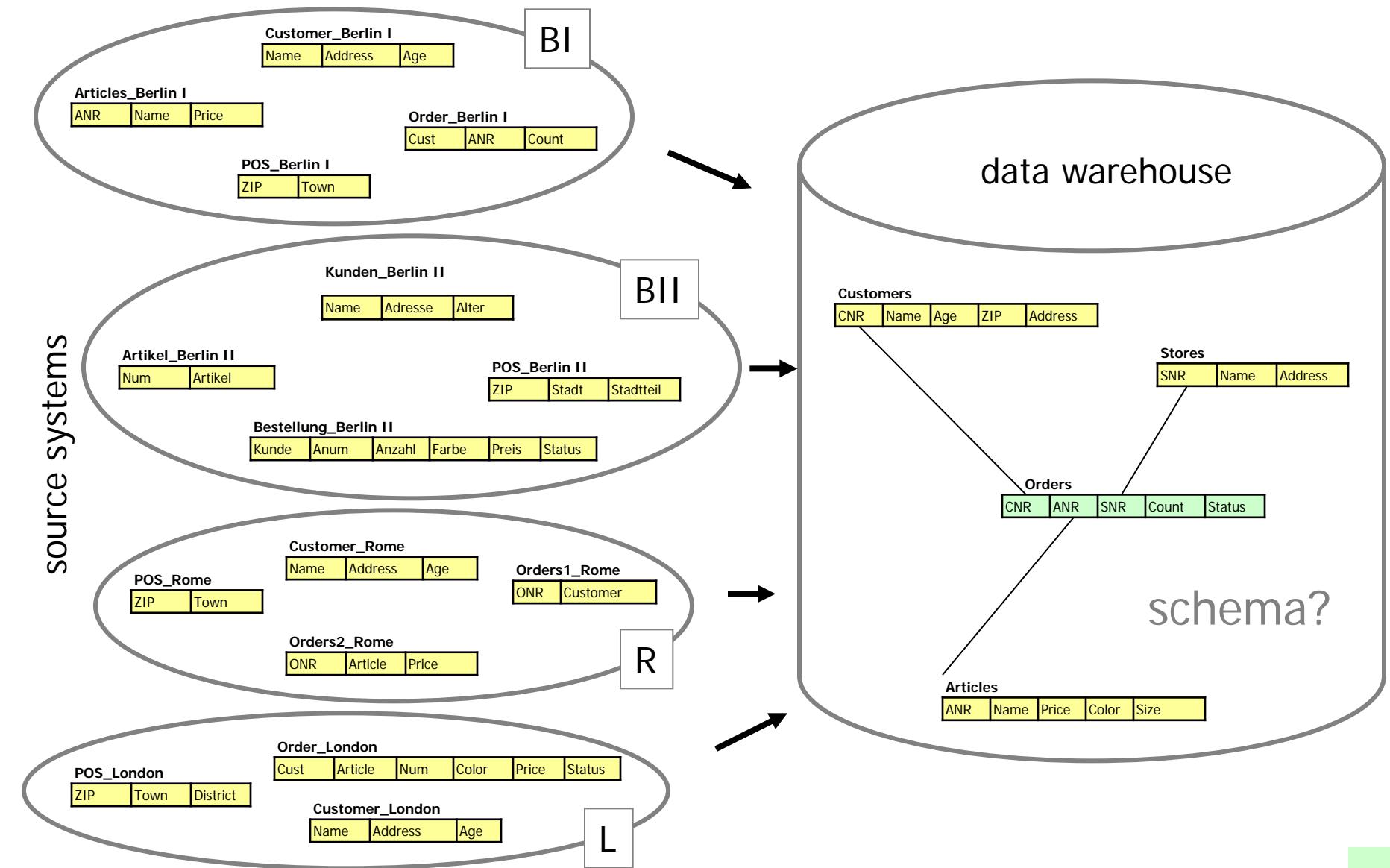
Multidimensional Model



„Cube“ Metaphor



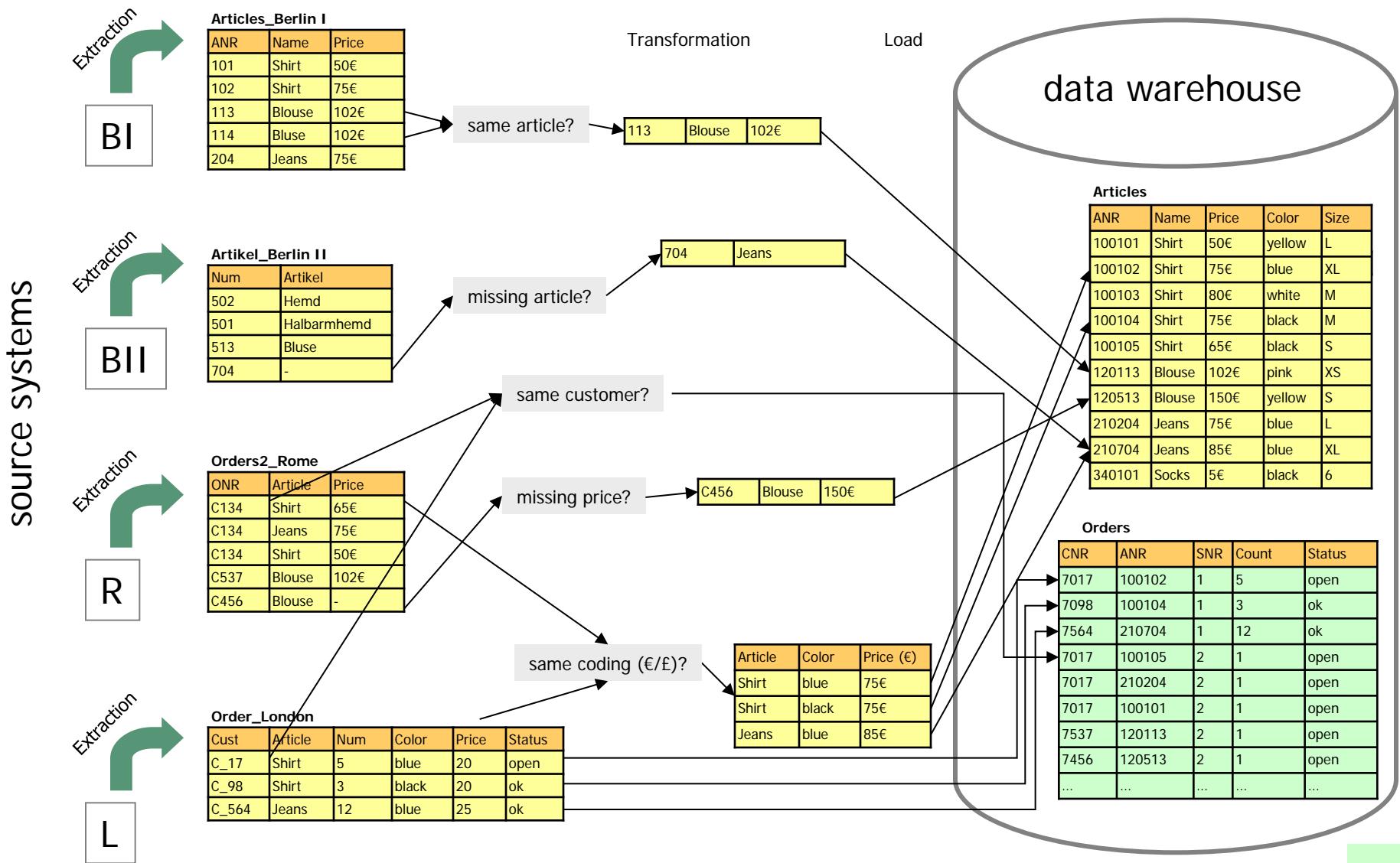
Data Warehouse Design



Chapter 3: Data Warehouse Design

- Data Warehouse Design Process
- Conceptual Design
 - Multidimensional Model
 - Dimensional Fact Model, UML-based, starER...
- Logical Design
 - Star Schema
 - Snowflake Schema
 - ...
- Dimensional Modeling
 - Extended Dimension Table Design
 - Slowly Changing Dimensions, Large Dimensions with Frequent Changes, ...
 - Extended Fact Table Design
 - Modeling Events and Coverage, Factless Fact Table, ...
- Physical Design

Extraction, Transformation, Load



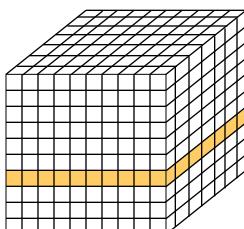
Chapter 4: Extraction, Transformation, Load

- Monitoring
- Extraction
 - Export, Import, Filter, Load
 - Direct Integration
- Load
 - Bulk Load
 - Replication
 - Materialized Views
- Transformation
 - Schema Integration
 - Data Integration
 - Data Cleansing
- Tools

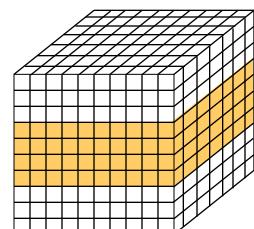
OLAP

- Online Analytic Processing
- Technologies and tools that support (ad-hoc) analysis of multi-dimensionally aggregated data
- Individual analysis is supported, i.e., the user is not restricted to available standard reports/analysis
- Graphical user interface is available for analysis specification
- Knowledge of a query language or programming language is not required
- Result information is given graphically and made available for incorporation into other applications
- Users: Analysts, Manager, "knowledge worker"
- Typical Analysis scenarios:
 - Multi-dimensional views, e.g. turnover per product group and month
 - Comparisons, e.g. turnover in Q4 compared to that of Q3
 - Ranking, e.g. top 10 product in a certain group ranked by turnover

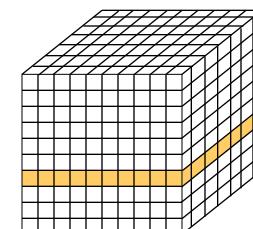
OLAP



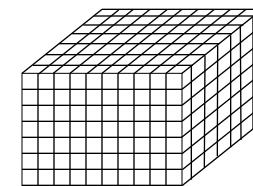
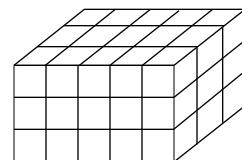
Slicing



**Slicing +
Aggregation**



Drilling



- Examples:
 - **Slicing:** Analysis of a certain product group
 - **Aggregation:** Sum of turnover for all products of a given product group
 - **Drilling:** Based on a previous analysis of a certain business area, further analysis on a level of product group and even on the level of a single product are performed

OLAP

The screenshot illustrates the MicroStrategy OLAP environment. At the top, a menu bar includes File, View, Format, Help, Back, Fwd, Home, History, Help, and Exit. The main title is "ToP-CHain - THE retail company". A sidebar on the left lists "Development of Orders", "Top Products", "Top 10 Orders per Year", and "Top 50 Orders". Below this is a bar chart titled "Current Order Sta" showing monthly order quantities from March to August. The Y-axis ranges from 0 to 25,000, and the X-axis shows months. A callout points to this chart with the text "predefined reports".

On the right, a "Templates" window lists categories like Customer Management, Inventory Management, db2vl, dbpadm, demo_templates, test_templates, and Wizard Templates (DB2VL). A "Template Editor: Price Info" window is open, showing a "Layout" tab with a grid for defining a template. The "Available components" list includes Suppliers, Parts, Orders, Customers, Shipdates, Commitdates, Receiptdates, Orderdates, and Metrics. The "Current template definition" grid has columns for Order, Orderday, Sum Endprice, Sum Orderquantity, and Price Per Unit. A callout points to this editor with the text "define individual analysis".

define individual analysis

Chapter 5: Online Analytic Processing

- Introduction
- OLAP Operations
- OLAP Characteristics
 - OLAP Product Evaluation Rules
 - FASMI Test
- Multidimensional Database Systems
 - Multidimensional Arrays
 - Sparse Cubes
 - Multidimensional Query Language
- Architecture
 - MOLAP, ROLAP, HOLAP

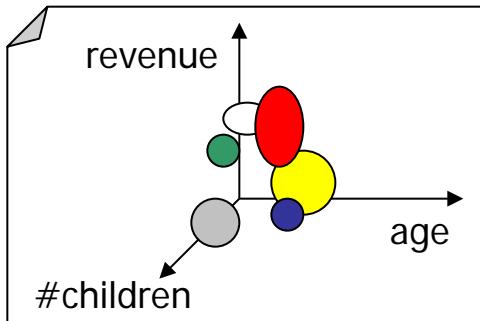
Data Mining

- Data mining is the process of discovering hidden, previously unknown and usable information from a large amount of data. The data is analyzed without any expectation on the result. Data mining delivers knowledge that can be used for a better understanding of the data.
[ISO/IEC JTC1/SC32 WG4 SQL/MM Part 6 WD, 2000]

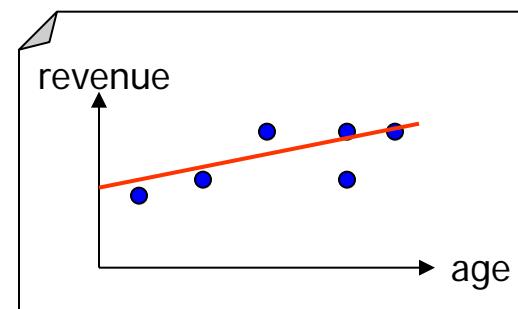
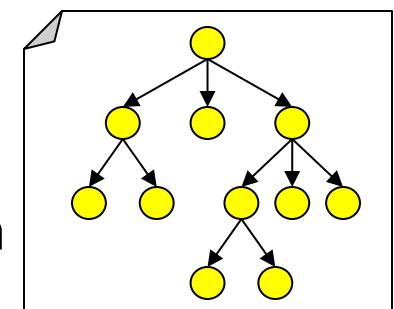
- Data Mining Techniques:
 - Association Rule Discovery

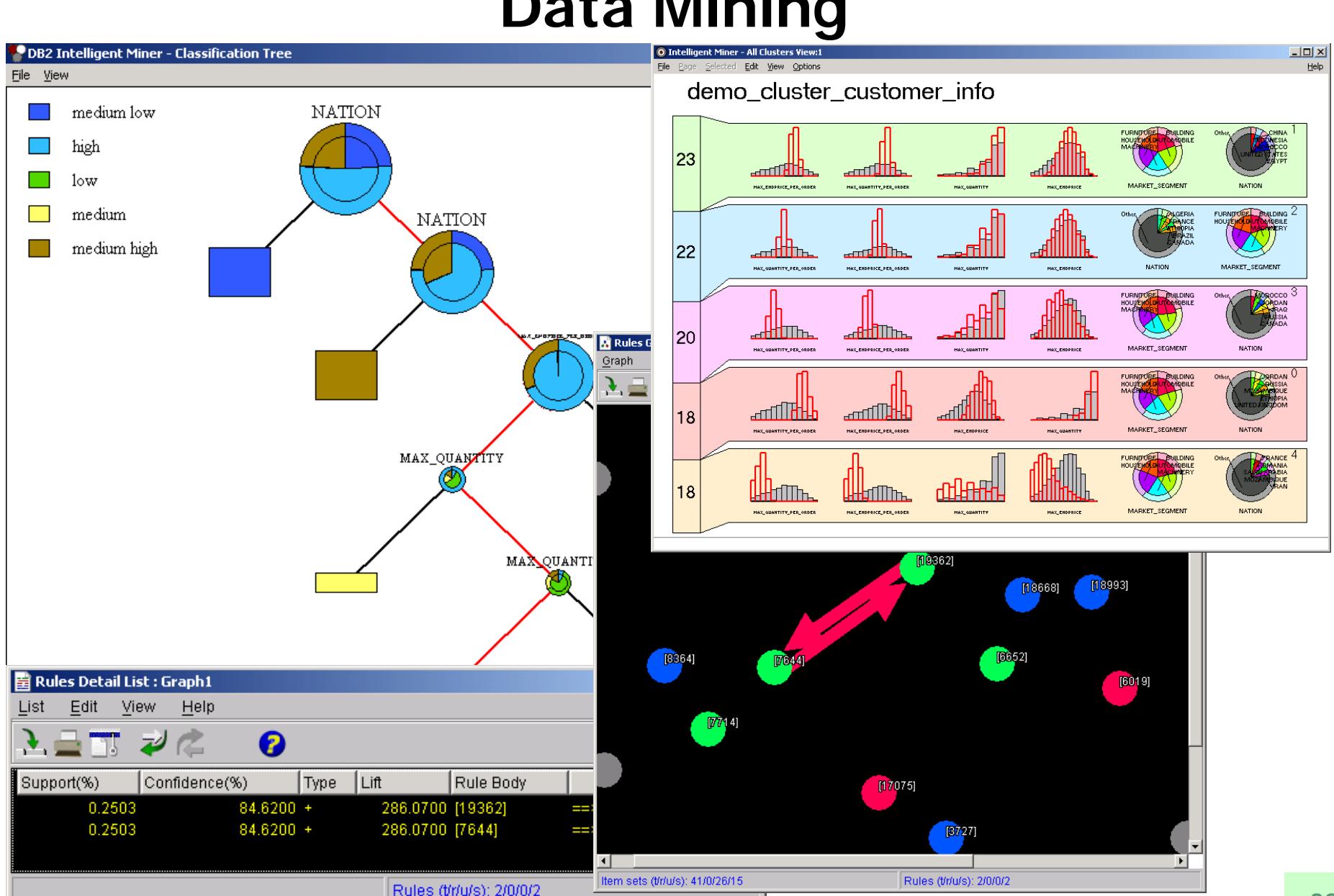
$\{\text{beer, nappies}\} \rightarrow \{\text{potato chips}\}$
support = 0.04
confidence = 0.81

- Clustering



- Classification
- Regression





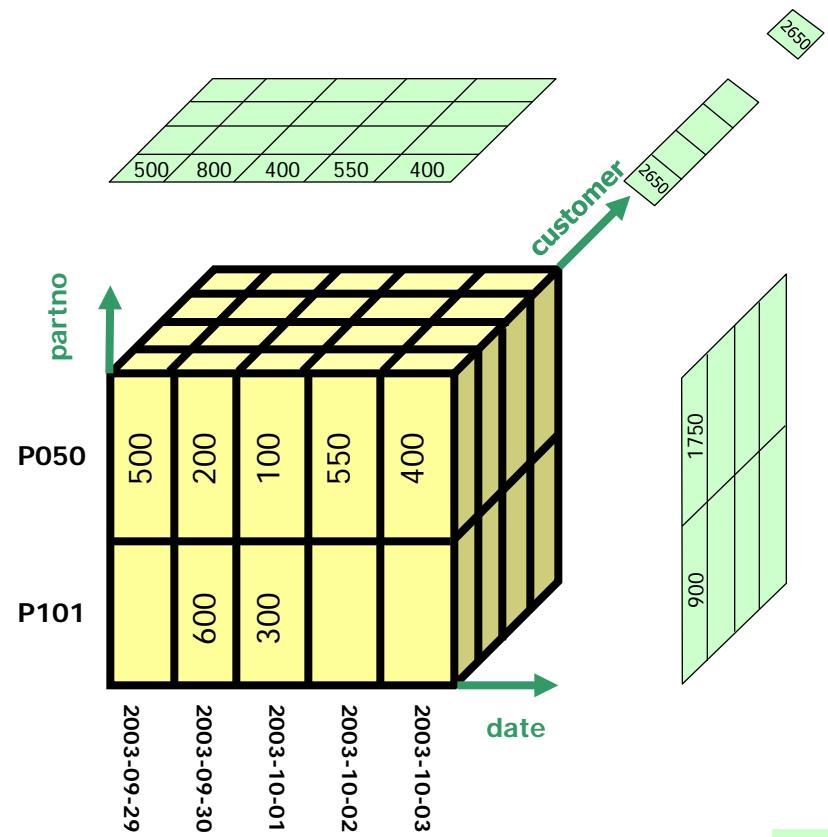
Chapter 6: Data Mining

- Introduction
 - Terms & Definitions
 - Disciplines & Applications
- Data Mining Techniques
 - Classification
 - Regression
 - Association Rule Discovery
 - Clustering
- Data Mining Systems
 - Tool
 - Trends

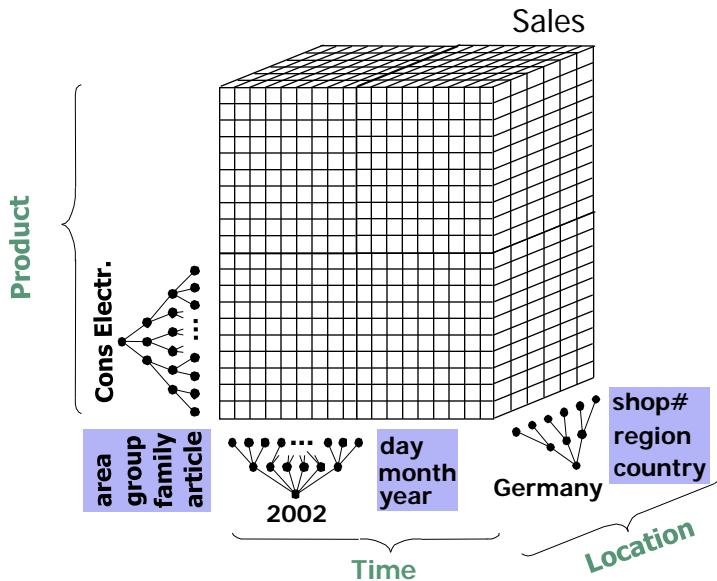
OLAP Support in SQL: 1999

- WINDOWS allow to apply aggregate functions to the current row and its neighboring rows.
- GROUP BY clause is extended by the CUBE and ROLLUP keywords which allows multidimensional summaries.

partno	date	sales	Sum(sales)
P050	2003-09-29	500	800
P050	2003-09-30	200	850
P050	2003-10-01	100	1550
P050	2003-10-02	550	950
P050	2003-10-03	400	
P101	2003-09-30	600	
P101	2003-10-01	300	



CUBE



- Support Roll-up (dimension reduction)
- Compute:
 - Sales for all aggregation groups of attributes country, month and group.

query using grouping sets

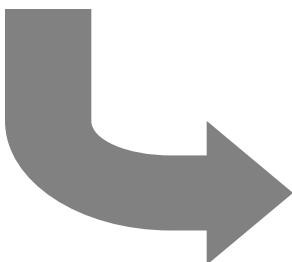
```
SELECT country, month, group, SUM(sales)
FROM Sales
GROUP BY GROUPING SETS (
  (country, month, group),
  (country, month),
  (country, group),
  (month, group),
  (country), (month), (group), () )
```

GROUP BY country, month, group
 GROUP BY country, group
 GROUP BY country, month
 GROUP BY month, group
 GROUP BY country
 GROUP BY month
 GROUP BY group

CUBE

equivalent query using CUBE

```
SELECT country, month, group, SUM(sales)
FROM Sales
GROUP BY CUBE
  (country, month, group)
```



country	month	group	sales
F	01/04	100	500
F	01/04	200	1000
F	01/04	300	250
F	01/04	-	1750
F	02/04	100	750
F	02/04	200	1250
F	02/04	300	2000
F	02/04	400	500
F	02/04	-	4500
F	-	-	6250
GB	01/04	100	400
GB	01/04	200	800
GB	01/04	300	300
GB	01/04	-	1500
GB	02/04	100	2000
GB	02/04	200	2500
GB	02/04	300	100
GB	02/04	400	100
GB	02/04	500	100
GB	02/04	600	100
GB	02/04	-	4900
GB	-	-	6400
I	01/04	200	250
I	01/04	300	750
I	01/04	400	200
I	01/04	500	1500
I	01/04	600	800
I	01/04	-	3500
I	-	-	3500
-	-	-	16150

country	month	group	sales
F	-	100	1250
F	-	200	2250
F	-	300	2250
F	-	400	500
GB	-	100	2400
GB	-	200	3300
GB	-	300	400
GB	-	400	100
GB	-	500	100
GB	-	600	100
I	-	200	250
I	-	300	750
I	-	400	200
I	-	500	1500
I	-	600	800
-	01/04	100	900
-	01/04	200	2050
-	01/04	300	1300
-	01/04	400	200
-	01/04	500	1500
-	01/04	600	800
-	02/04	100	2750
-	02/04	200	3750
-	02/04	300	2100
-	02/04	400	600
-	02/04	500	100
-	02/04	600	100
-	01/04	-	6750
-	02/04	-	9400
-	-	100	3650
-	-	200	5800
-	-	300	3400
-	-	400	800
-	-	500	1600
-	-	600	900

+

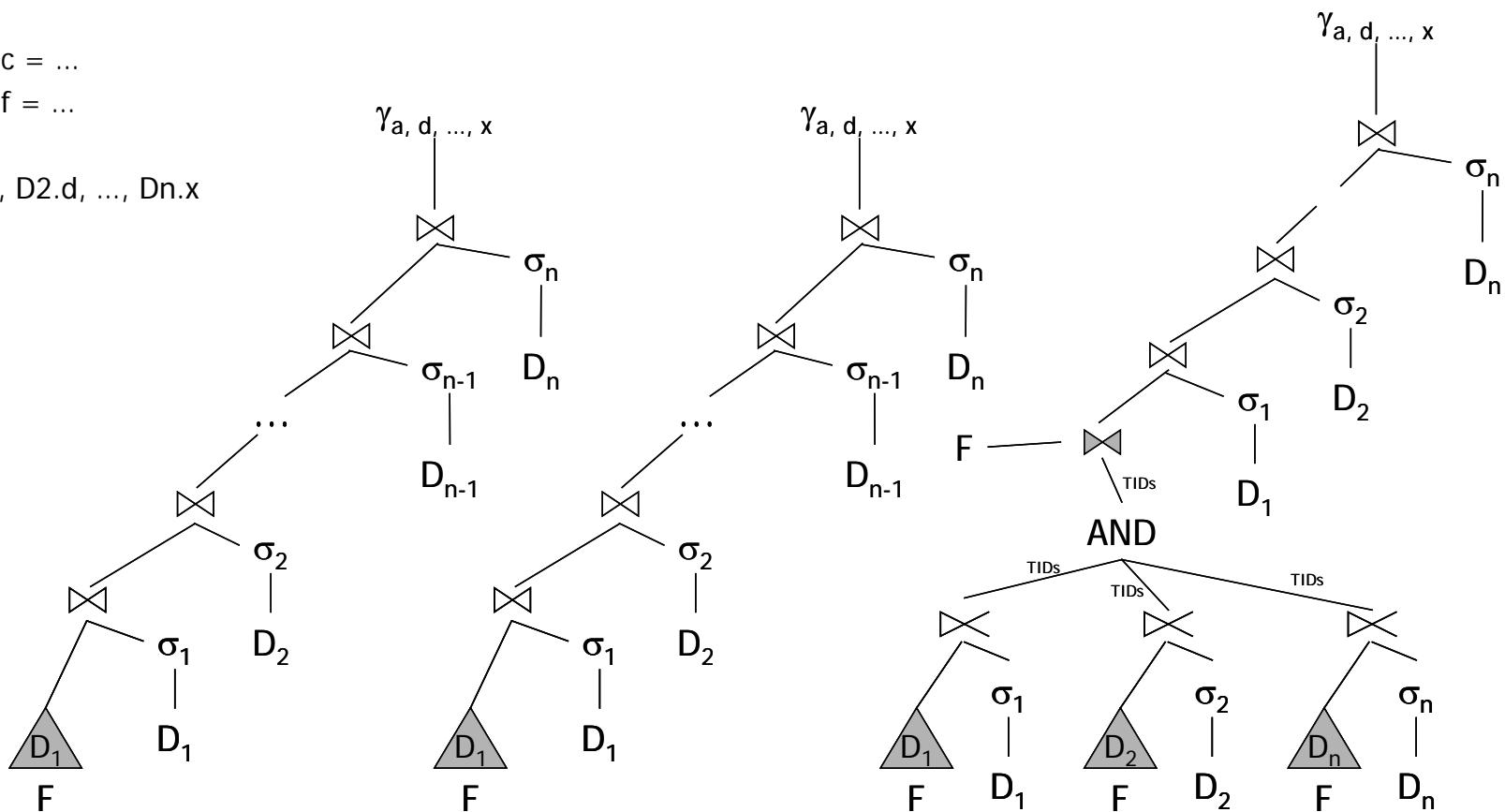
"-": NULL value

Evaluating Star Queries

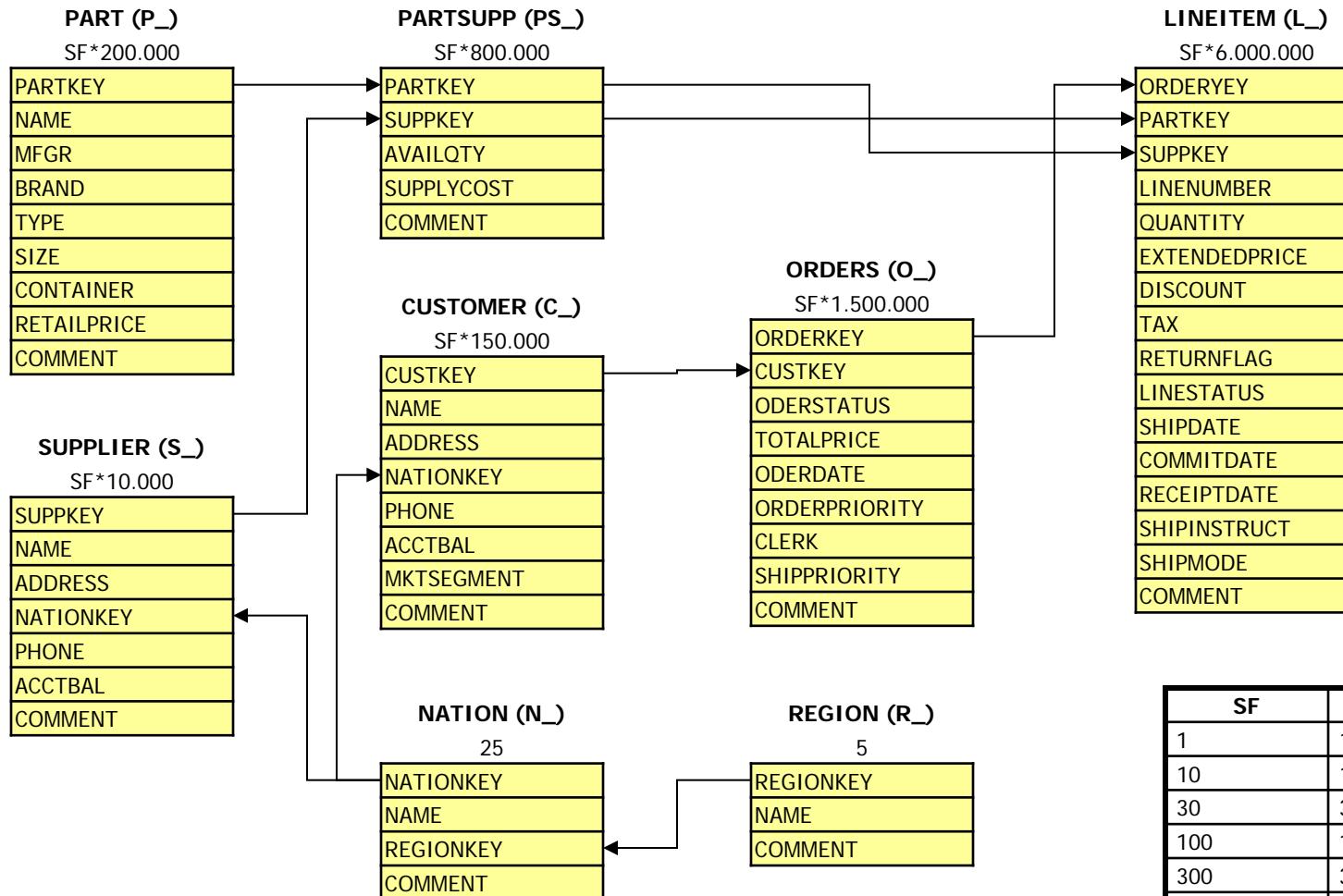
```

SELECT      D1.a, D2.d, ..., Dn.x, f(F факт)
FROM        F, D1, D2, ..., Dn
WHERE       F.b = D1.b
AND         F.e = D2.e
...
AND         D1.c = ...
AND         D2.f = ...
...
GROUP BY   D1.a, D2.d, ..., Dn.x
  
```

- Optimizer of the DBMS has to decide on join order, index usage, predicate push-down, ...
- Alternative query plans for star queries:



TPCH Benchmark Schema



Database Scaling:

SF	database size
1	1 GB
10	10 GB
30	30 GB
100	100 GB
300	300 GB
1000	1000 GB = 1 TB
3000	3000 GB
10000	10000 GB
30000	30000 GB
100000	100000 GB

Chapter 7: Database Support

- Support for OLAP and Data Mining in SQL
 - OLAP:
ROLLUP, CUBE, WINDOW, OLAP Functions
 - Data Mining (SQL/MM)
- Database Support for OLAP
 - Partitioning
 - Materialized Views
 - Indexes
 - Optimization of OLAP Queries (Star Joins)
- Performance Evaluation: TPC Benchmarks

Books

- [BG04] A. Bauer, H. Günzel: Data Warehouse Systeme. 2. Aufl., dpunkt, 2004.
 - [Len03] W. Lehner: Datenbanktechnologie für Data-Warehouse-Systeme. dpunkt, 2003.
 - [KR+98] R. Kimball, L. Reeves, M. Ross, W. Thornthwaite: The Data Warehouse Lifecycle Toolkit. Wiley, 1998.
 - [Inm05] W. H. Inmon: Building the Data Warehouse. 4th Edition, Wiley, 2005.
-
- [HK00] J. Han, M. Kamber: Data Mining – Concepts and Techniques. Morgan Kaufmann, 2nd Edition, 2006.
 - [JL+02] M. Jarke, M. Lenzerini, Y. Vassiliou, P. Vassiliadis: Fundamentals of Data Warehouses. Springer, 2002.
 - [Kim96] R. Kimball: The Data Warehouse Toolkit. Wiley, 1996
 - [LN07] U. Leser, F. Naumann: Informationsintegration, dpunkt, 2007.
 - [Wes01] P. Westerman: Data Warehousing. Morgan Kaufmann. 2001.

Papers

- [Ber98] P. Bernstein: Repositories and Object Oriented Databases, SIGMOD Record 27(1):88-96, 1998.
- [CD97] S. Chaudhuri, U. Dayal: An Overview of Data Warehousing and OLAP Technology, SIGMOD Record 26(1):65-74, 1997.
- [HLV00] B. Hüsemann, J. Lechtenbörger, G. Vossen: Conceptual Data Warehouse Design. Proc. of the Second International Workshop on Design and Management of Data Warehouses, Stockholm, 2000.
- [Zeh03] T. Zeh: Data Warehousing als Organisationskonzept des Datenmanagements. In: Informatik Forschung und Entwicklung, Band 18, Heft 1, August 2003.

Data-Warehouse-, Data-Mining- und OLAP-Technologien

Chapter 2: Data Warehouse Architecture

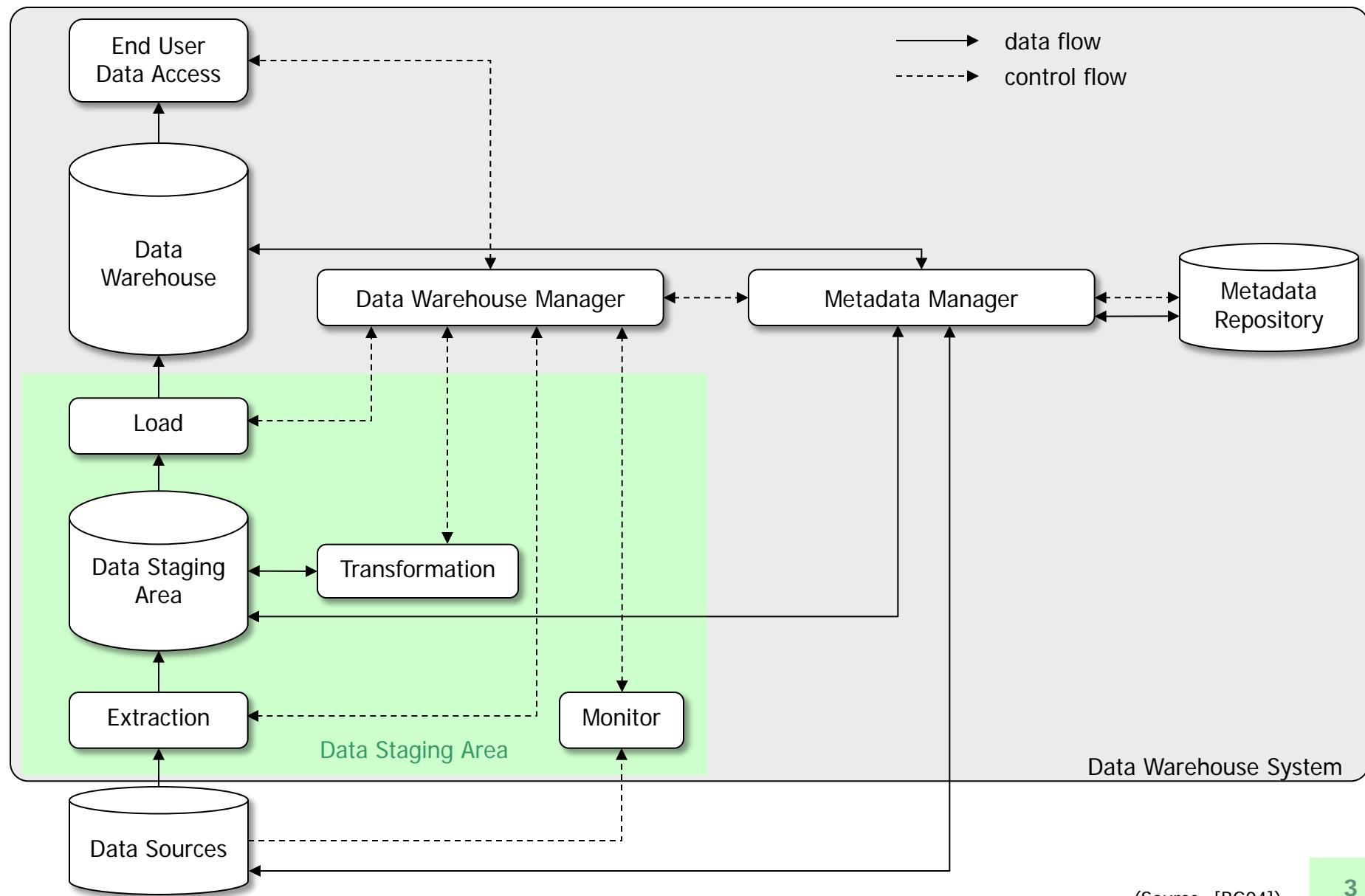
Bernhard Mitschang
Universität Stuttgart

Winter Term 2015/2016

Overview

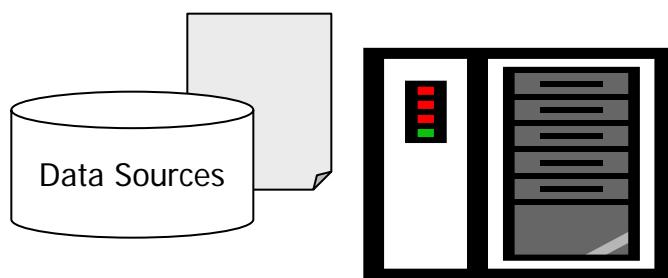
- Data Warehouse Architecture
 - Data Sources and Data Quality
 - Data Mart
 - Operational Data Store
- Information Integration
 - materialized vs. virtual
 - Federated Information Systems
- Metadata
 - Metadata Repository
 - Metadata in Data Warehousing
 - CWM Metamodel

Architecture



Data Sources

- Characteristics of source systems:
 - narrow, "account-based" queries
 - no queries in a broad and unexpected way, like DW
 - maintain little historical data
 - no conformed dimensions (product, customer, geography, ...) with other legacy systems
 - use keys (production keys) to make certain things unique (product, customer, ...)
- Important issues in selecting data sources:
 - Purpose of the data warehouse
 - Quality of data sources
 - Availability of data sources (organizational prerequisites, technical prerequisites)
 - Costs (internal data, external data)



Data Quality

consistency

- Are there contradictions in data and/or metadata?

correctness

- Do data and metadata provide an exact picture of the reality?

completeness

- Are there missing attributes or values?

exactness

- Are exact numeric values available?
- Are different objects identifiable? Homonyms?

reliability

- Is there a Standard Operating Procedure (SOP) that describes the provision of source data?

understandability

- Does a description for the data and coded values exist?

relevance

- Does the data contribute to the purpose of the data warehouse?

Dimensions of Data Sources

origin

- internal vs. external data

time

- current vs. historic data

usage

- data vs. metadata

type

- number, string, time, graphic, audio, video, ...
numeric, alphanumeric, boolean, binary, ...

character set

- ASCII, EBCDIC, UNICODE, ...

orientation

- left to right, right to left, top-down

confidentiality

- strictly confidential, confidential, public, ...

Monitoring

- Goal: Discover changes in data source incrementally
- Approaches:

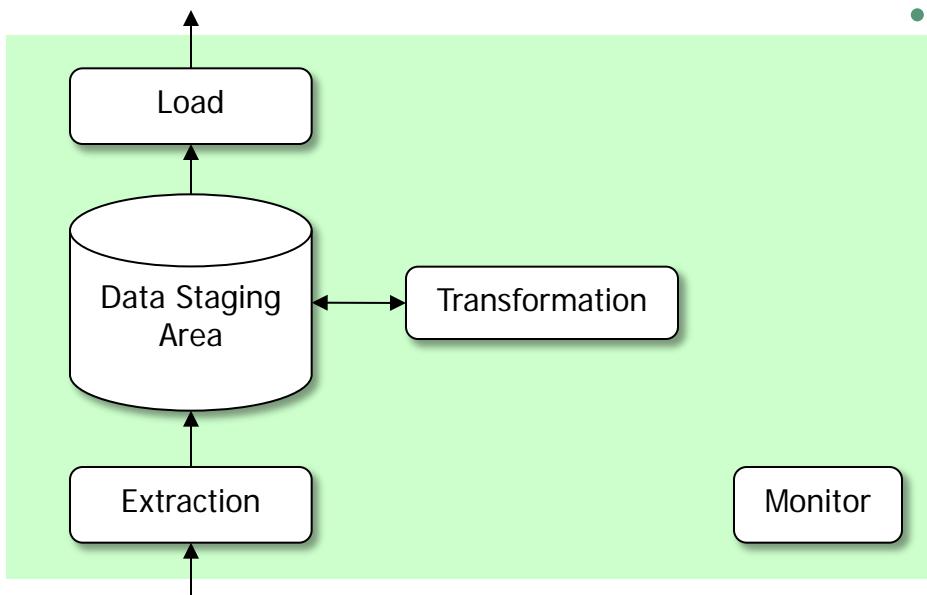
	Based on ...	Changes identified by ...
Trigger	triggers defined in source DBMS	trigger writes a copy of changed data to files
Replica	replication support of source DBMS	replication provides changed rows in a separate table
Timestamp	timestamp assigned to each row	use timestamp to identify changes (supported by temporal DBMS)
Log	log of source DBMS	read log
Snapshot	periodic snapshot of data source	compare snapshots

Data Staging Area (DSA)

Data Staging Area:

A storage area and a set of processes that clean, transform, combine, deduplicate, household, archive, and prepare source data for use in the data warehouse.

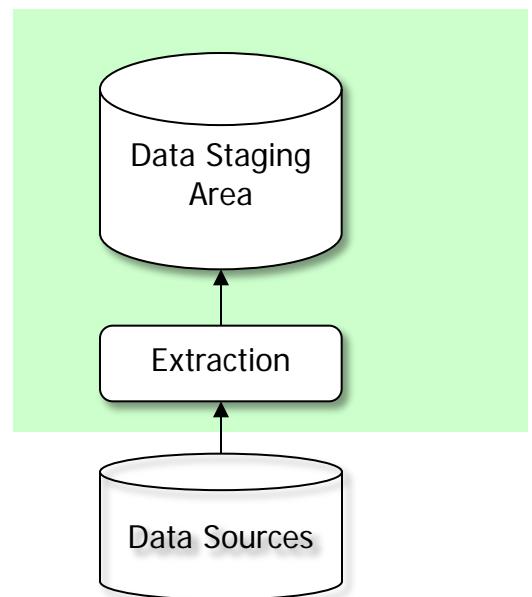
(R. Kimball et al: The Data Warehouse Lifecycle Toolkit, 1998)



- Data is temporarily stored in the data staging area before it is loaded into the data warehouse.
- All transformations are performed in the DSA.
 - Preprocessing does not influence data sources or data warehouse
- DSA is the central repository for ETL (Extraction - Transformation - Load) processing.

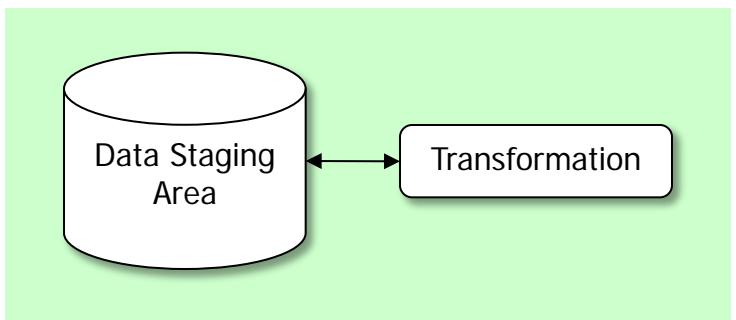
Extraction

- Transfer data from data source into the data staging area.
- Extracted subset of data sources and schedule of the extraction depends on the kind of analysis that should be supported.
- Method depends on the monitoring strategy used:
 - Read data from a file written by triggers.
 - Read data from replication tables.
 - Select data based on the timestamp.
 - Read data from log.
 - Read output of snapshot comparison.
- Multiple extract types:
 - periodic
 - started by the admin/user
 - event-driven
 - immediate after changes in data sources



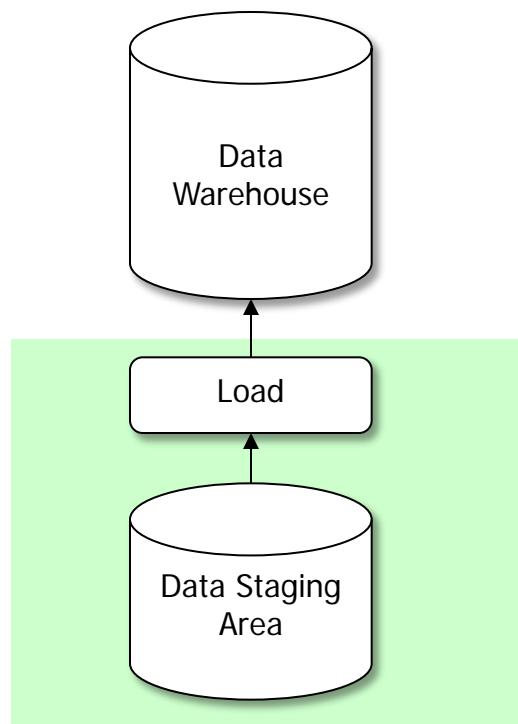
Transformation

- Convert the data into something representable to the users and valuable to the business.
 - Transformation of structure **and** content
- Typical transformations:
 - denormalization, normalization
 - data type conversion
 - calculation, aggregation
 - standardization of strings and date values
 - conversion of measures
 - cleansing (missing, wrong, and inconsistent values)



Load

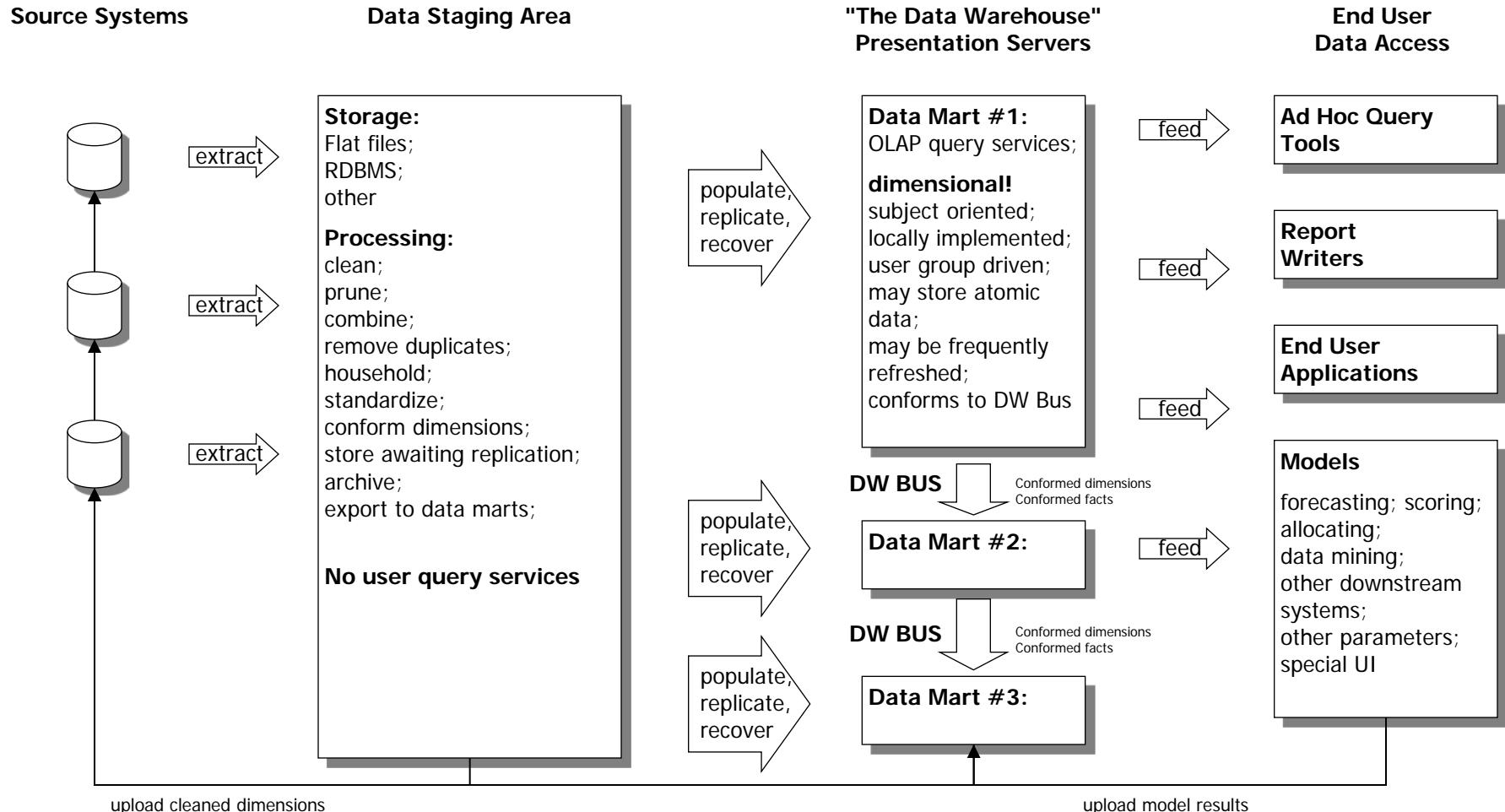
- Transfer data from the data staging area into the data warehouse.
- Data in the warehouse is rarely replaced. The history of values/changes is stored instead.
- Mainly based on bulk load tools of the DBMS.
- Offline vs. online load.
- Parallel load may be required.



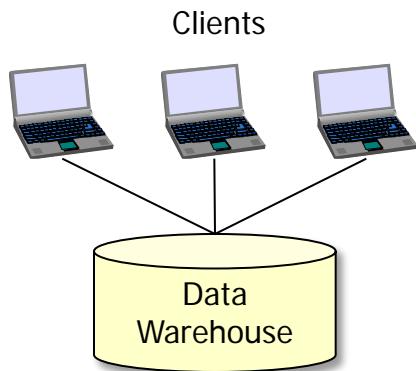
Data Warehouse Manager

- Controls all components of the data warehouse system:
 - **Monitor:** Discover changes in data sources
 - **Extraction:** Select and transfer data from data sources to the data staging area
 - **Transformation:** Consolidate data
 - **Load:** Transfer data from data staging area to the data warehouse
 - **End User Data Access:** Analysis of data in the data warehouse

Basic Elements of the Data Warehouse

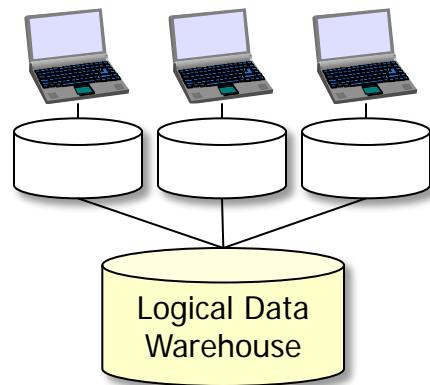


Architecture



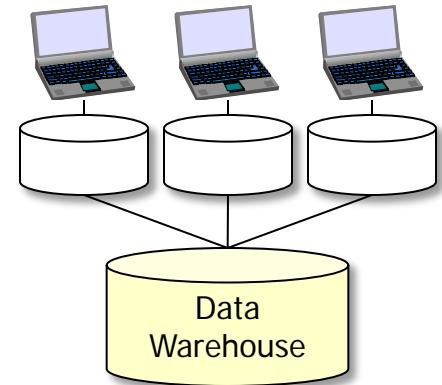
Central architecture

- only one data model
- performance bottleneck
- complex to build
- easy to maintain



Federal architecture

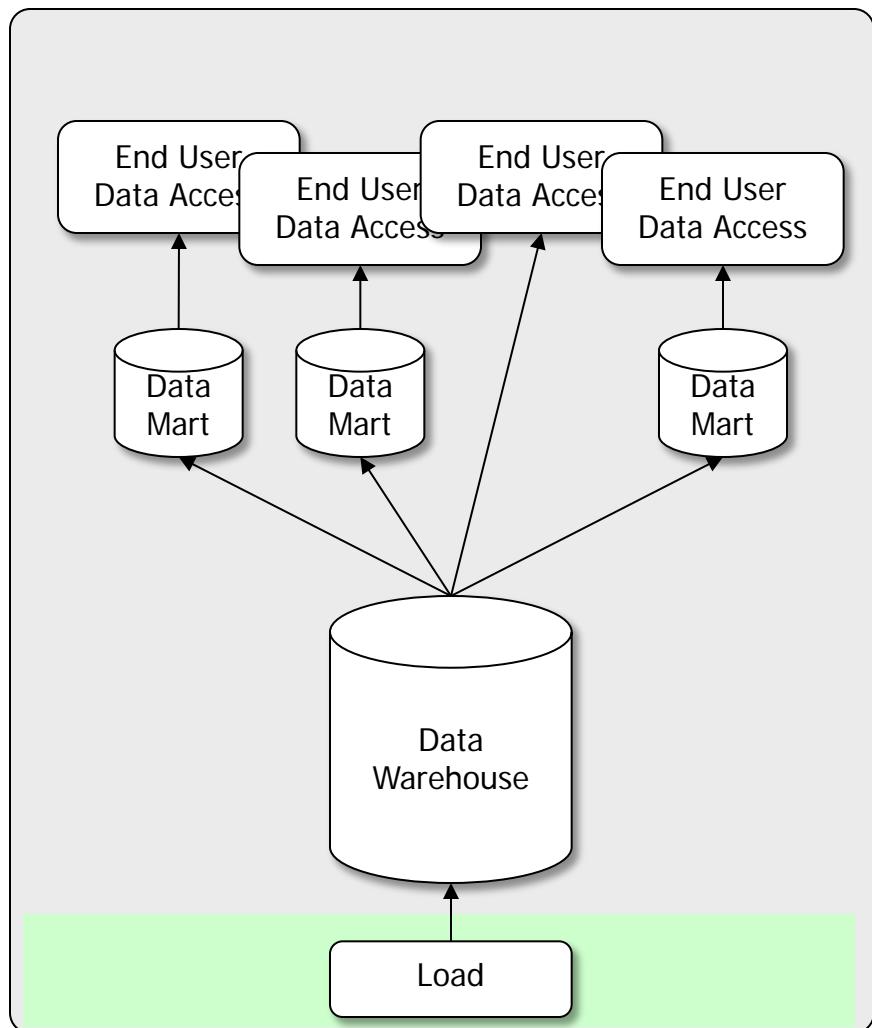
- logically consolidated
- separate physical databases that store detailed data
- faster response time



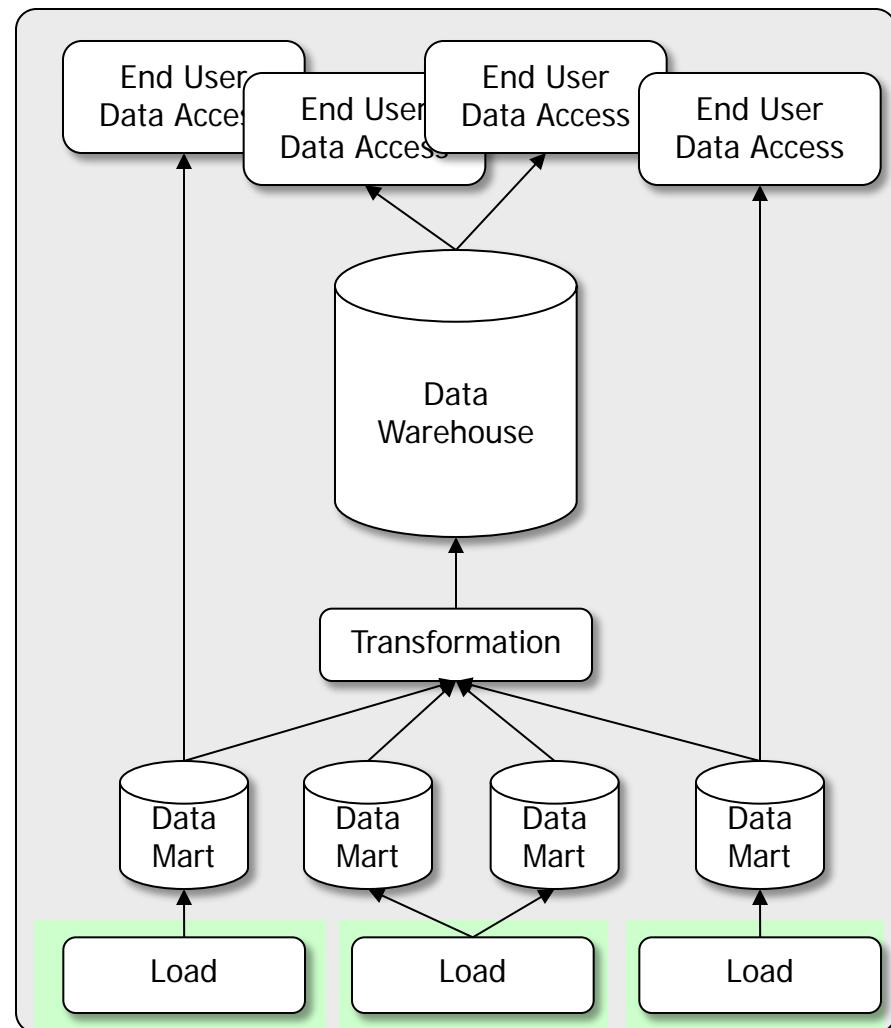
Tiered architecture

- physical central data warehouse
- separate physical databases that store summarized data
- faster response time

Data Marts



dependent data marts



independent data marts

Data Marts

dependent data marts
(tiered architecture)

- Central data warehouse (DW) is build first
- Extracts of the data warehouse are provided as data marts (materialized views)
- Establish ETL process for DW only
- Consistent analysis on DW and DM

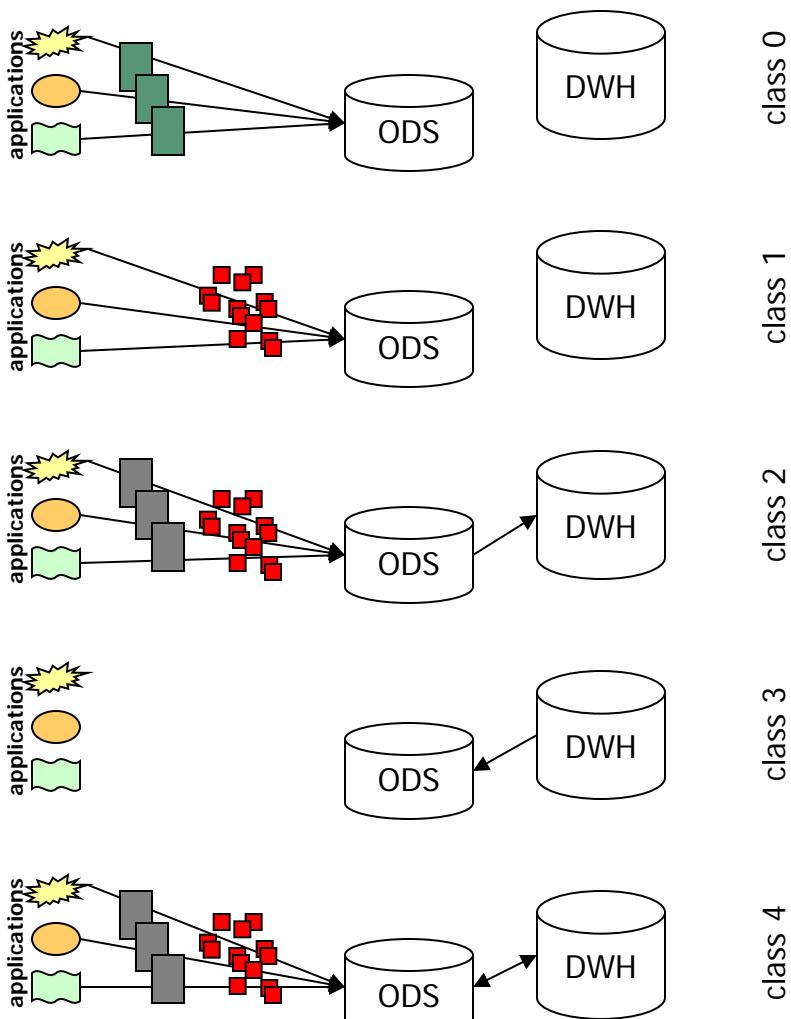
independent data marts
(federated architecture)

- Several data marts (DM) are build first
- Data marts are integrated by means of a second transformation step
- Establish ETL process for each DM and the central DW
- Inconsistent analysis is possible
- Virtual data warehouse possible (federated architecture)

Operational Data Store (ODS)

- Term has taken many definitions. For example:
 - **Point of integration for operational systems**
 - refreshed within a few seconds after the operational data sources are updated
 - very little transformations are performed
 - Example: Banking environment where data sources keep individual accounts of a large multinational customer, and the ODS stores the total balance for this customer.
 - ➡ true operational system separated from the data warehouse
 - **Decision support access to operational data**
 - integrated and transformed data are first accumulated and then periodically forwarded to the ODS
 - involves more integration and transformation processing
 - Example: Bank that stores in the ODS an integrated individual bank account on a weekly basis
 - ➡ part of the data warehouse or separate system?

Classes of Operational Data Stores

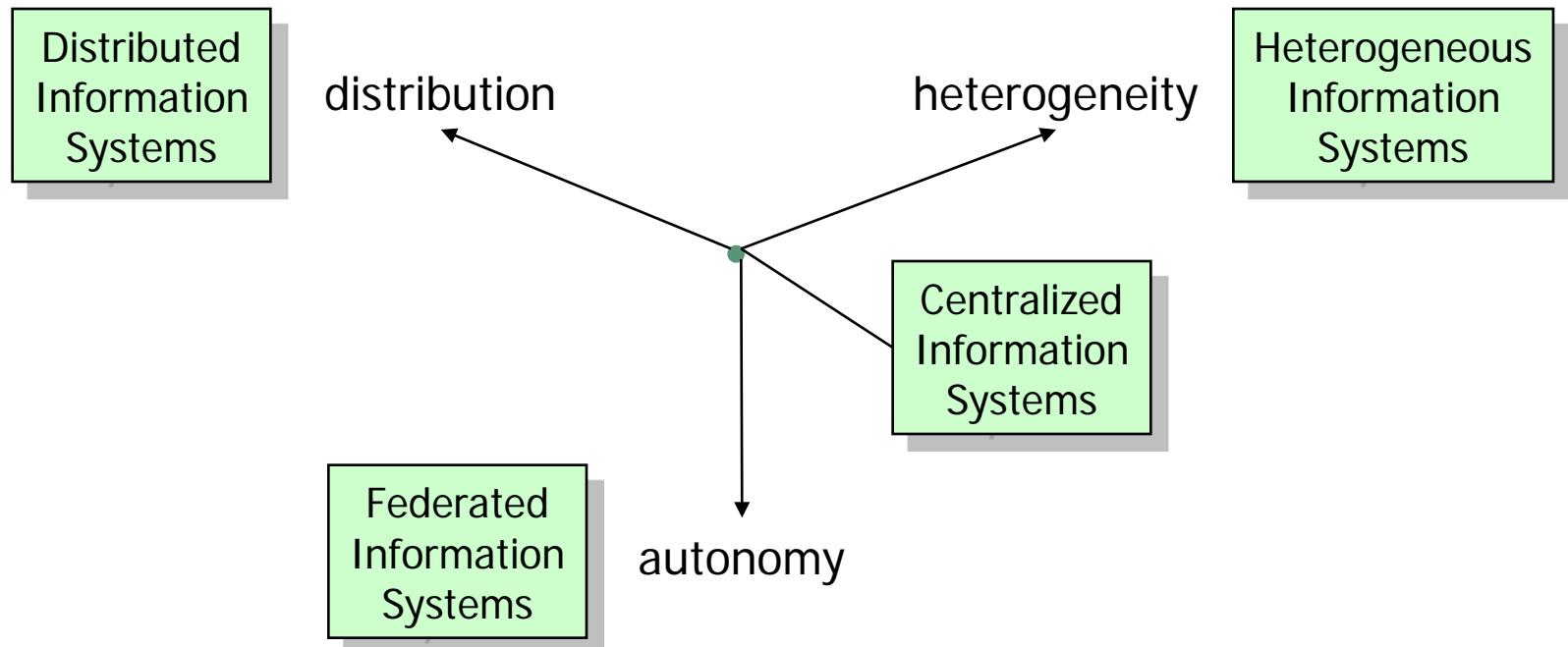


- Tables are copied from the operational environment
- Transactions are moved to the ODS in an immediate manner (range of one to two seconds)
- Activities in the operational environment are stored, integrated, and forwarded to the ODS
- ODS is fed aggregated analytical data from the data warehouse
- Combination of integrated data from the operational environment and aggregated data from the analytical environment

Overview

- Data Warehouse Architecture
 - Data Sources and Data Quality
 - Data Mart
 - Operational Data Store
- Information Integration
 - materialized vs. virtual
 - Federated Information Systems
- Metadata
 - Metadata Repository
 - Metadata in Data Warehousing
 - CWM Metamodel

Dimensions of Information Systems



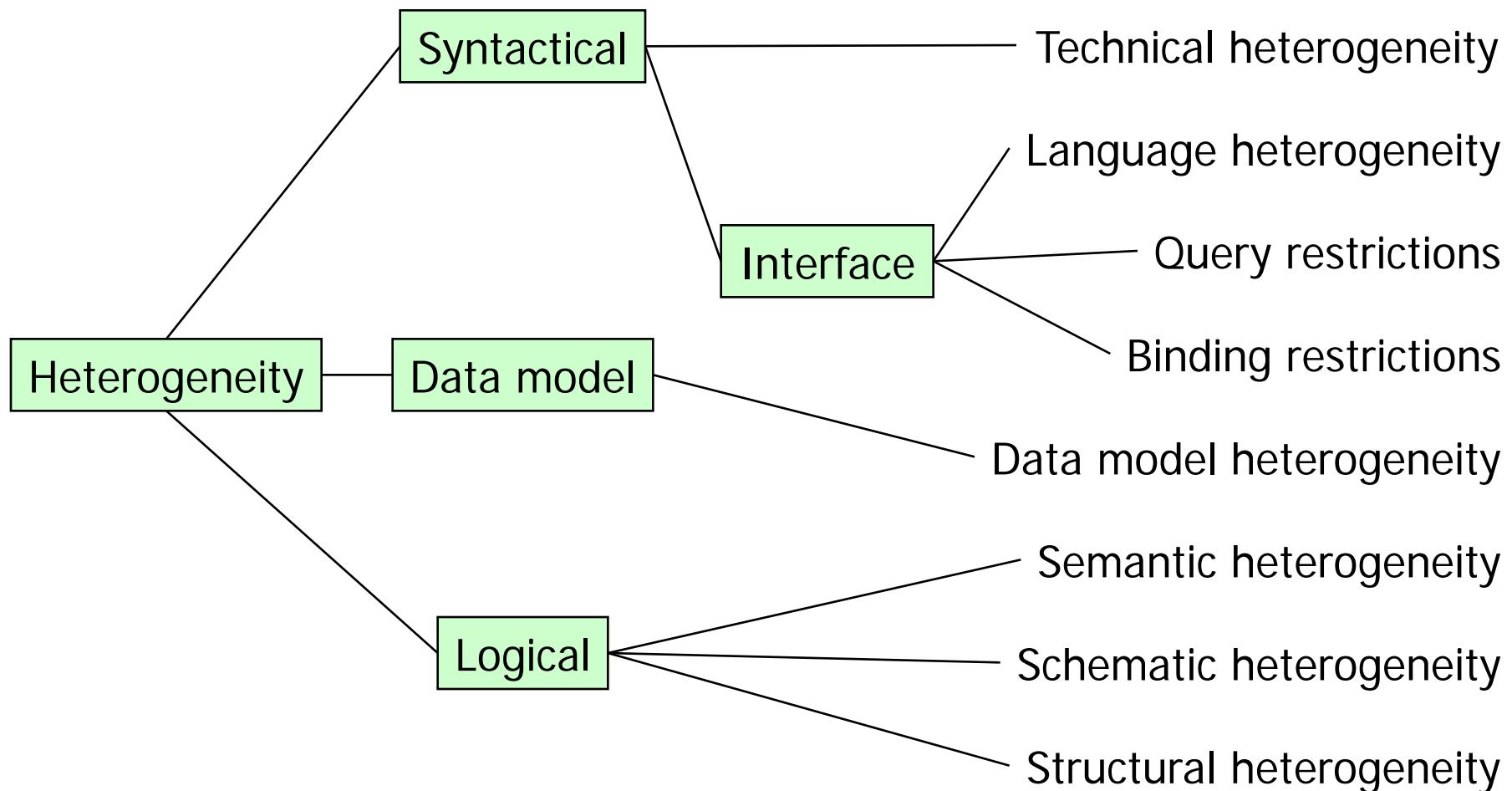
- **Definition:** Federated Information System

A federated information system consists of a set of distinct and autonomous system components, the participants of this federation. Participants in first place operate independently, but have given up some autonomy in order to participate in the federation.

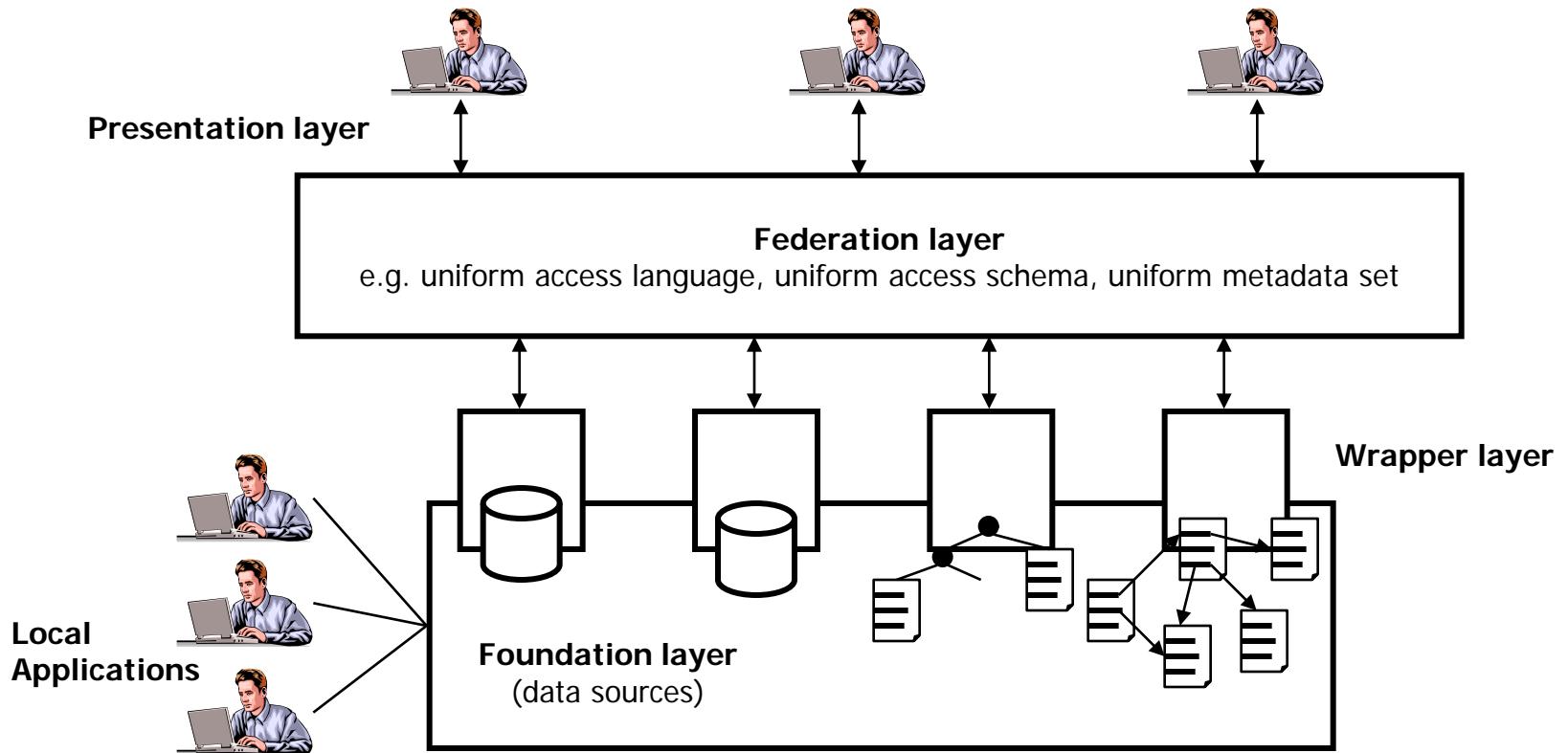
Autonomy of Information Systems

- Design autonomy
 - The design of one source system is independent from the design of other systems, e.g., in the universe of discourse, the data model, naming concepts, ...
- Communication autonomy
 - Source systems decide independently with which other systems they communicate.
 - Source systems can leave and enter a federation at any time.
- Execution autonomy
 - Source systems independently decide on execution and scheduling of incoming requests.

Heterogeneity in Information Systems



Federated Information Systems



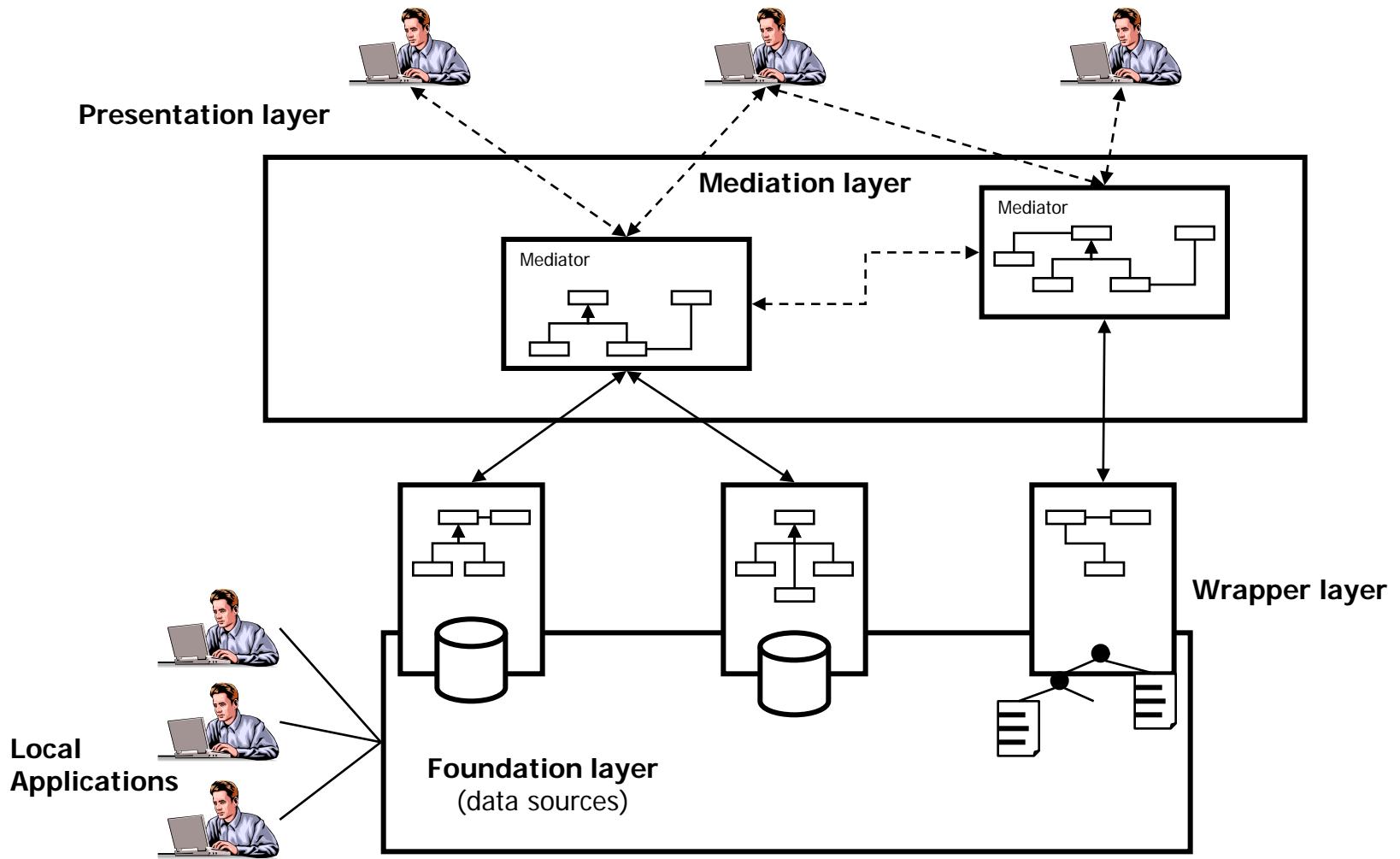
Types of Federated Information Systems

	Loosely coupled Information Systems	Federated Databases	Mediator-based Information Systems
Types of heterogeneity addressed	Only technical and language heterogeneity	All, except query restriction heterogeneity; schema integration difficult for schematic heterogeneity	All
Loss of autonomy?	Execution autonomy	Execution autonomy; notification of schema changes	Execution autonomy
Transparency	Language	Location, schema and partly language	Location, schema and language
Kind of components	Structured	Structured	Any
Access methods	Query language	Query language	Any
Access restrictions	No	No	Yes
Write access?	Yes	Yes	No

Types of Federated Information Systems

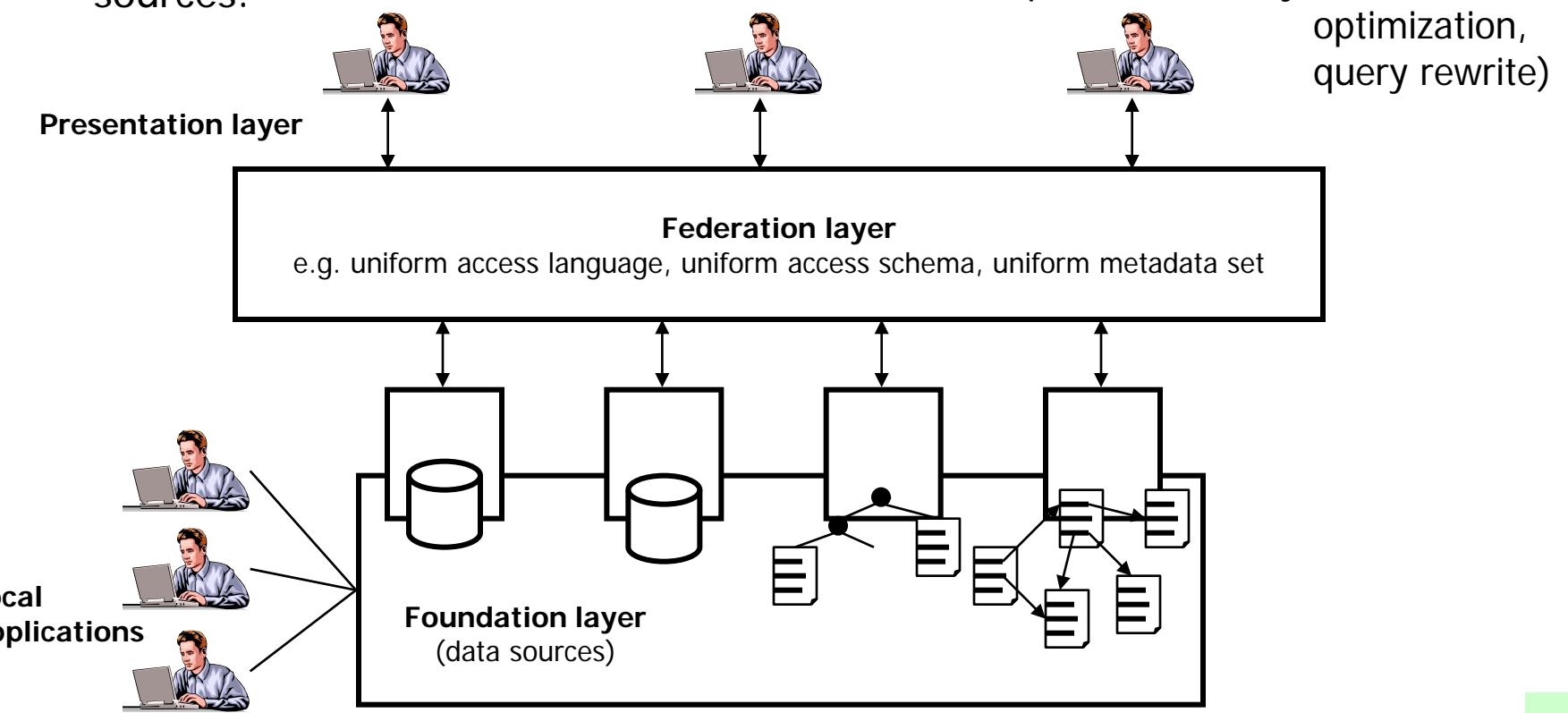
	Loosely coupled Information Systems	Federated Databases	Mediator-based Information Systems
Tight vs. loose integration	Loose	Tight	Tight
Kinds of semantic integration	Collection	Collection and fusions	Collection, fusion, sometimes abstraction
Necessary metadata	Technical, infrastructure	Logic, technical, semantic	Logic, technical, semantic
Bottom-up vs. top- down	n.a.	Bottom-up	Top-down
Virtual vs. materialized	Virtual	Virtual	Virtual
Evolvability	High	Low	High

Mediator-based Information Systems

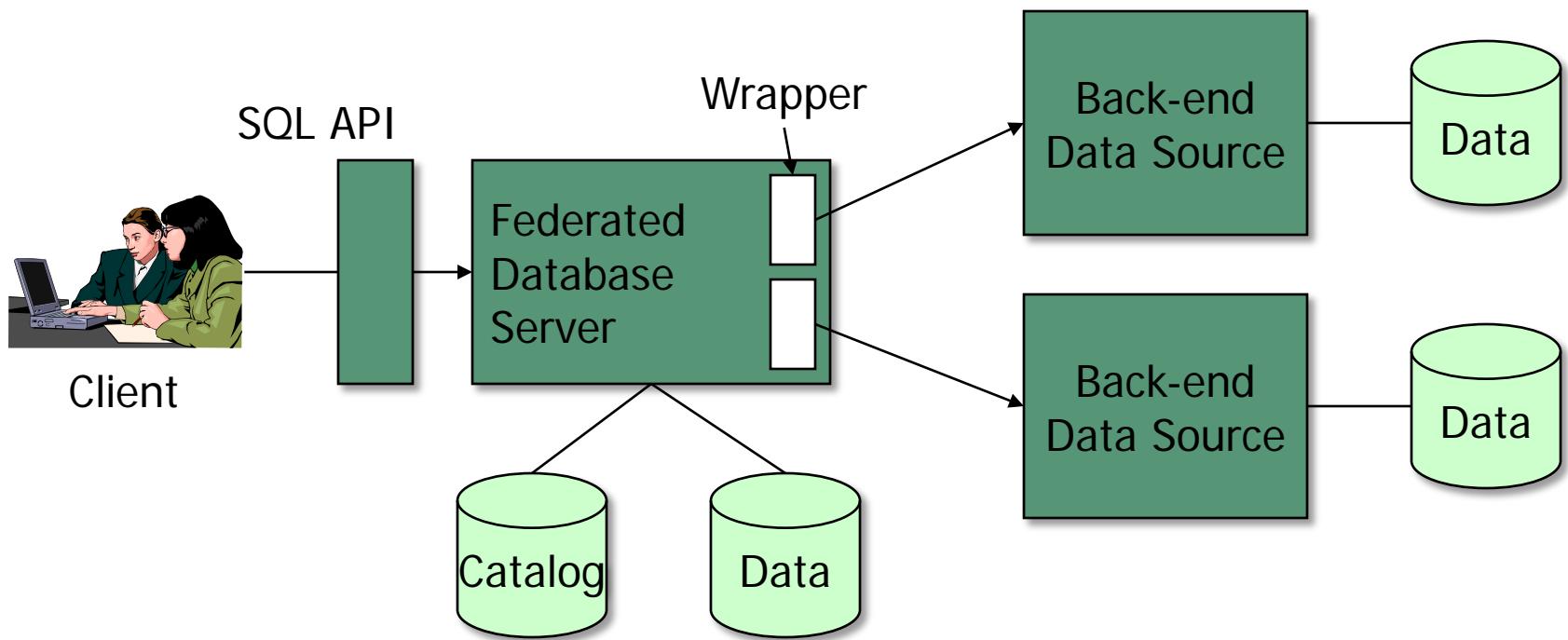


Federated Database Systems

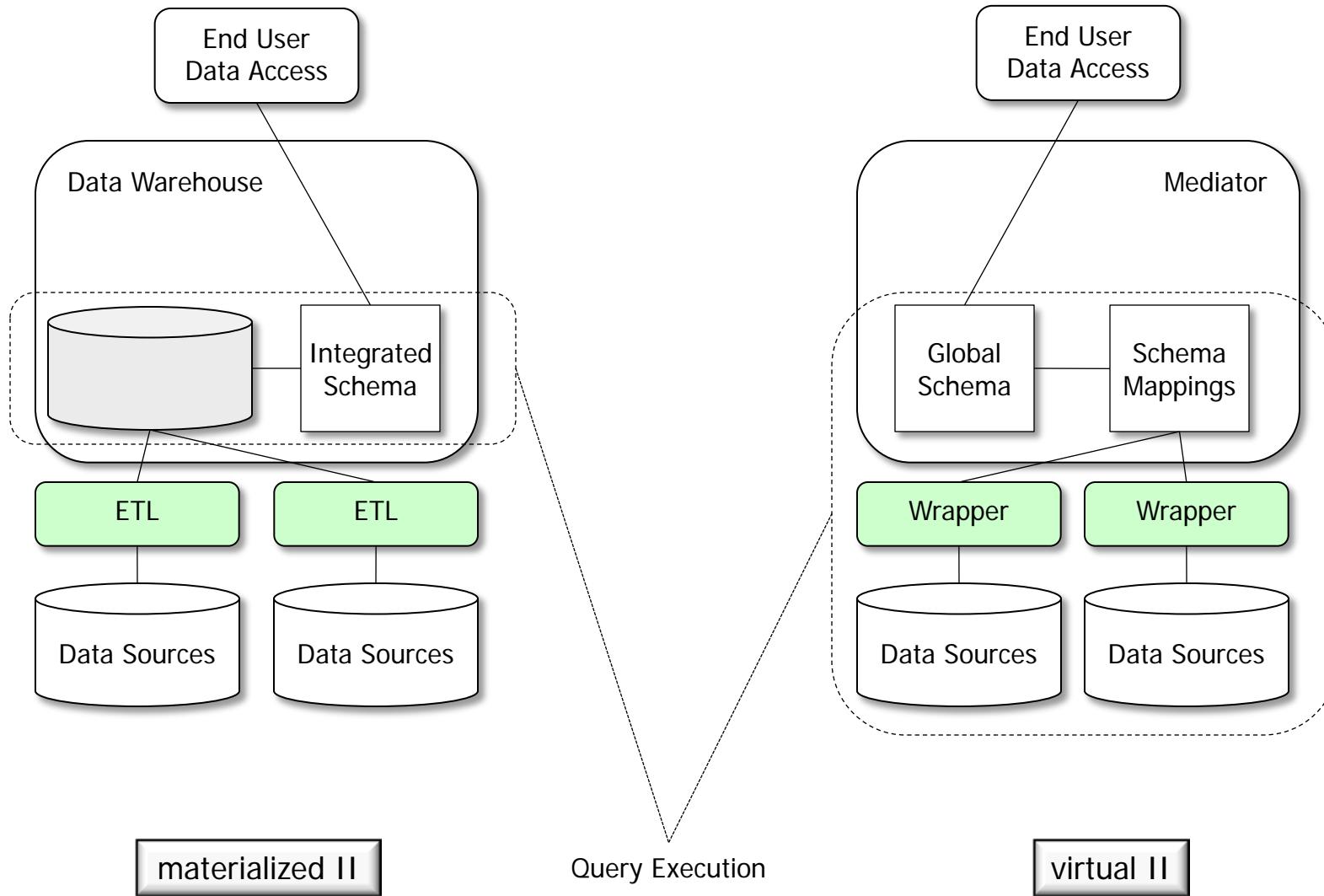
- Federated DBMS (FDBS)
 - Transparent access to a collection of heterogeneous and semi-autonomous data sources.
 - Complete, extensible database engine
 - Function compensation
 - Powerful (global) query optimizer (pushdown analysis, cost-based optimization, query rewrite)



Architecture for a FDBS



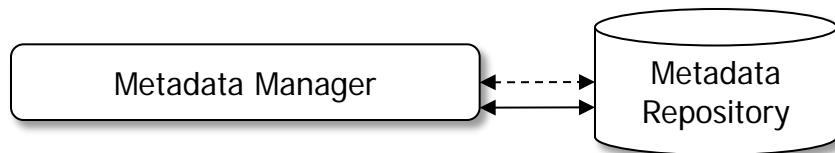
Information Integration



Overview

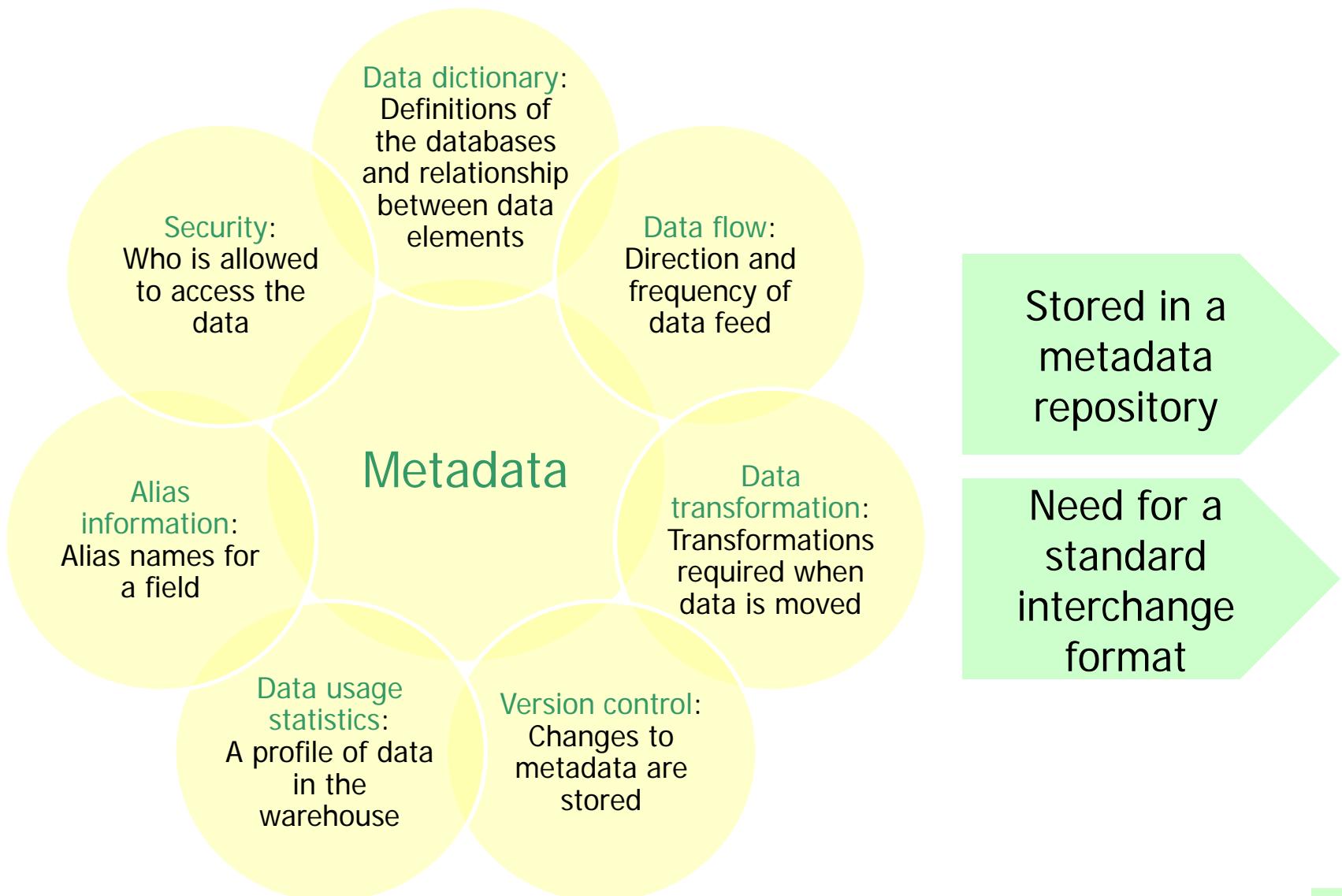
- Data Warehouse Architecture
 - Data Sources and Data Quality
 - Data Mart
 - Operational Data Store
- Information Integration
 - materialized vs. virtual
 - Federated Information Systems
- Metadata
 - Metadata Repository
 - Metadata in Data Warehousing
 - CWM Metamodel

Metadata Repository



- A repository is a shared database of information about engineering artifacts, such as software, documents, maps, information systems, and manufactured components and systems.
- Functions of a repository:
 - Object management
 - Dynamic extensibility
 - Relationship management
 - Notification
 - Version management
 - Configuration management

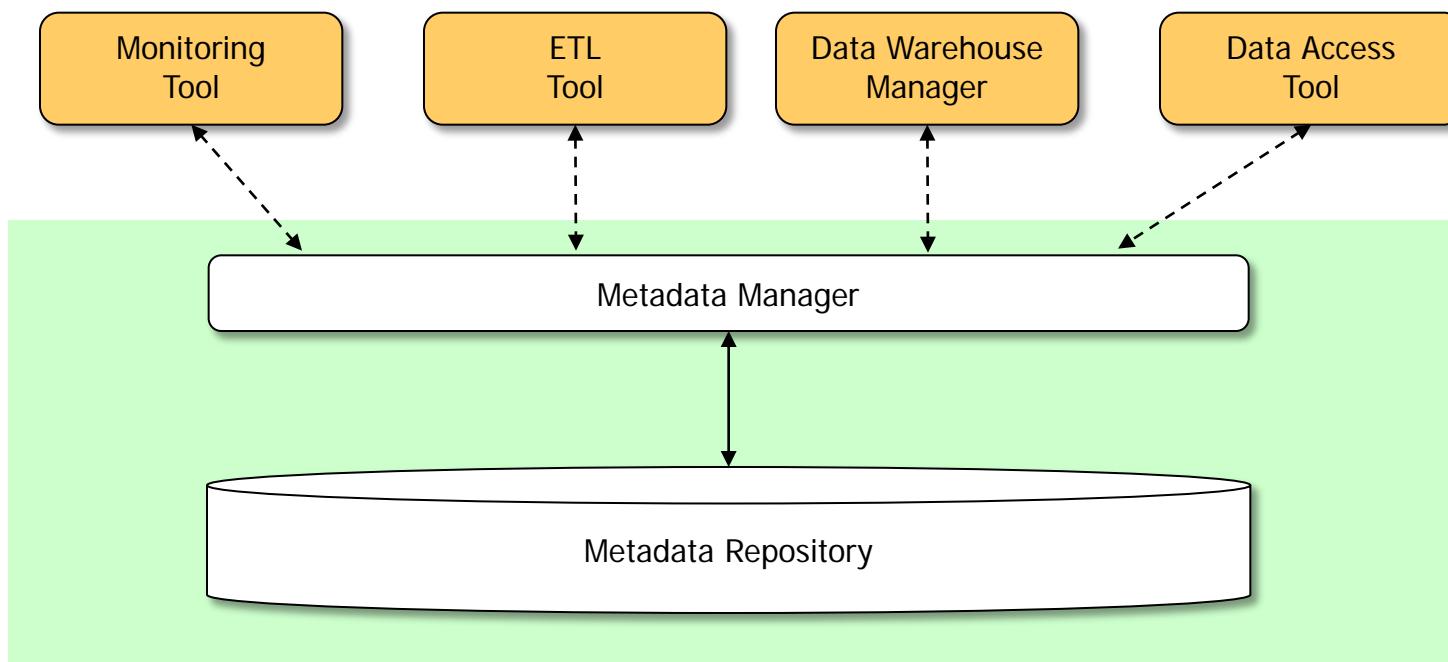
Metadata in Data Warehousing



Metadata in Data Warehousing

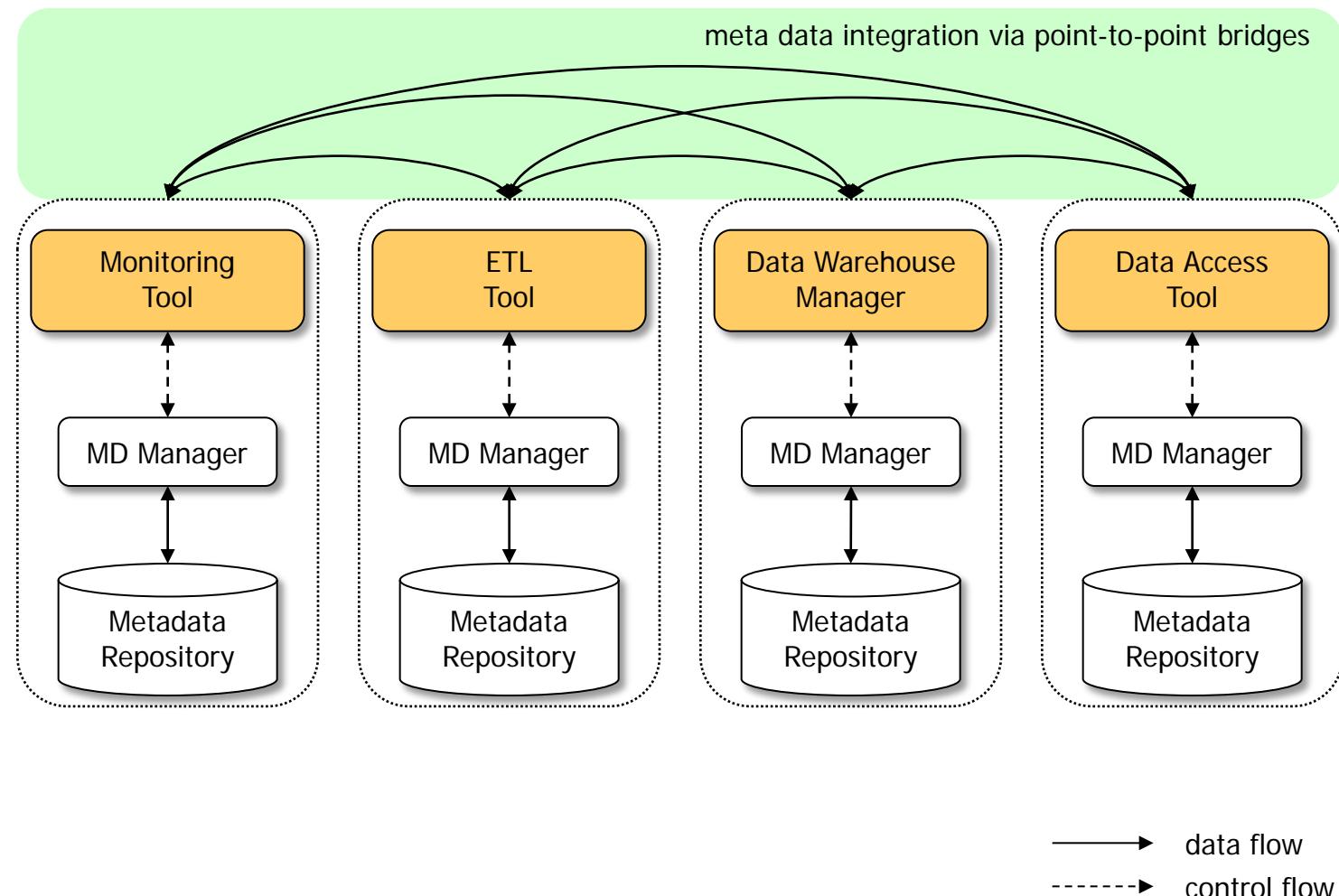
- Criteria to identify important classes of metadata in data warehousing:
 - Type of data
 - Abstraction
 - User
 - Origins
 - Time
- Usage of metadata in data warehousing:
 - passive
 - active
 - semi-active
- Main goals:
 - Support development and operation of a data warehouse
 - system integration
 - processes for DW administration
 - flexible application development
 - access rights
 - Provide information for data warehouse users
 - quality of data
 - consistent terminology
 - support for data analysis

Centralized Metadata Management

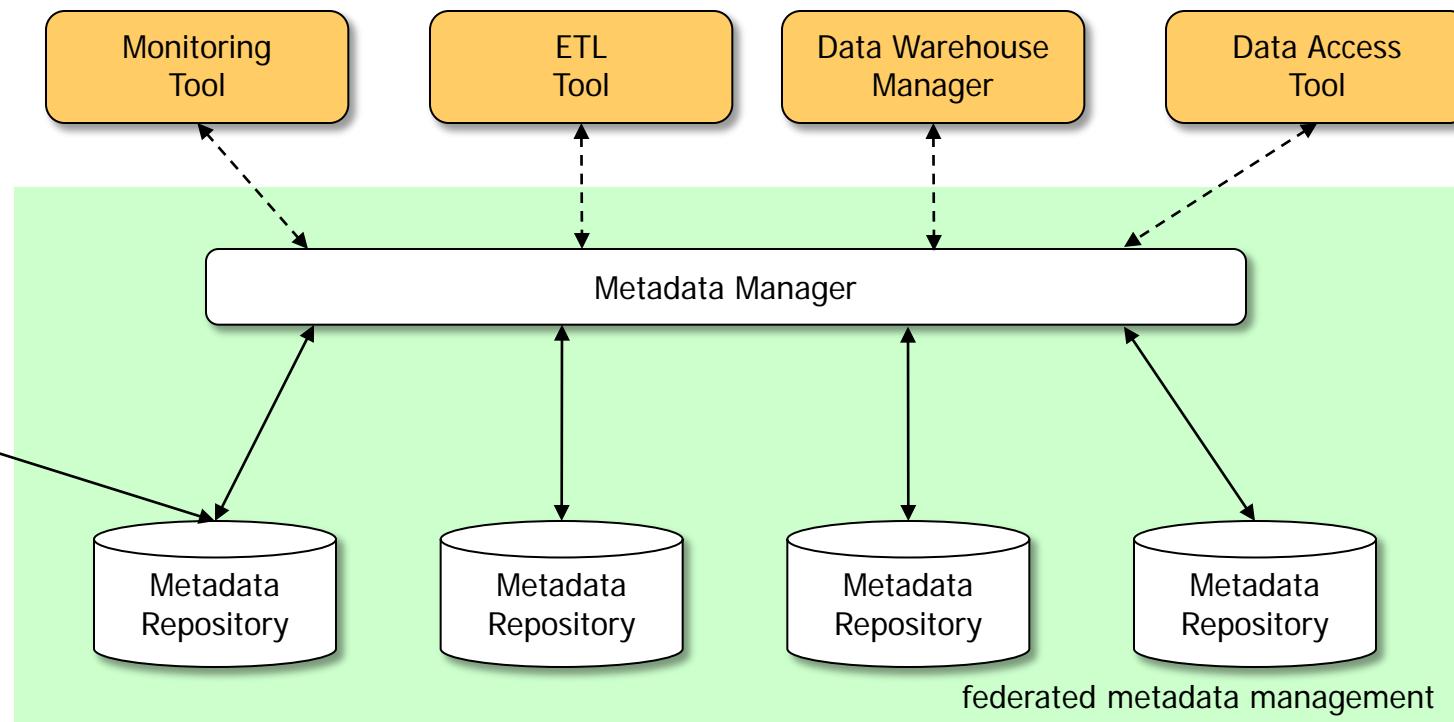


→ data flow
→ control flow

Decentralized Metadata Management



Federated Metadata Management



Metadata Interchange

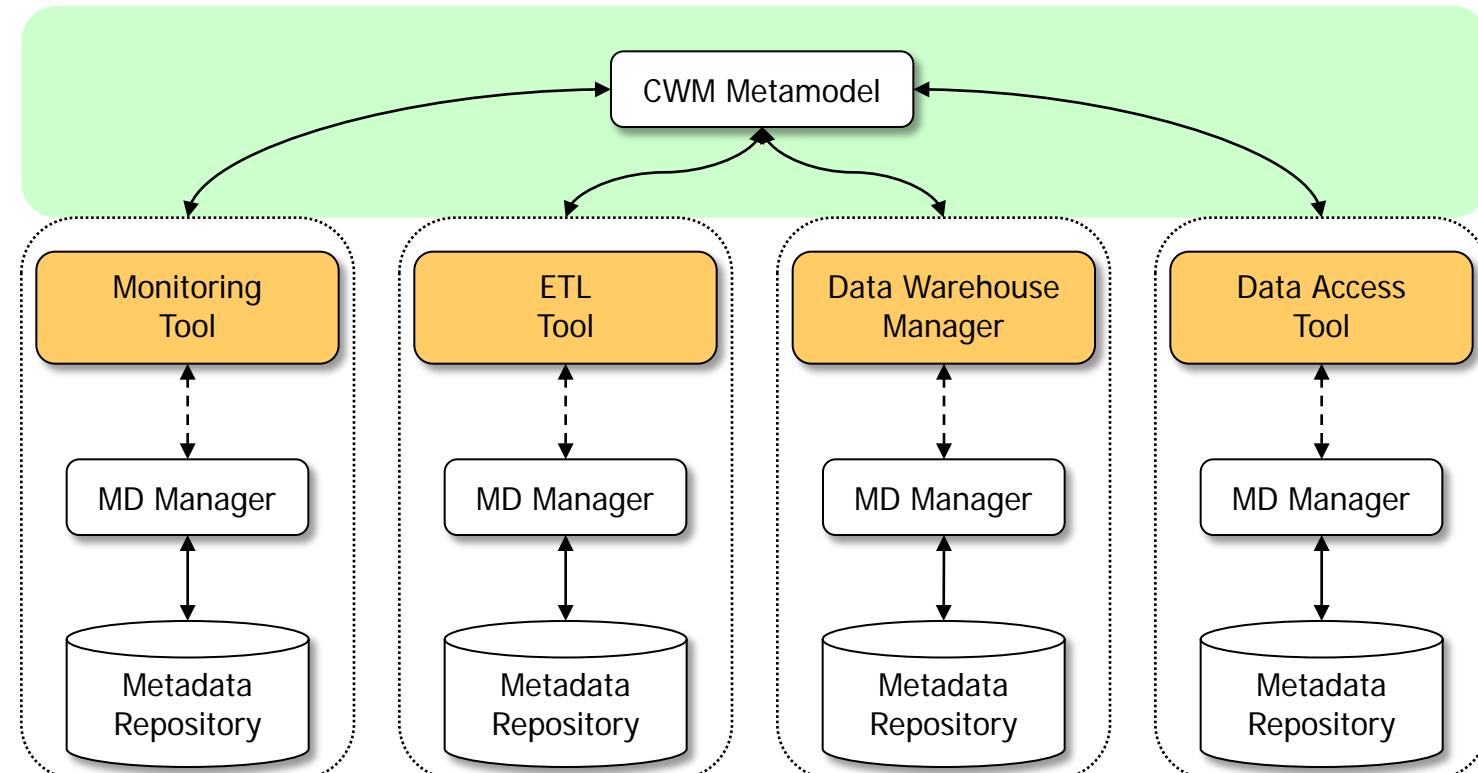
- Standardization Approaches:
 - Case Data Interchange Format (CDIF)
 - Electronic Industries Association (EIA)
 - Focus on CASE Tools
 - Interchange based on files
 - Open Information Model (OIM)
 - Meta Data Coalition's Metadata Model
 - Interchange based on XML
 - Merged with CWM since 1999
 - Common Warehouse Model (CWM)
 - Metadata Model published by the Object Management Group (OMG)
 - Interchange based on XML

CWM Metamodel

The main purpose of CWM is to enable easy interchange of warehouse and business intelligence metadata between warehouse tools, warehouse platforms and warehouse metadata repositories in distributed heterogeneous environments.

- Design goals: The CWM model should ...
 - reuse concepts in the UML metamodel where applicable.
 - be subdivided into packages allowing implementation of relevant subsets of the model.
 - be independent of any specific data warehouse implementation. Yet it should contain features that are effective in, and mappable to, a broad range of representative warehouse configurations based on specific tools.

Model-based Metadata Management



→ data flow
↔ control flow

CWM Metamodel

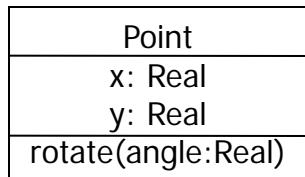
- CWM is based on three key industry standards:
 - UML - MOF - XMI
- CWM needs a formal language capable of representing meta data in terms of shared platform-independent models.
- CWM is based on a framework that supports any kind of meta data and allows new kinds of meta data to be added as required.
- Meta data integration based on CWM requires a common interchange format for exchanging instances of shared meta data.



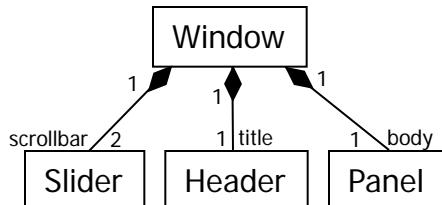
UML

"The Unified Modeling Language (UML) is a graphical language for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system. The UML offers a standard way to write a system's blueprints, including conceptual things such as business processes and system functions as well as concrete things such as programming language statements, database schemas, and reusable software components." (OMG)

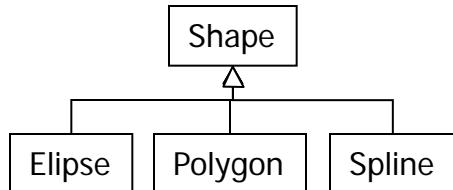
- Class Model



- Composition



- Inheritance



- The UML notation is used in the diagrammatic representations of the CWM metamodel.

Meta Object Facility (MOF)

- The MOF model can be used as a model for defining information models.
- In this context, the MOF Model is referred to as a meta-metamodel because it is being used to define metamodels (UML, CWM, ...).
- The main modeling concepts are:
 - **Classes**, which model MOF meta-objects.
 - **Associations**, which model binary relationships between meta-objects.
 - **DataTypes**, which model other data (e.g. primitive types, external types, etc.).
 - **Packages**, which modularize the models.

Metadata Architecture

3 The "MOF Model"

2 UML Metamodel

1 UML Model

0 modelled
system

meta-
metamodel

metamodel,
meta-metadata

model,
metadata

object,
data

The "MOF Model"

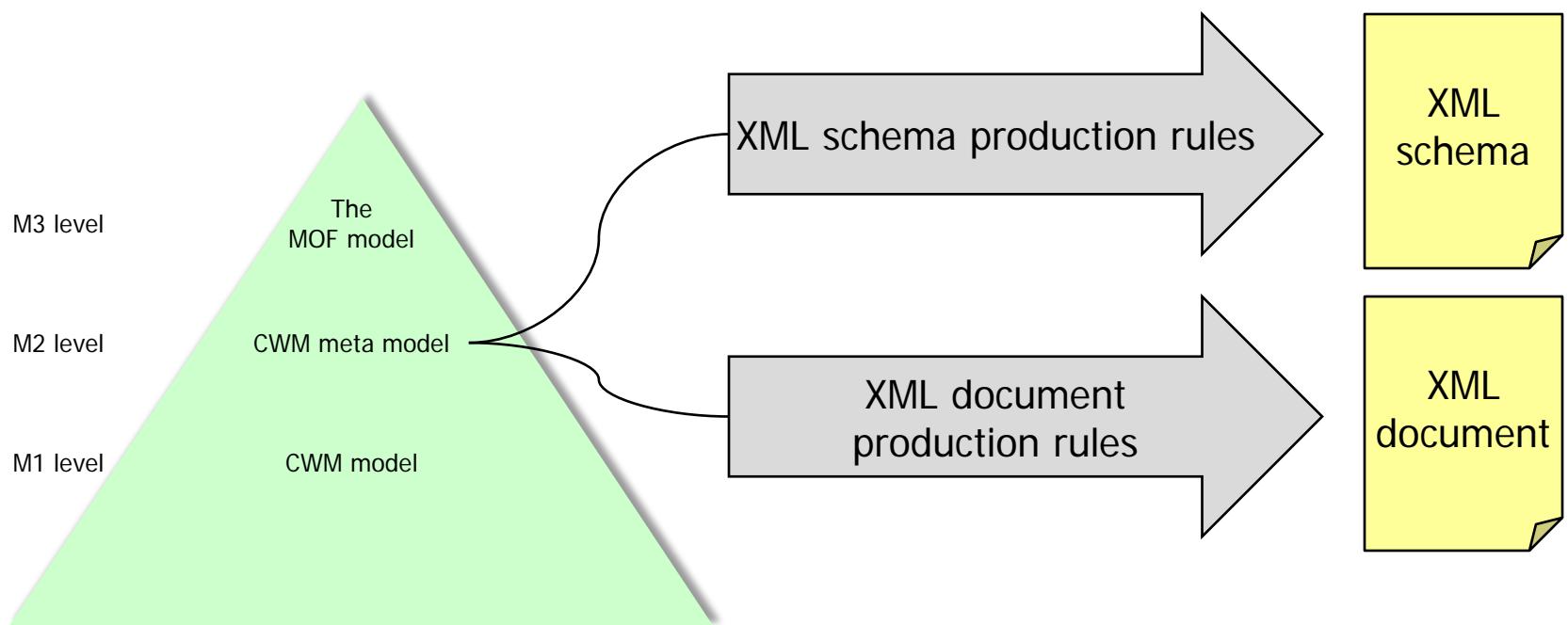
CWM Metamodel

CWM Model

warehouse
data

XML Metadata Interchange (XMI)

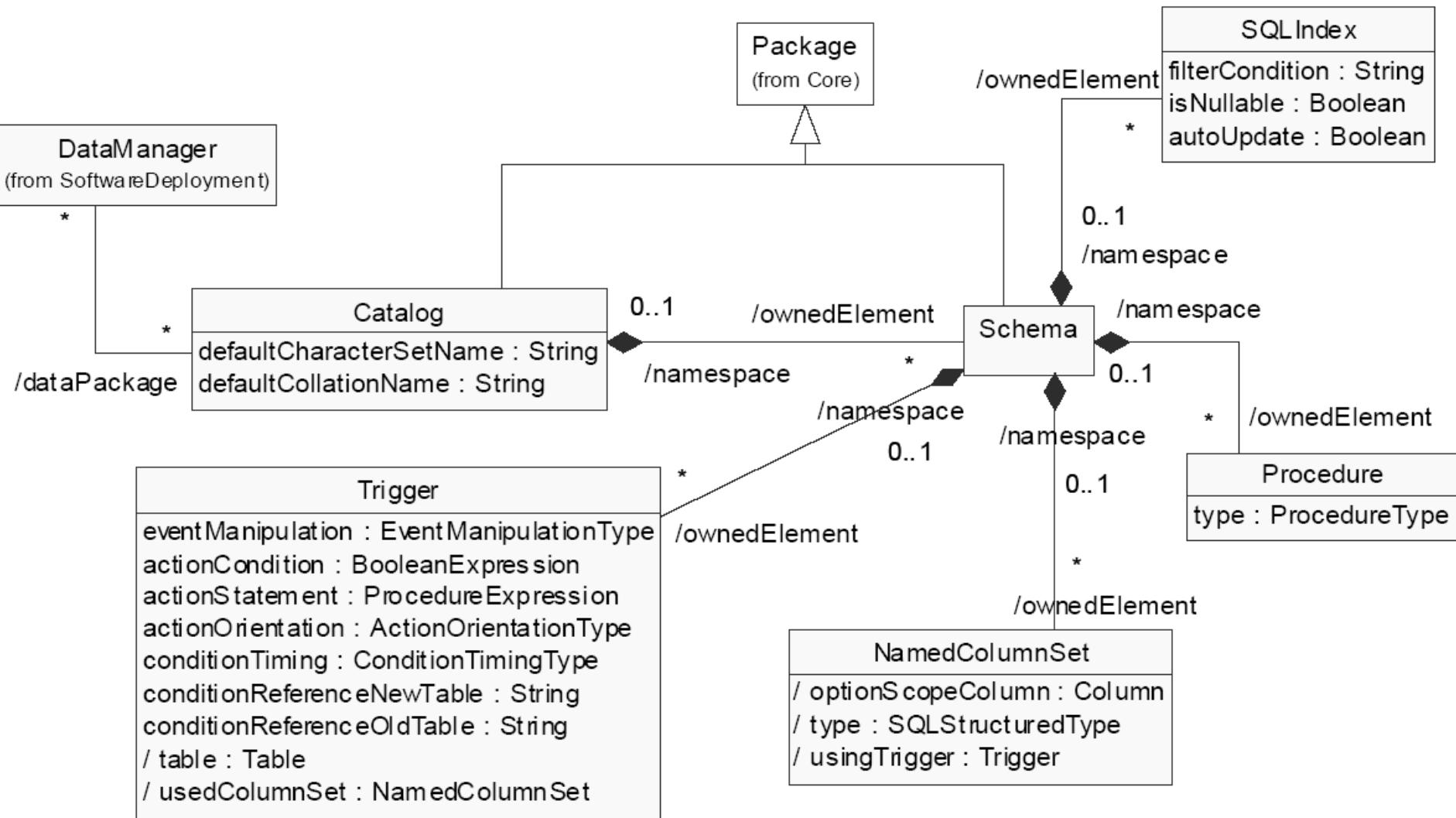
- Supports interchange of MOF-based meta-data.
- Applicable to a wide variety of objects: UML, Java, C++, EJB, IDL, CWM.
- Production of XML Schemas starting from an object model.
- Production of XML documents starting from objects.



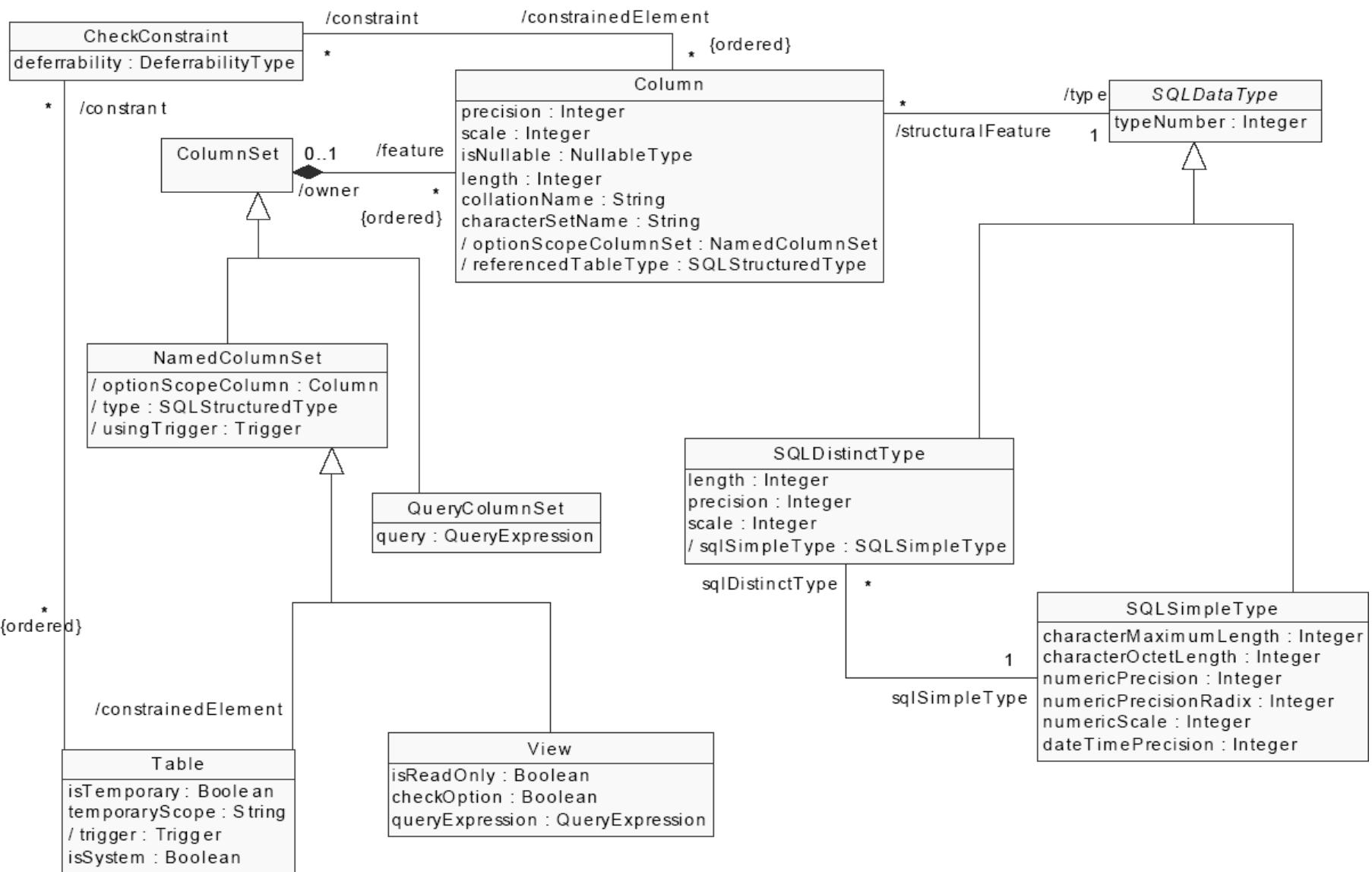
CWM Metamodel

Management Analysis Resource Foundation	Warehouse Process			Warehouse Operation		
	Transformation		OLAP	Data Mining	Information Visualization	Business Nomenclature
	Object Model	Relational	Record	Multidimensional		XML
	Business Information	Data Types	Expression	Keys and Indexes	Type Mapping	Software Deployment
	Object Model					

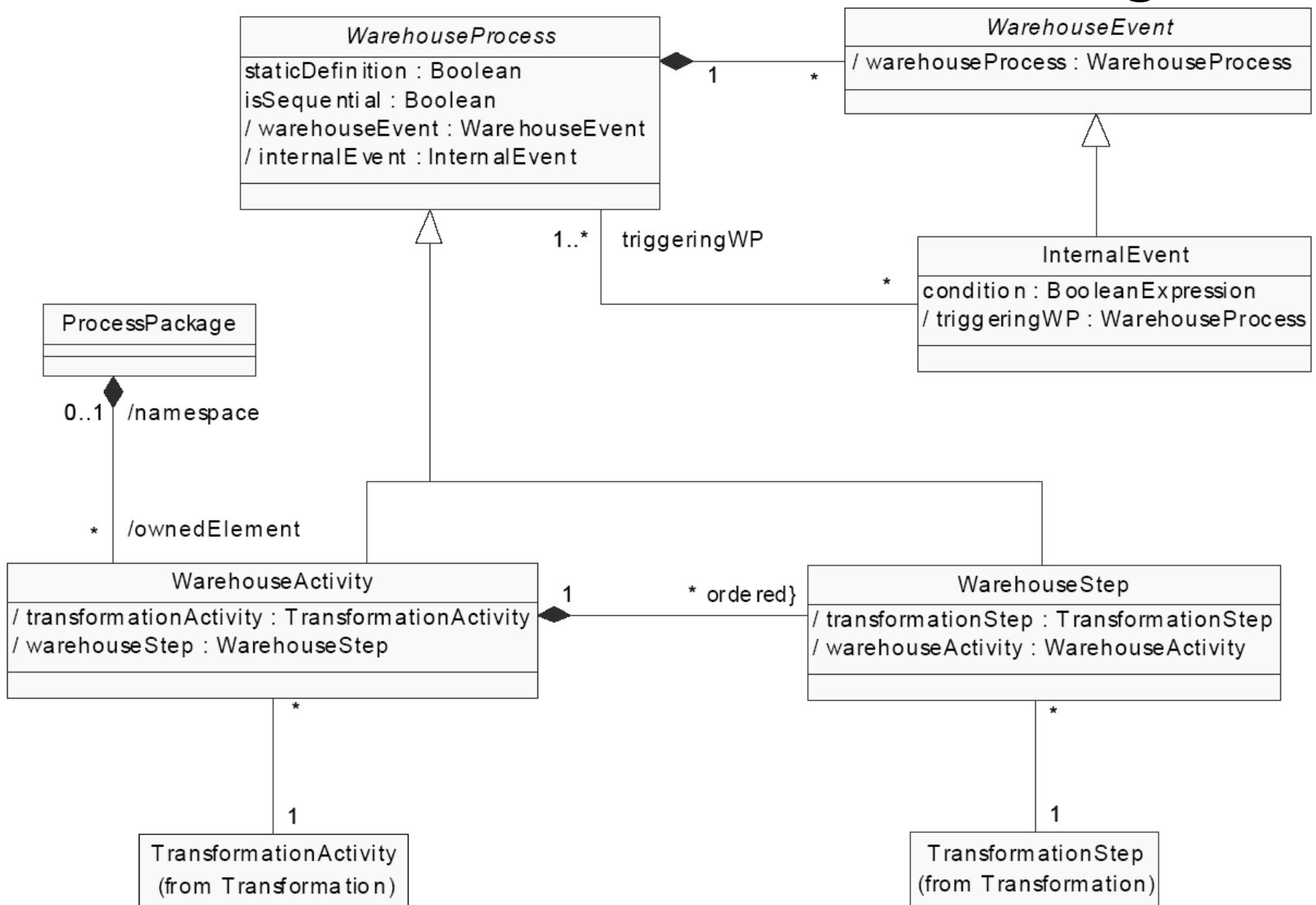
CWM Relational Package



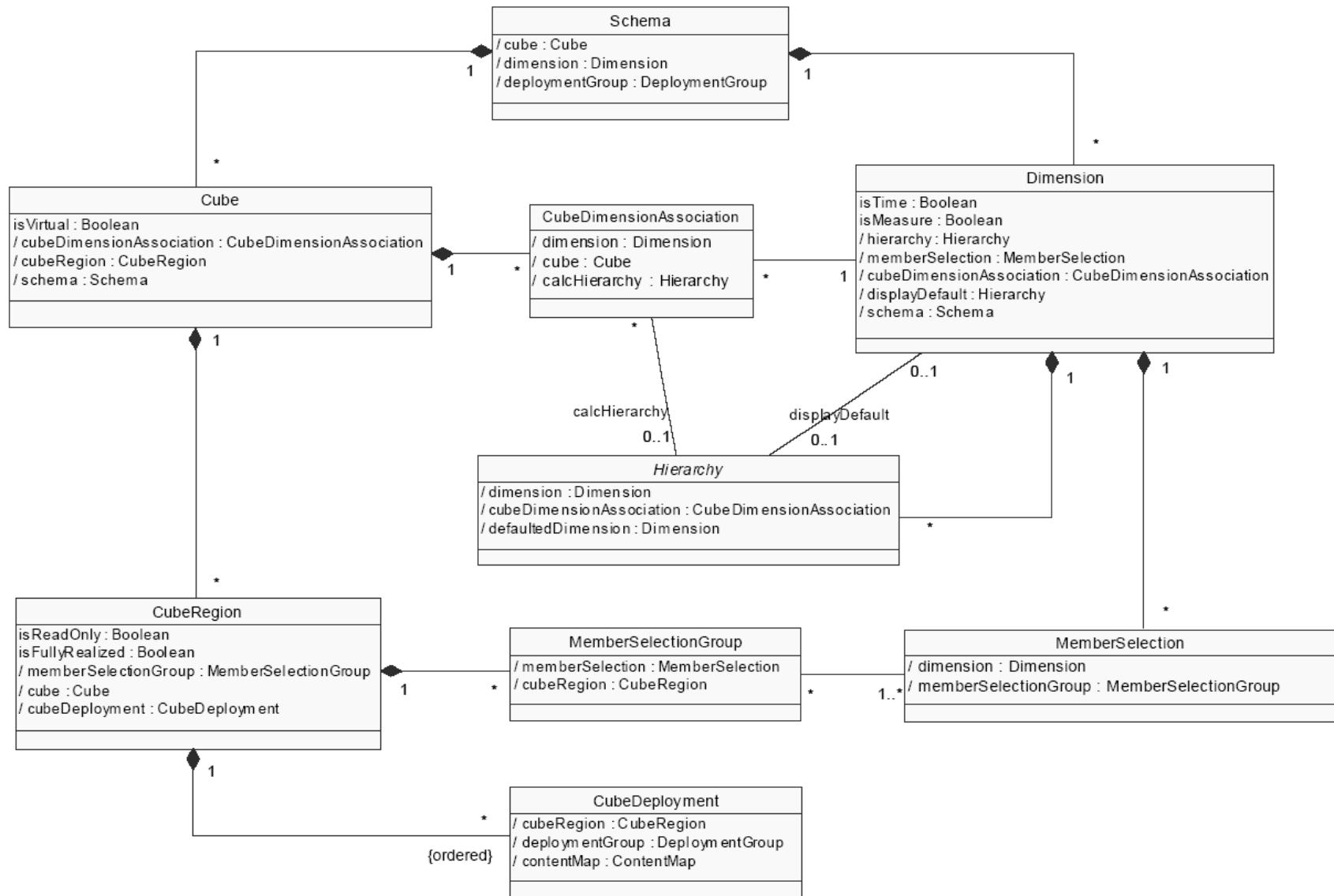
CWM Relational Package



CWM Warehouse Process Package



CWM OLAP Package



Summary

- Basic Components:
 - Data Staging Area: Extraction, Transformation, Load
 - Data Warehouse Database
 - Data Warehouse Manager
 - Metadata Repositories and Metadata Manager
- Data Marts: Distributed Data Warehouse
- Materialized vs. virtual information integration
- Metadata is important to:
 - Support development and operation of a data warehouse
 - Provide information for data warehouse users
- Metadata standards are important to interchange metadata between warehouse tools, warehouse platforms and warehouse metadata repositories.

Papers & Books

- [Ber98] P. A. Bernstein: Repositories and Object Oriented Databases. SIGMOD Record 27(1): 88-96 (1998).
- [BK+99] S. Busse, R.-D. Kutsche, U. Leser, H. Weber: Federated Information Systems: Concepts, Terminology and Architectures. Forschungsberichte des Fachbereichs Informatik Nr. 99-9, TU Berlin April 1999.
- [PC+03] J. Poole, D. Chang, D. Tolbert, D. Mellor: Common Warehouse Metamodel: Developer's Guide. OMG Press, 2003.
- [SL90] A. P. Sheth, J. A. Larson: Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases. ACM Comput. Surv. 22(3): 183-236(1990).

Data-Warehouse-, Data-Mining- und OLAP-Technologien

Chapter 3: Data Warehouse Design

Bernhard Mitschang
Universität Stuttgart

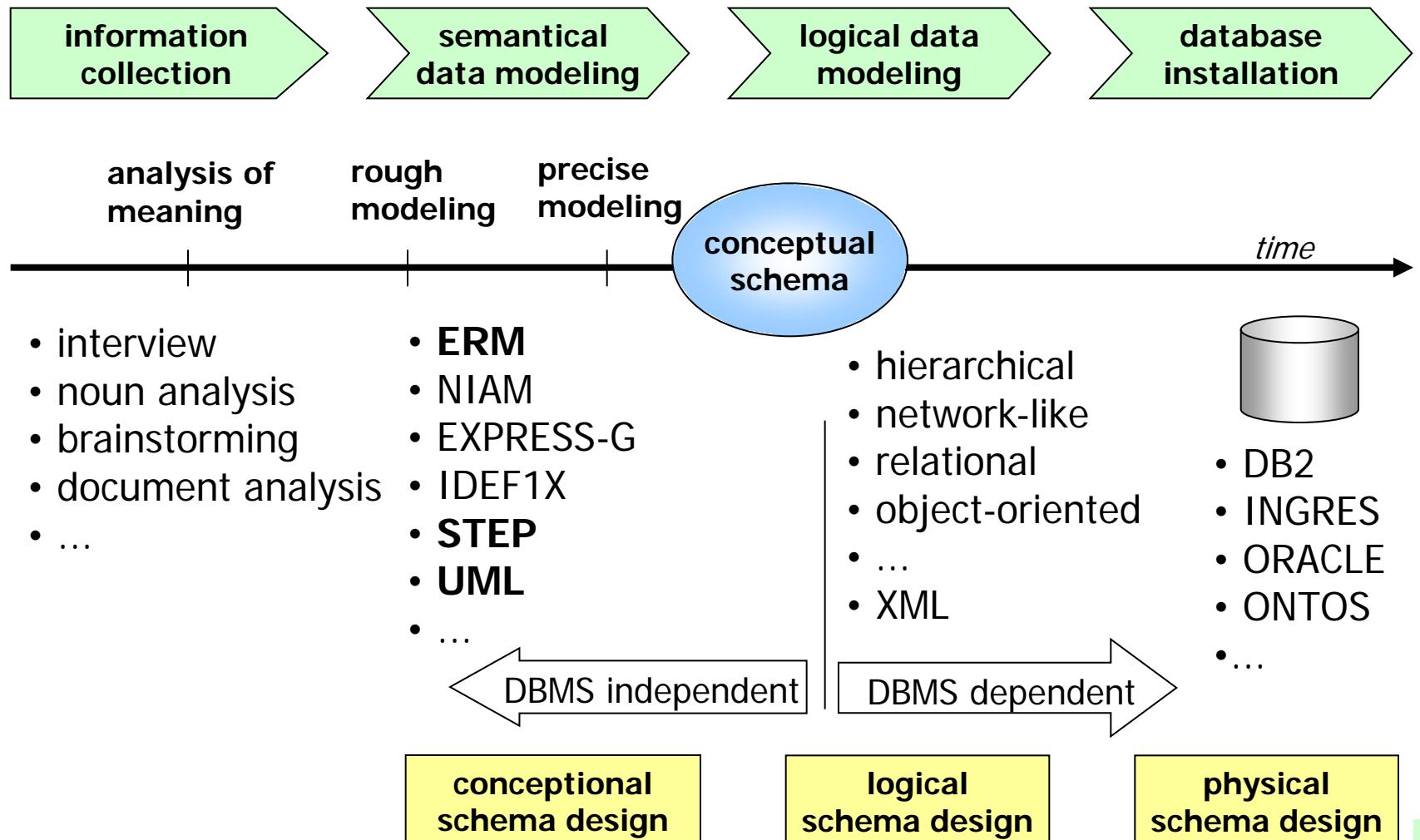
Winter Term 2015/2016

Overview

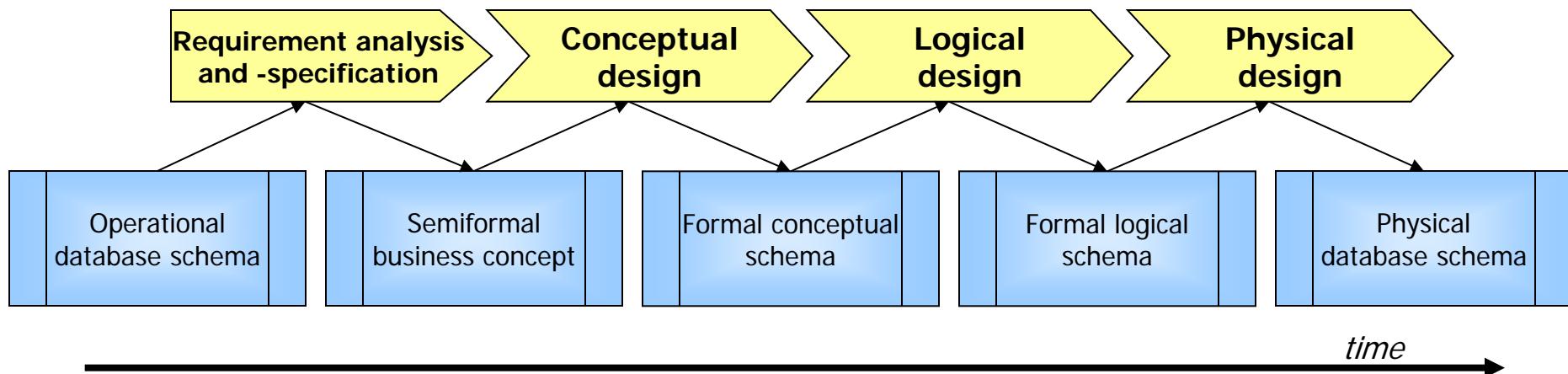
- Data Warehouse Design Process
- Conceptual Design
- Logical Design
- Details of Logical Design
- Physical Design

Database Design

- Process Model:



Data Warehouse Design Process



- interview
- noun analysis
- brainstorming
- document analysis
- ...
- starER
- dimensional fact model
- ME/R
- mUML
- ...
- multidimensional
- relational
- object-relational
- ...
- DB2
- ORACLE
- MS Server
- Essbase
- MS Analysis Services
- ...

Overview

- Data Warehouse Design Process
- Conceptual Design
 - Multidimensional Model
 - Dimensional Fact Model
 - starER
 - UML Profile for Multidimensional Modeling
- Logical Design
- Details of Logical Design
- Physical Design

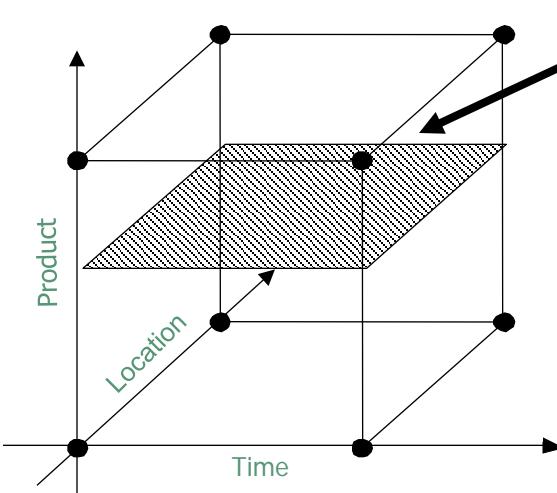
Conceptual Design

Transformation of the semi-formal business requirements specification into a formalized conceptual multidimensional schema.

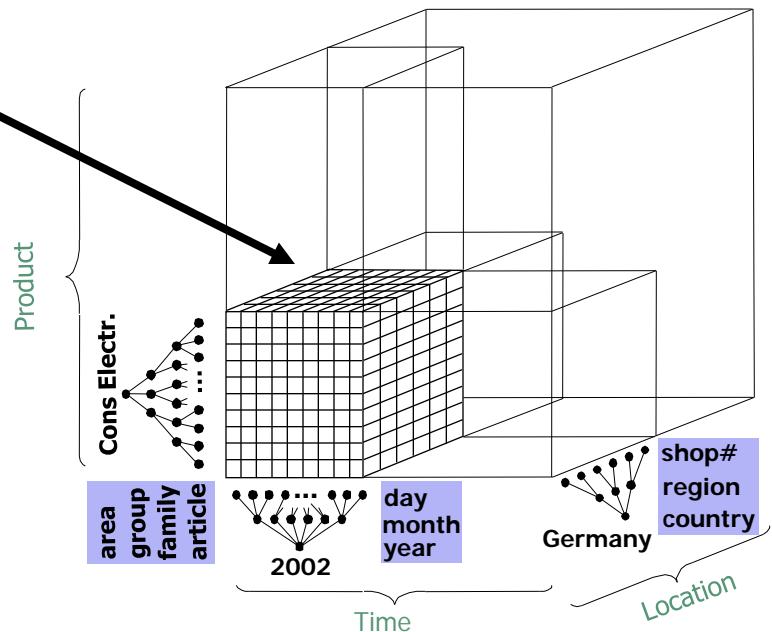
- Most approaches produce a graphical multidimensional schema: Dimensional Fact Model, starER, UML-based, ...
- Conceptual Design is based on:
 - business requirement specification
 - ER schema of the operational systems
- Process model to conceptual data warehouse design:
 - context definition of measures
 - dimensional hierarchy design
 - definition of summarizability constraints

Multidimensional Model

Multidimensional Model



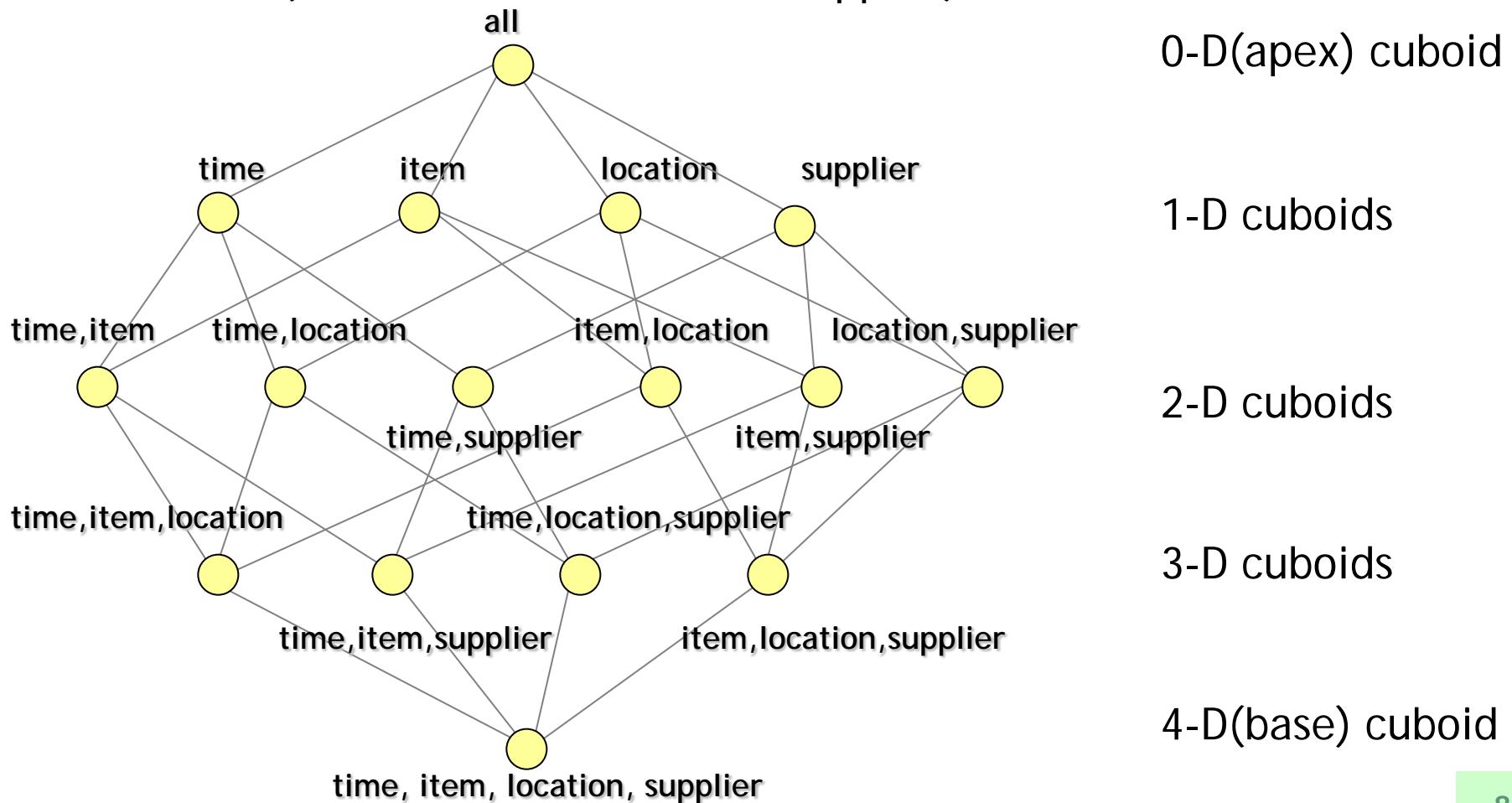
„Cube“ Metaphor



- A data warehouse is based on a multidimensional data model which views data in the form of a **data cube**. A data cube, such as sales, allows data to be modeled and viewed in multiple dimensions.
- In data warehousing literature, an n-D base cube is called a base cuboid. The top most 0-D cuboid, which holds the highest-level of summarization, is called the apex cuboid. The lattice of cuboids forms the data cube.

Data Cube

- Example Data Cube for SALES modeled and viewed in multiple dimensions (item, time, location, and supplier)



Basic Elements of a Conceptual Model

Fact data

- Mostly numeric data that is observed or measured
- Example: turnover/sales, number of pieces, ...

Qualities

- Represent a state, a status or a mode
- Cannot be aggregated (in contrast to fact data)
- Example: shipping mode, status

Attributes

- Describe dimension objects
- Mostly text-based descriptions
- Example: product descriptions, customer profiles, addresses

Dimensions

- Strongly associated attributes
- Typically 5 to 20 attributes
- Showing mostly hierarchical relationships
- Example: product information, customer information, time references

Conformed Dimensions

A conformed dimension is a dimension that means the same thing with every possible fact to which it can be joined.

- A conformed dimension is identically the same dimension in each data mart.
- Conformed dimensions support:
 - A single dimension can be used against multiple facts in the same database space.
 - User interfaces and data content are consistent whenever the dimension is used.
 - There is a consistent interpretation of attributes and, therefore, rollups across data marts.

Dimensional Fact Model (DFM)

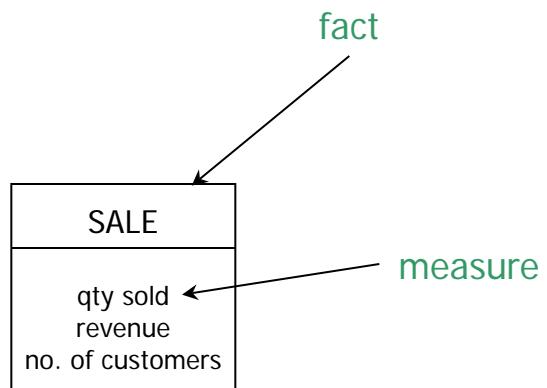
- Part of work on a complete and consistent design methodology.
- Multidimensional conceptual model including a graphical representation.
- Representation of reality consists of a **set of fact schemes**.
- Basic elements of a fact scheme $f = (M, A, N, R, O, S)$ are **facts**, **measures**, **attributes**, **dimensions**, and **hierarchies**.
- Other features which may be represented on fact schemes are the **additivity** of fact attributes along dimensions, the **optionality** of dimension attributes, and the existence of **non-dimension attributes**.
- The multidimensional model may be mapped on the logical level differently depending on the underlying DBMS.

DFM: Basic Definitions

- Let $g=(V, E)$ be a directed, acyclic and weakly connected graph.
- g is a **quasi-tree** with root in $v_0 \in V$ if each other vertex $v_j \in V$ can be reached from v_0 through at least one directed path. $\text{path}_{0j}(g) \subseteq g$ denotes a directed path starting in v_0 and ending in v_j .
- Given $v_i \in \text{path}_{0j}(g)$,
 - $\text{path}_{ij}(g) \subseteq g$ denotes a directed path starting in v_i and ending in v_j ,
 - $\text{sub}(g, v_i) \subset g$ denotes the quasi-tree rooted in $v_i \neq v_0$.

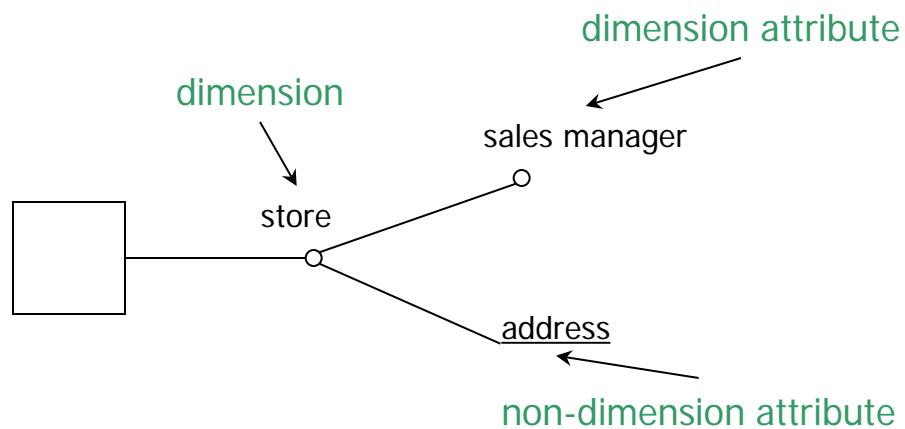
DFM: Facts and Measures

- A **fact** is a focus of interest for the decision-making process. It models an event occurring in the enterprise world (e.g. sales and shipments).
- **M** is a set of measures. Each **measure** $m_i \in M$ is defined by a numeric or Boolean expression which involves values acquired from the operational information systems.
- Graphical representation:
 - A fact is represented by a box which reports the fact name and, typically, one or more measures.



DFM: Attributes and Dimensions

- **Dimensions** are discrete attributes which determine the minimum granularity adopted to represent facts.
- **A** is a set of **dimension attributes**.
 - Each dimension attribute $a_i \in A$ is characterized by a discrete domain of values, $\text{Dom}(a_i)$.
- **N** is a set of **non-dimension attributes**.
 - A non-dimension attribute contains additional information about a dimension attribute and is connected by a -to-one relationship.
 - Unlike dimension attributes, non-dimension attributes cannot be used for aggregation.
- Graphical representation:
 - Dimension attributes are represented by circles, non-dimension attributes by lines.

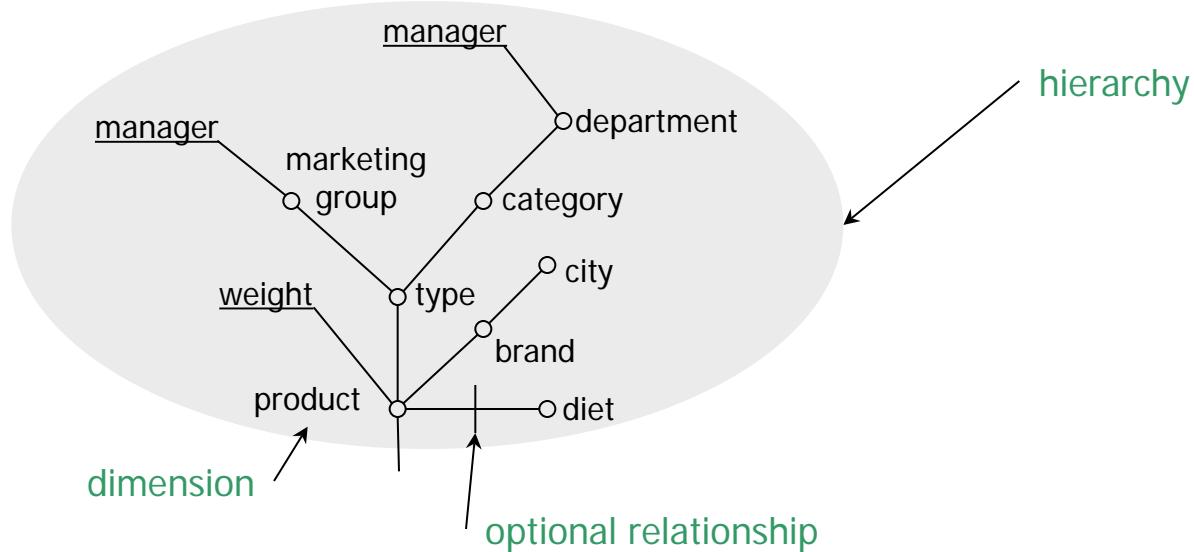


DFM: Hierarchies

- **Hierarchies** are made up of discrete dimension attributes linked by -to-one relationships, and determine how facts may be aggregated and selected significantly for the decision-making process.
 - **R** is a set of ordered tuples, each having the form (a_i, a_j) where $a_i \in A \cup \{a_0\}$ and $a_j \in A \cup N$ ($a_i \neq a_j$), such that the graph $qt(f) = (A \cup N \cup a_0, R)$ is a quasi-tree with root a_0 .
 - a_0 is a **dummy attribute** playing the role of the fact f on which the scheme is centered. The couple (a_i, a_j) models a -to-one relationship between attributed a_i and a_j .
 - Each element in $\text{Dim}(f) = \{a_i \in A \mid (a_0, a_i) \in R\}$ is a **dimension**.
 - The **hierarchy** on dimension $d_i \in \text{Dim}(f)$ is the quasi-tree rooted in d_i : $\text{sub}(qt(f), d_i)$.
 - $O \subset R$ is a set of **optional relationships**. The domain of each dimension attribute a_j such that $\exists (a_i, a_j) \in O$ includes a NULL value.

DFM: Hierarchies

- Graphical representation:
 - Subtrees rooted in dimensions are hierarchies. The arc connecting two attributes represents a -to-one relationship between them.
 - Optional relationships are represented by marking with a dash the corresponding arc.
- Example:



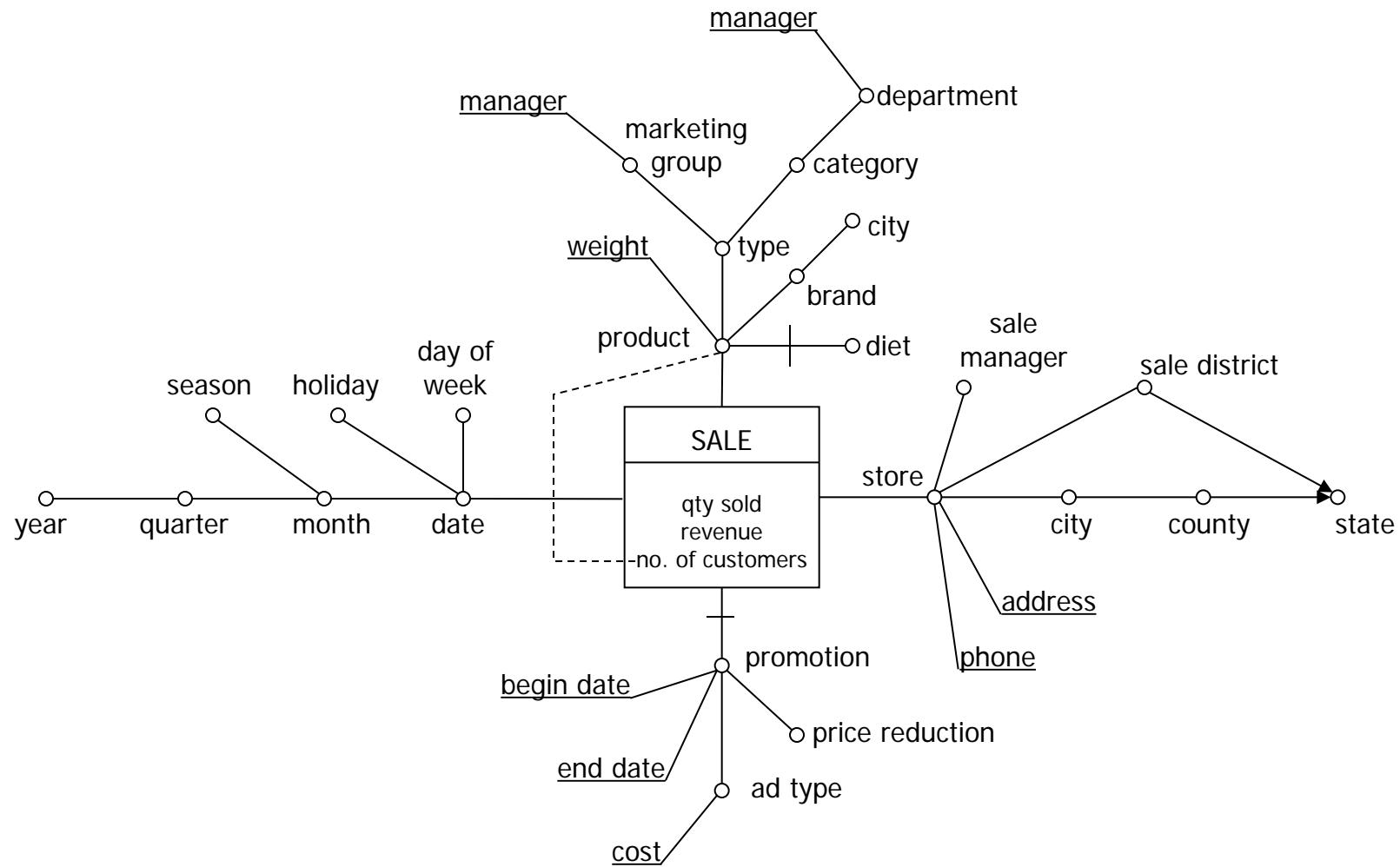
DFM: Aggregation

- **S** is a set of **aggregation statements**.
 - Each aggregation statement consists of a triple (m_j, d_i, Ω) where $m_j \in M$, $d_i \in \text{Dim}(f)$ and $\Omega \in \{\text{'SUM'}, \text{'AVG'}, \text{'COUNT'}, \text{'MIN'}, \text{'MAX'}, \text{'AND'}, \text{'OR'}, \dots\}$ (aggregation/grouping operator).
 - Statement $(m_j, d_i, \Omega) \in S$ declares that measure m_j can be aggregated along dimension d_i by means of the operator Ω .
 - If no aggregation statement exists for a given pair (m_j, d_i) , then m_j cannot be aggregated at all along d_i .

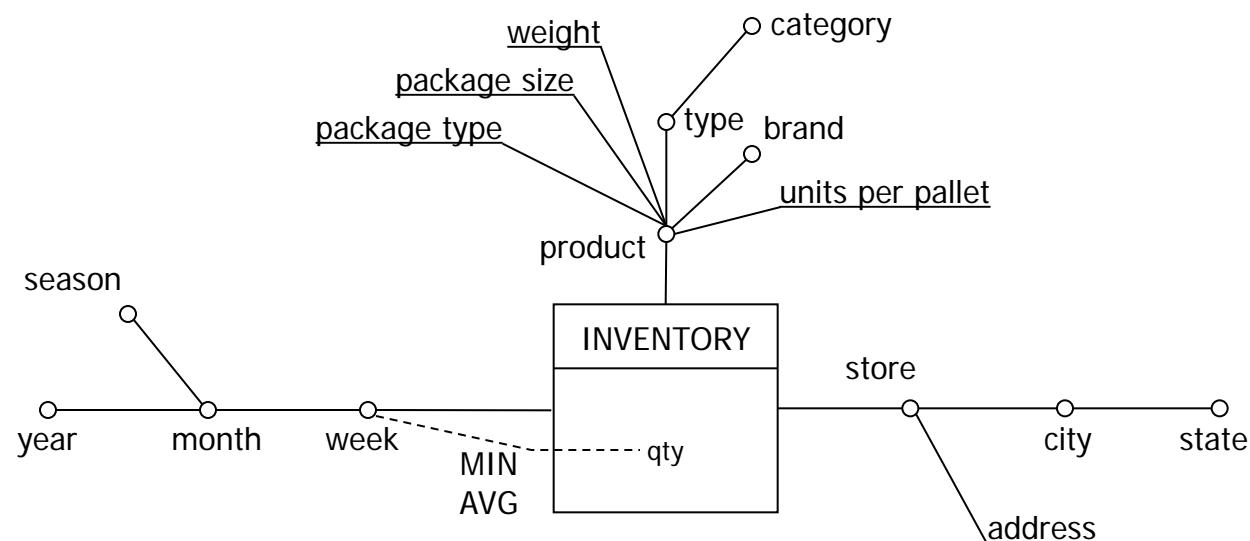
DFM: Aggregation

- A measure is **additive** on a dimension if its values can be aggregated along the corresponding hierarchy by the 'SUM' operator.
- Graphical representation:
 - If $(m_j, d_i, \text{'SUM'}) \notin S$, m_j and d_i are connected by a dashed line labelled with all aggregation operators Ω such that $(m_j, d_i, \Omega) \in S$.
 - If $(m_j, d_i, \text{'SUM'}) \in S$:
 - If $\neg \exists \Omega \neq \text{'SUM'} \mid (m_j, d_i, \Omega) \in S$, m_j and d_i are not graphically connected.
 - Otherwise, m_j and d_i are connected by a dashed line labelled with the symbol '+' followed by all the other operators $\Omega \neq \text{'SUM'}$ such that $(m_j, d_i, \Omega) \in S$.

SALE Fact Schema



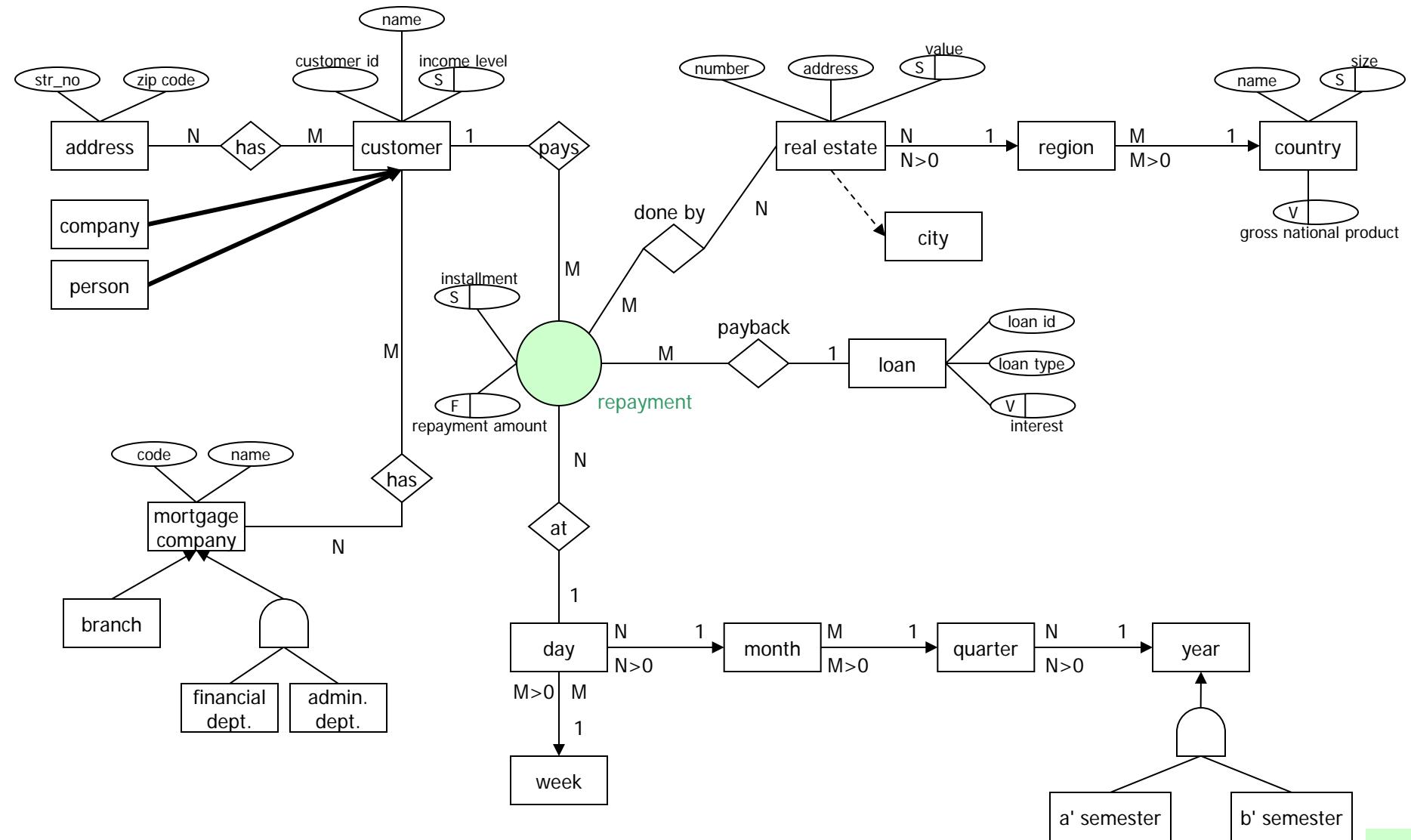
INVENTORY Fact Schema



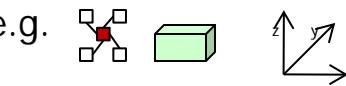
starER

- Combines the star structure with the semantically rich constructs of the ER model.
- Adds special types of relationships to support hierarchies.
- starER vs. DFM:
 - starER allows many-to-many relationships between dimensions and facts.
 - starER allows objects participating in the data warehouse, but not in the form of a dimension.
 - Specialized relationships on dimensions are permitted in starER (specialization/generalization).

starER: Mortgage Company Data Warehouse



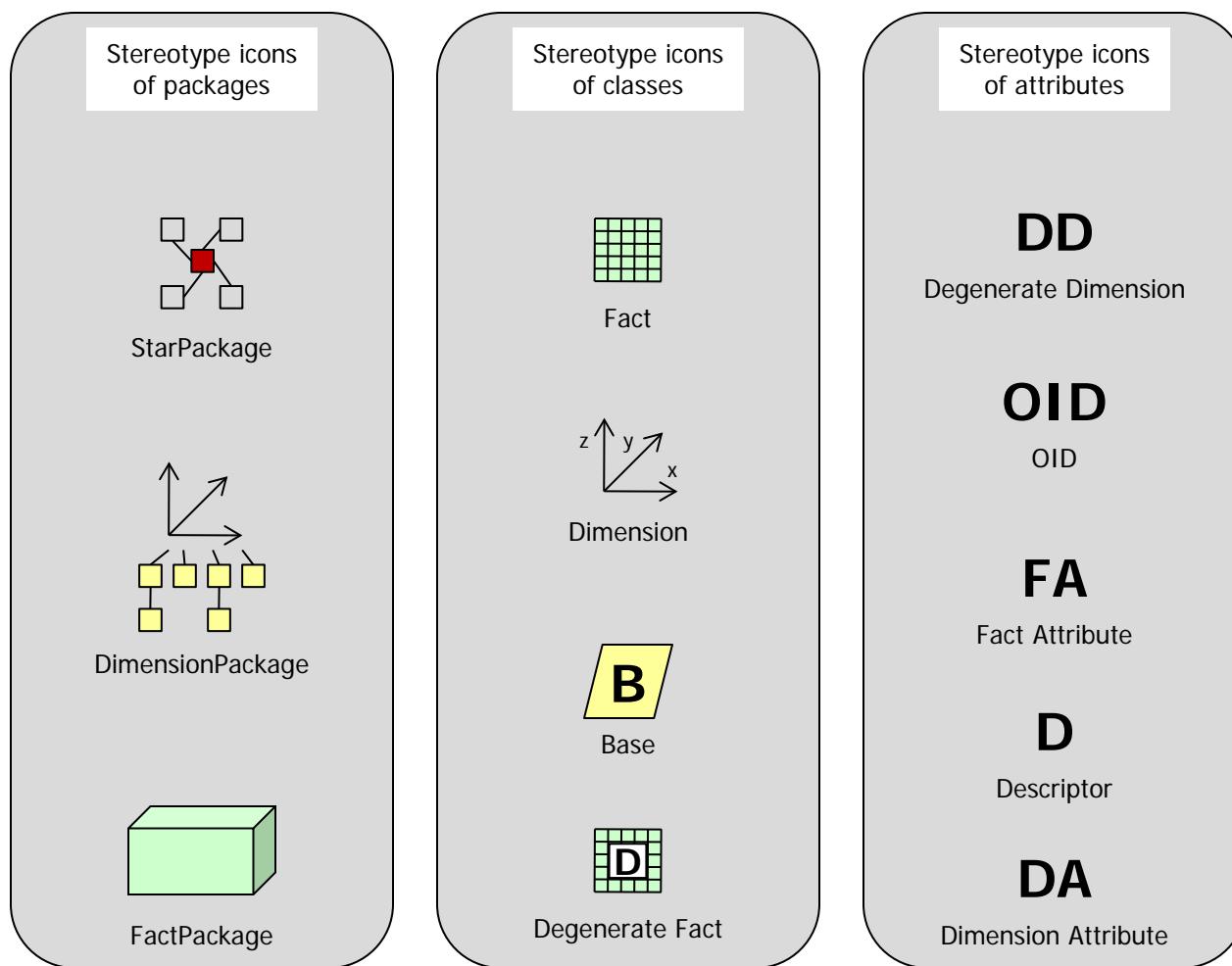
UML Profile for Multidimensional Modeling

- UML profile: extend UML by
 - stereotypes represented by icons
e.g. 
 - tagged values
e.g., {name = value}
 - constraints, e.g.,
{Quantity Is Not SUM Along Salesperson}

Main characteristics:

- Accurate
 - major features of multidimensional models supported
- No redundancy
 - concepts and elements are only defined once in the model and imported where needed
- Simplicity
 - minimal subset of UML and minimal extensions
- Understandable
 - define three levels of abstraction
 - use packages as grouping mechanism
 - support conformed dimensions

UML Profile: Overview



Three Levels of Detail

Level 1 Model definition

- A package represents a star schema
- A dependency between two packages indicates that star schemas share at least one dimension

Level 2: Star schema definition

- A package represents a fact or a dimension
- A dependency between two packages indicates that they share at least one level of the dimension hierarchy

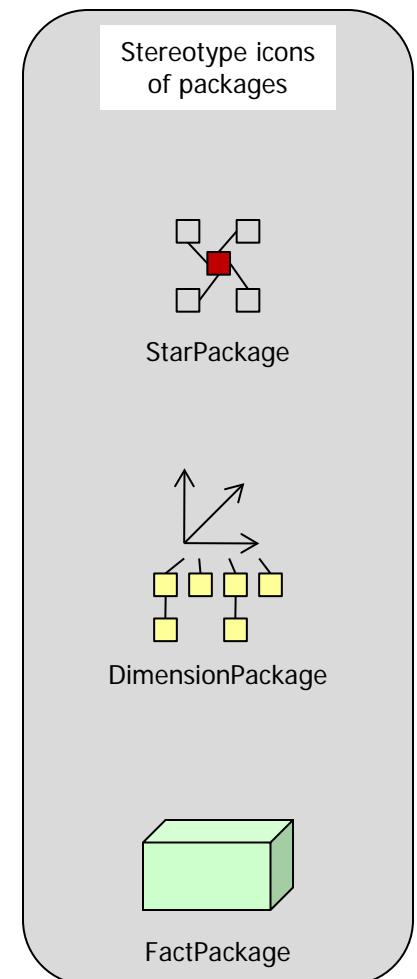
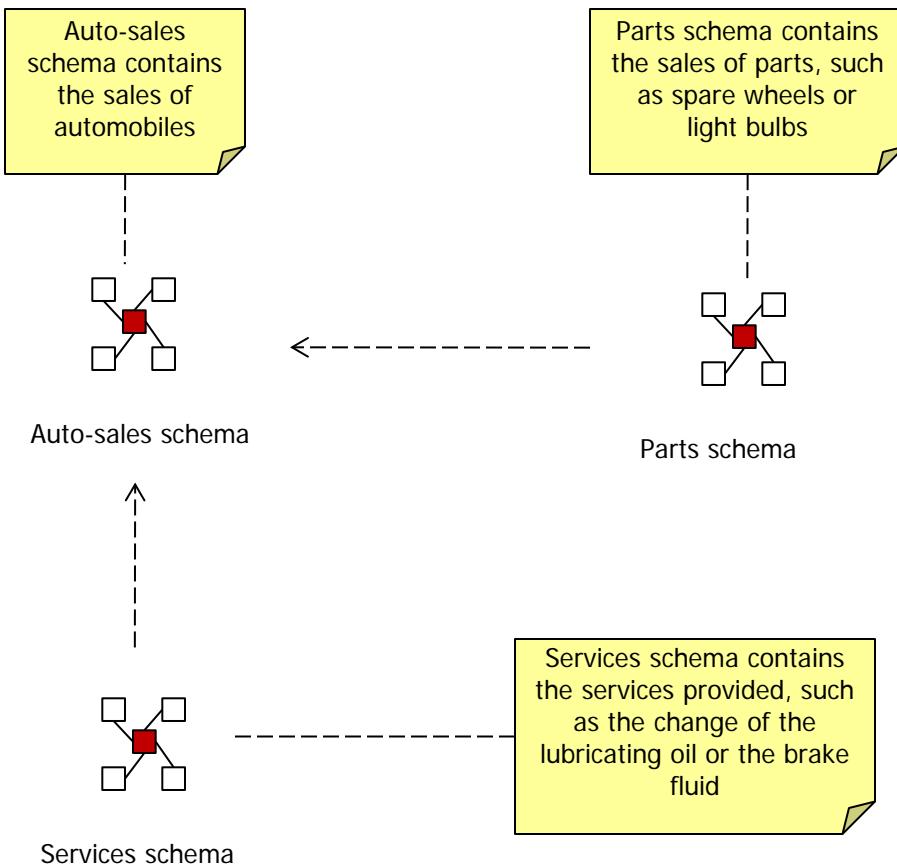
Level 3: Dimension Definition fact definition

- Set of classes representing the hierarchy levels, or
- set of classes representing the entire star schema

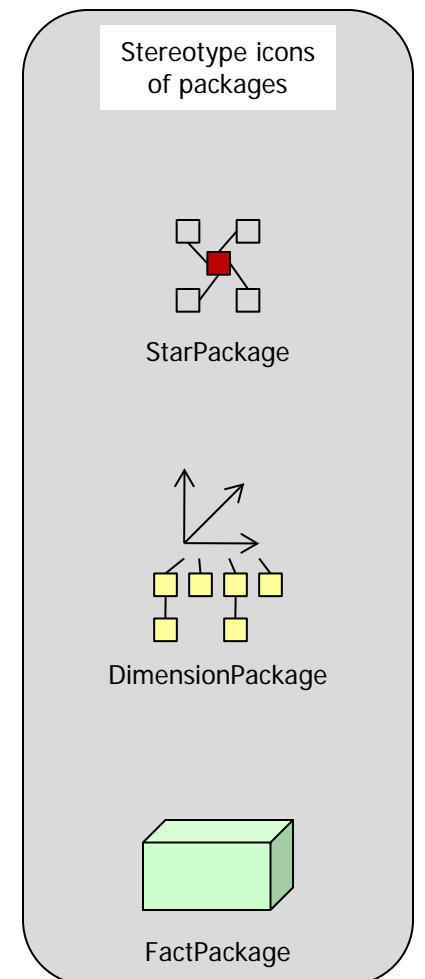
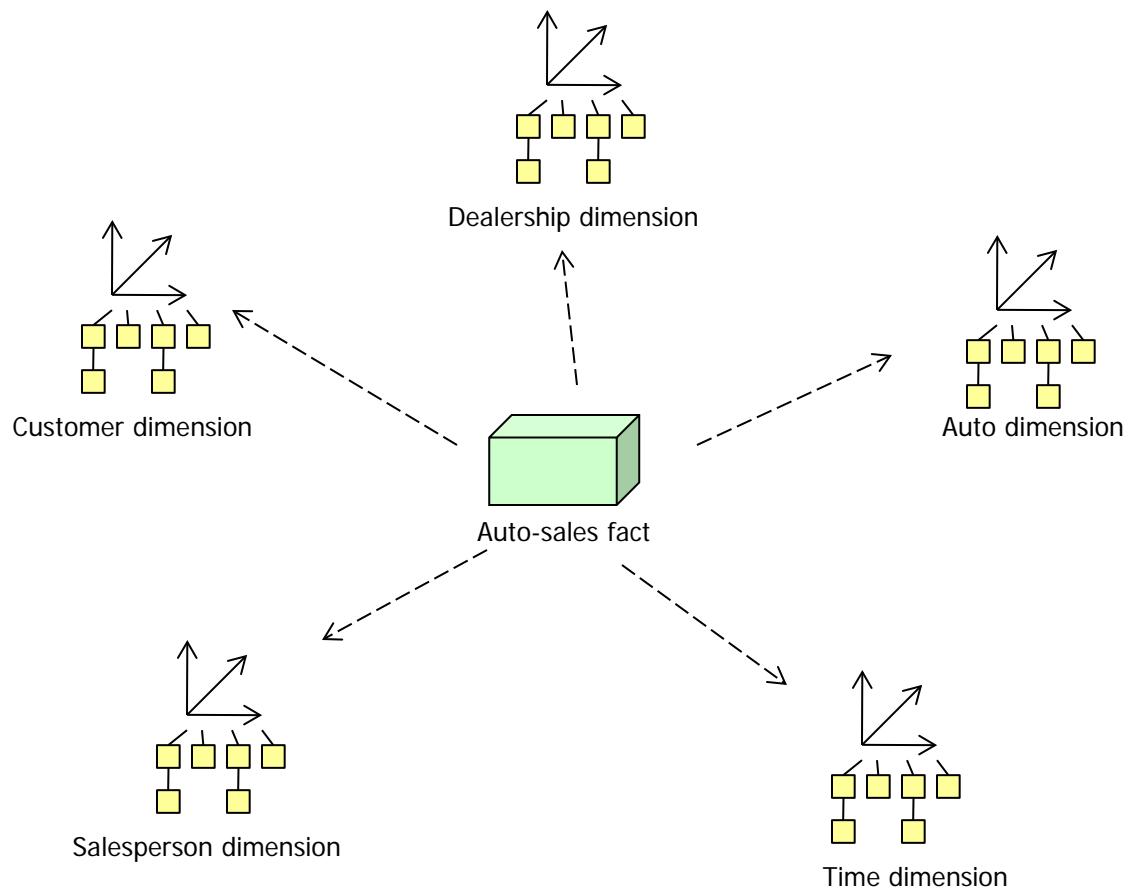
Running Example

- Company having several dealerships that sell automobiles (cars and vans) across several states
- Data warehouse should support analysis of
 - sales of automobiles
 - sales of parts such as spare wheels or light bulbs
 - service works such as change of lubricating oil or brake oil

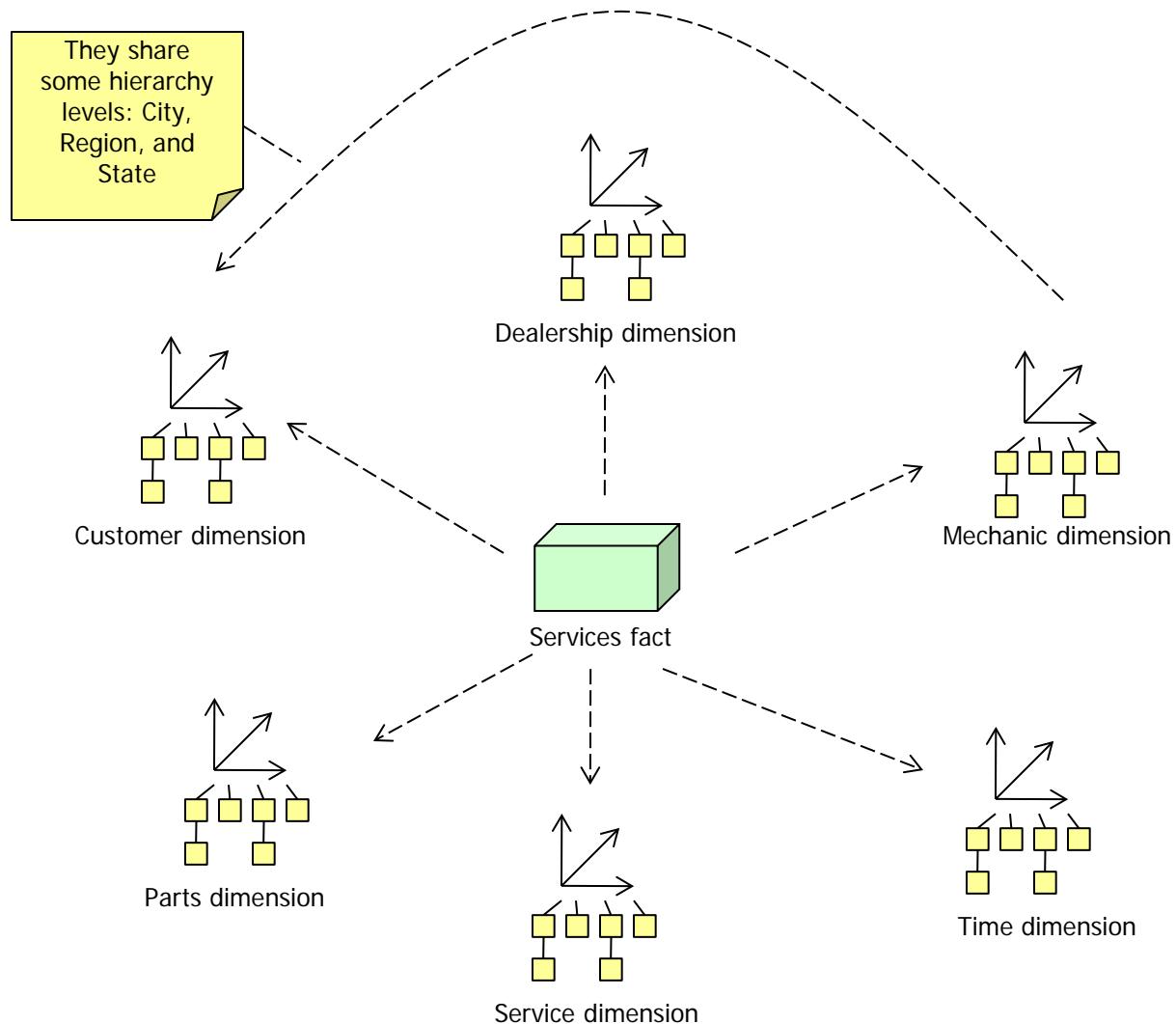
Level 1: Model Definition



Level 2: Auto-Sales Schema

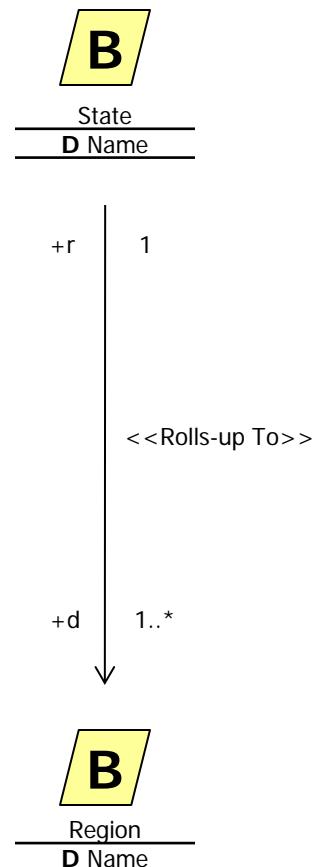


Level 2: Services Schema

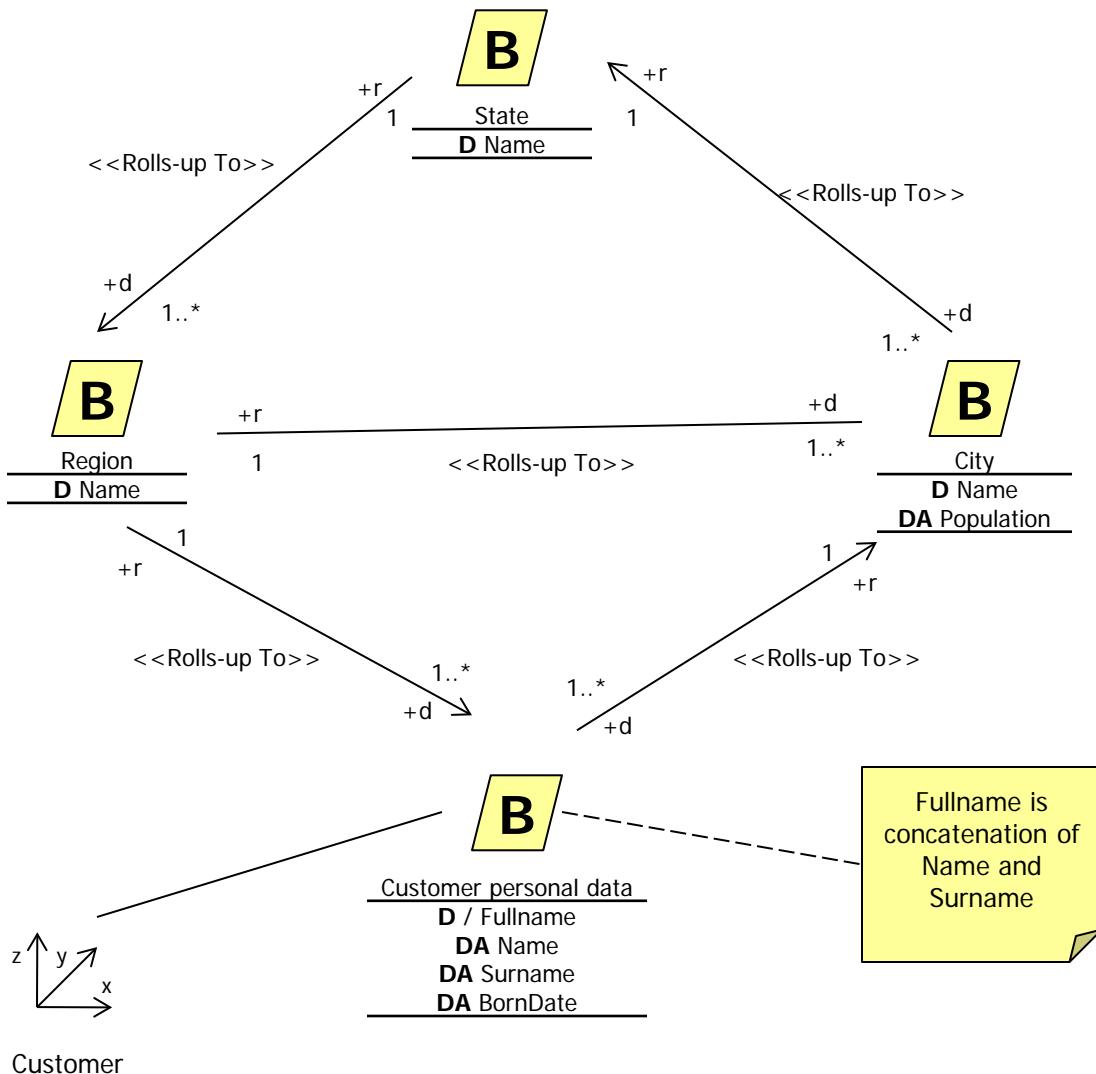


Dimension Hierarchies

- Classification hierarchy levels are specified by base classes 
- Associations show hierarchical relationship
 - drill-down direction is marked by +d
 - roll-up direction is marked by +r
 - default roll-up or drill-down directions are marked by arrows
- Multiplicity may be defined for each association
- Various attribute types:
 - DA:** dimension attribute
 - D:** description
 - OID:** object id



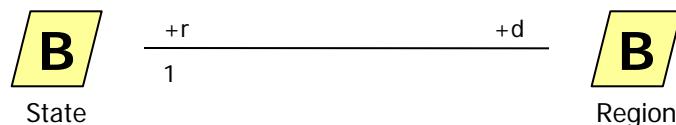
Level 3: Customer Dimension



Types of Classification Hierarchies

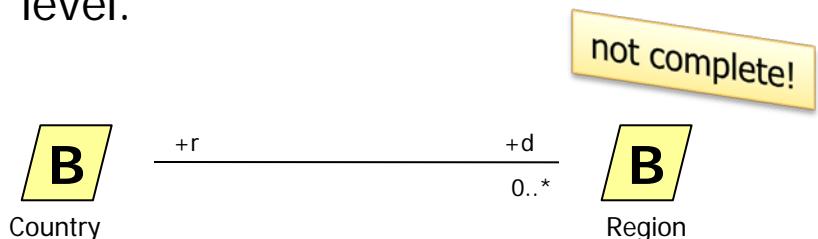
- **Strict hierarchy:**

Each object of the lower level (+d) belongs to only one object at the higher level (+r).



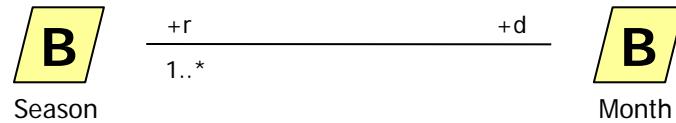
- **Completeness for drill-down:**

For each object at the higher level there exists an object at the lower level.



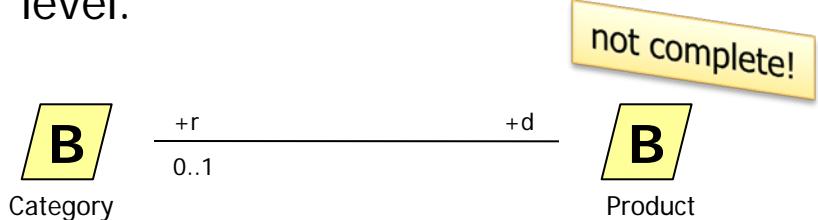
- **Non-strict hierarchy:**

An object of the lower level (+d) may belong to several objects at the higher level (+r).



- **Completeness for roll-up:**

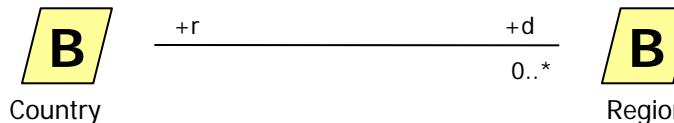
For each object at the lower level there exists an object at the higher level.



Non-strict and incomplete hierarchies may yield to inconsistent totals!

Summarizability Issues

- Drill-Down



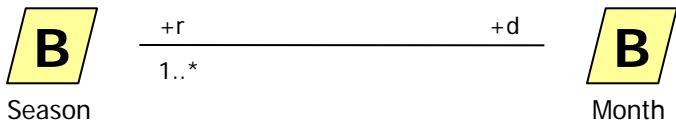
Country	Sales
Germany	50
France	40
Andorra	15

Total = 105

Region	Sales
NRW	20
BW	10
BAY	20
Paris	30
Province	10

Total = 90

- Non-strict



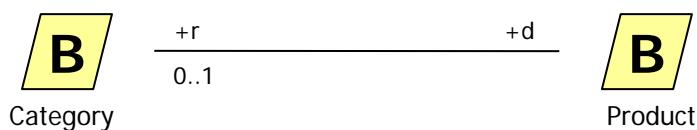
Season	Sales
Winter	560
Spring	500
Summer	100

Total = 1160

Month	Sales
Jan	200
Feb	180
Mar	180
Apr	160
May	160
Jun	100

Total = 980

- Roll-up



Category	Sales
Dairy	55
Bakery	25

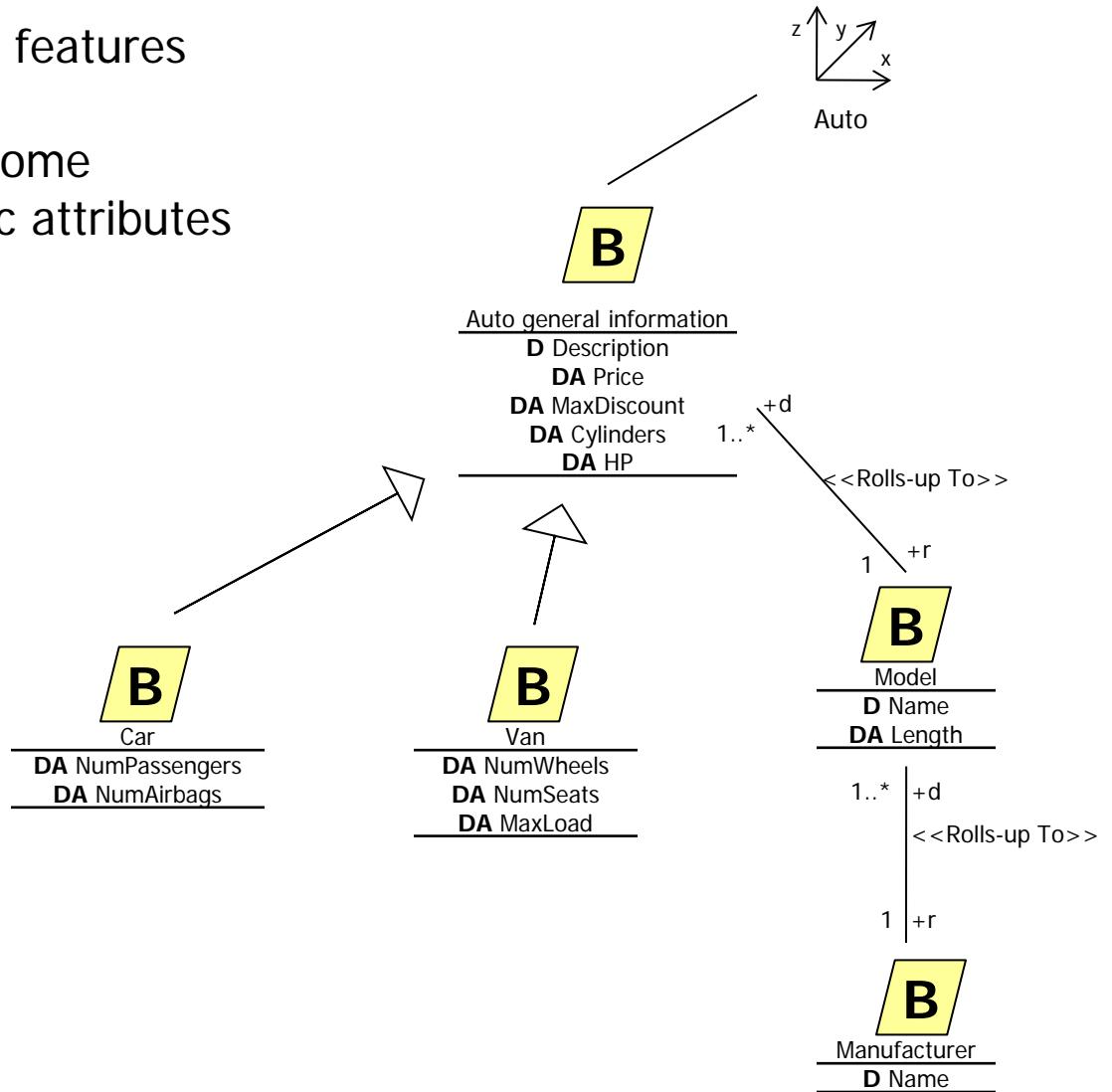
Total = 80

Product	Sales
Milk	15
Butter	10
Bread	25
Yogurt	30
Glasses	100

Total = 180

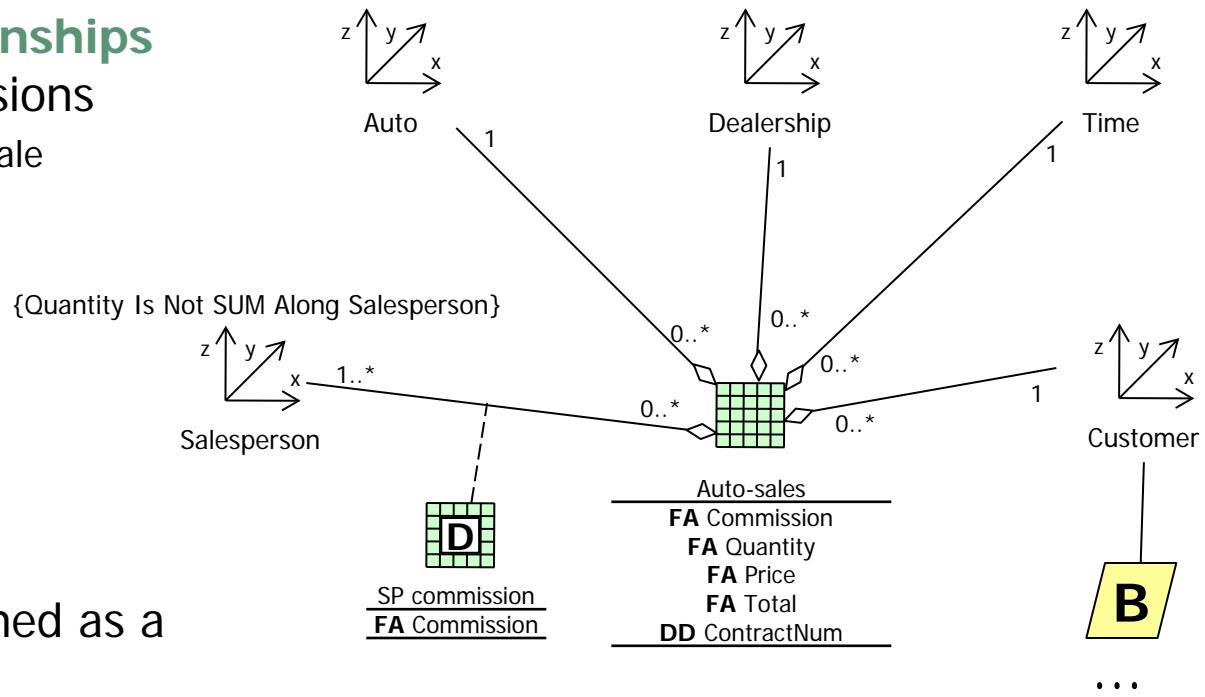
Categorization Hierarchies

- Allows to model additional features for subtypes of a class, e.g., cars and vans have some common and some specific attributes



Level 3: Auto-Sales Fact

- **Many-to-many-Relationships** between facts and dimensions
 - Several salespersons for a sale



- **Summarizability** is defined as a constraint

- **Degenerate Fact:**
- for m:n-relationships between facts and dimensions
- specific attribute that is provided for each instance combination in such a relationship

- **Degenerate Dimension:**
- Most of the properties are already presented by other elements (facts, dimensions)
- Remaining attributes are necessary to uniquely identify fact instances.

Important Features of a Conceptual Model

- Explicit hierarchies in dimensions
- Multiple hierarchies in a dimension
- Non-strict and non-complete hierarchies
- Conformed dimensions

Dimensions

- Support correct aggregation (summarizability)
- Degenerated facts
- Degenerated dimensions

Facts

- Many-to-many relationships between facts and dimensions
- Symmetric treatment of dimensions and measures

Facts and Dimensions

Others

- Changes of data over time
- Reflect uncertainty
- Different levels of granularity

Overview

- Data Warehouse Design Process
- Conceptual Design
- Logical Design
 - TPCH Benchmark Schema
 - Star Schema
 - Snowflake Schema
 - Informix Schema
- Details of Logical Design
- Physical Design

Logical Design

Convert the conceptual schema to a logical one with respect to the target logical data model.

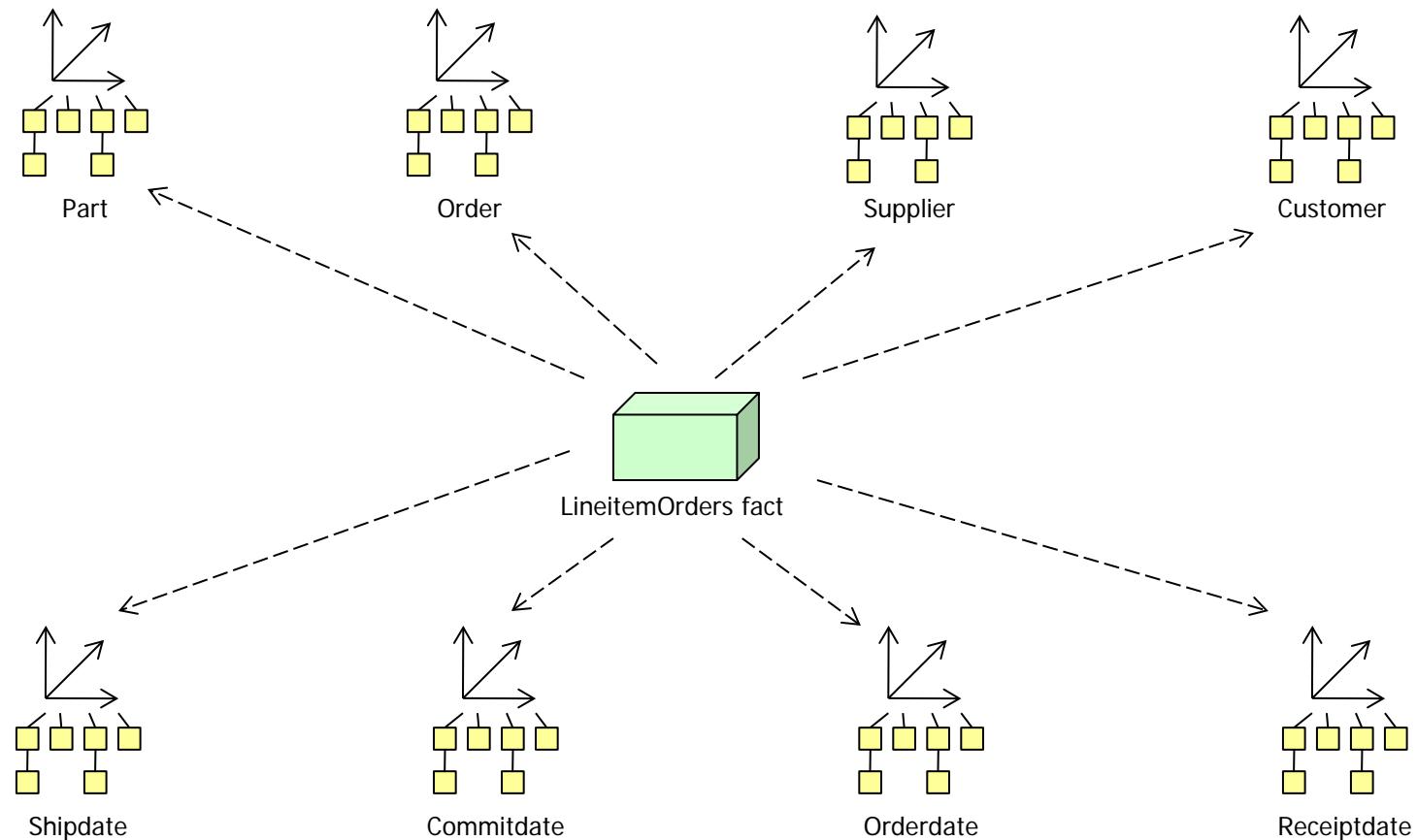
- Logical Design is based on:
 - conceptual diagrams
 - summarizability constraints
 - transformation rules
- Logical data models:
 - relational
 - multidimensional

Sample Scenario



- Company that manages, sells, or distributes a product worldwide (e.g., car rental, food distribution, parts, suppliers, etc.).
- Scenario taken from TPC-H benchmark
- Each order entry refers to a part, an order, a customer, a supplier and the dates for the ordering, the commitment, the shipment and the receipt.
- Order entries are further characterized by a line number, a status, shipment instructions and a return flag.

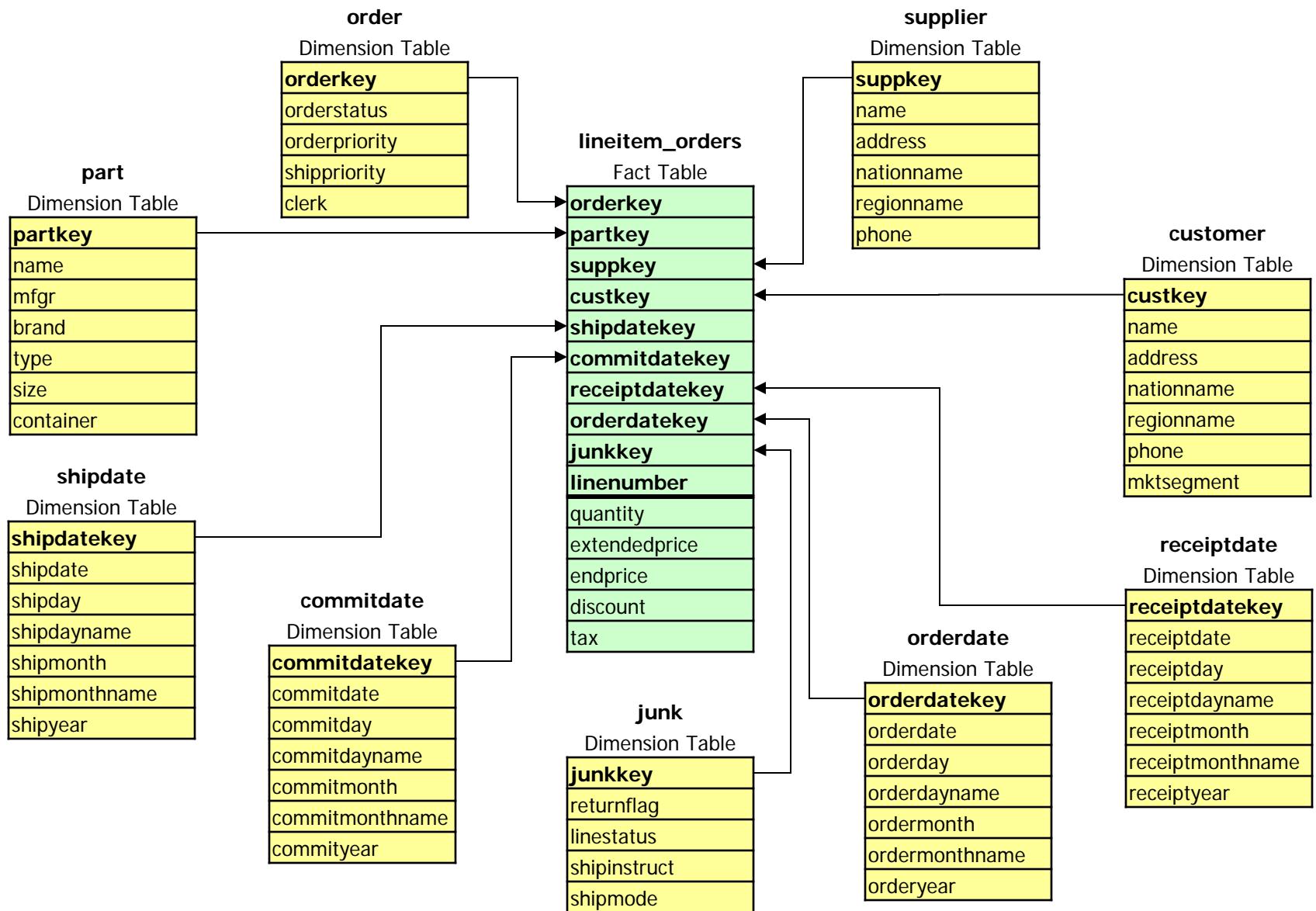
Conceptual Schema: Overview



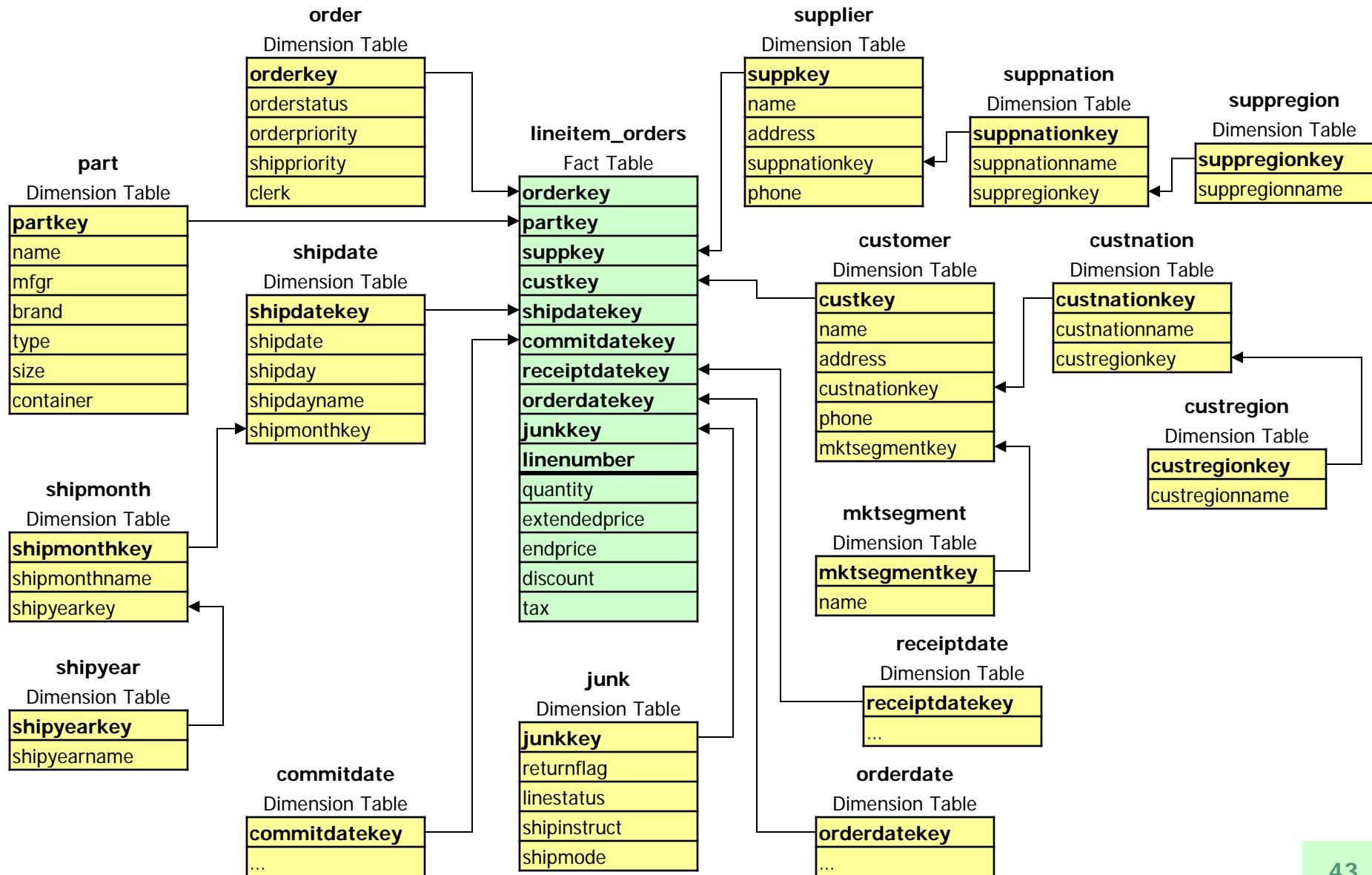
Logical Schema Types

- **Star Schema**
 - Each dimension is modeled by a single table
 - Redundancy within the dimension tables, e.g. nationname, regionname
 - Qualities are combined into one or a few 'junk' dimensions
- **Snowflake Schema**
 - Dimensions get normalized (3NF)
 - One dimension table per hierarchy level
- **Informix Schema**
 - Normalized attribute tables
 - Hierarchies are partly represented in dimension tables

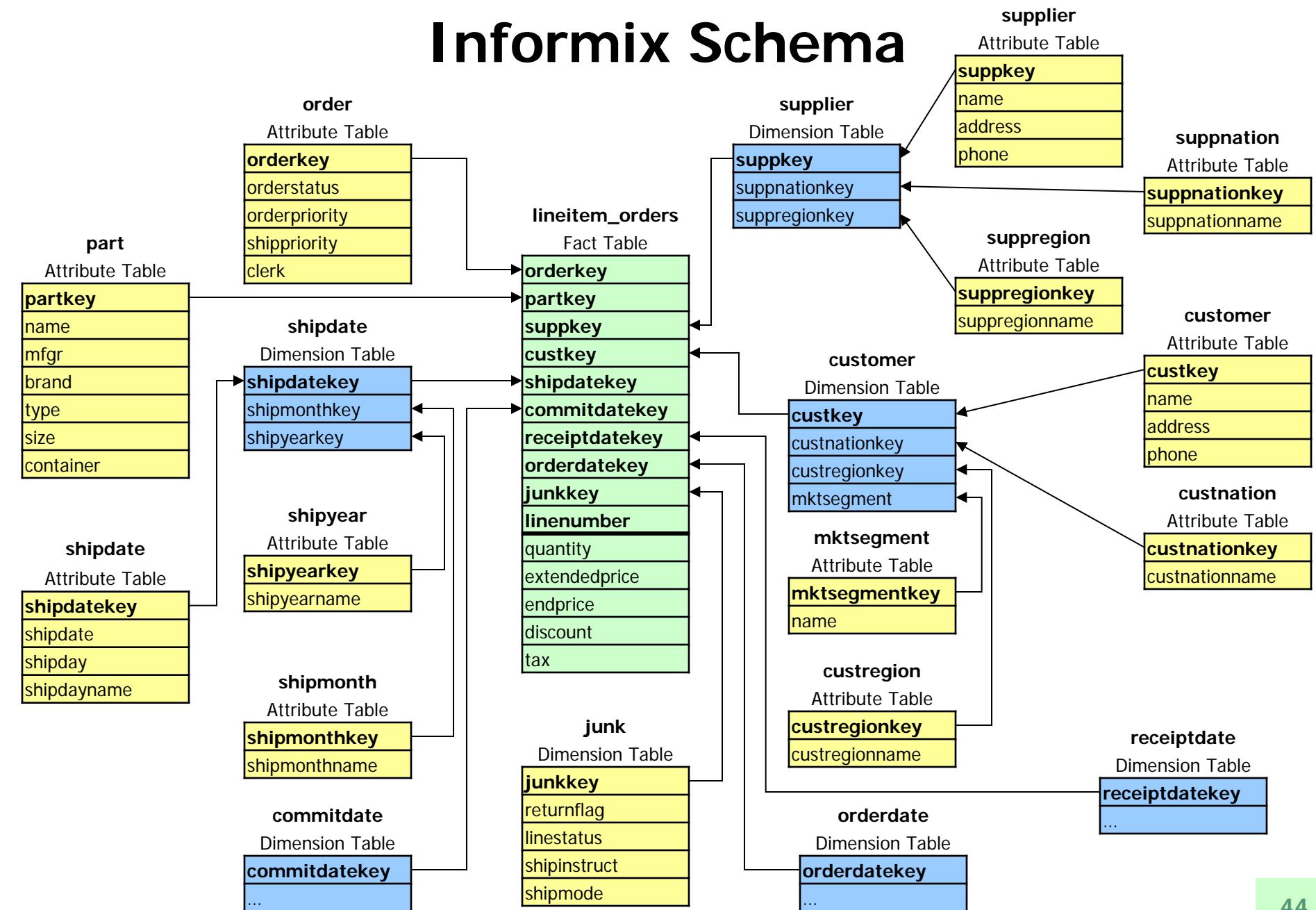
Star Schema



Snowflake Schema



Informix Schema



Comparing Logical Schema Types

- Is all the information of the conceptual model representable in the logical schema?
 - Explicit hierarchies in dimensions
 - Multiple hierarchies in a dimension
 - Non-strict and non-complete hierarchies
 - Many-to-many relationships between facts and dimensions
 - Symmetric treatment of dimensions and measures
 - Support correct aggregation (summarizability)
 - What effort is needed to reflect changes of the conceptual design in the logical schema:
 - insert hierarchy level
 - delete hierarchy level
 - insert measure
 - delete measure
 - insert dimension
 - delete dimension
 - modify granularity
- logical schema?
- metadata?

Comparison

	Star Schema	Snowflake Schema	Informix Schema
Clearness	one table per dimension	multiple tables per dimension	multiple tables per dimension
Redundancy	in the dimension table	normalization takes care of	normalization takes care of
Data Volume	high(er)	low	low
Hierarchies	not represented	represented in dimension tables	only one hierarchy
Summarizability	-	-	-
Adding a Dimension	one additional table	several additional tables	several additional tables
Adding an Attribute	one attribute appended to dimension table	changes to several tables	changes to several tables
Performance (Queries)	max. one join per dimension	several joins among the dimension tables	several joins among the dimension tables

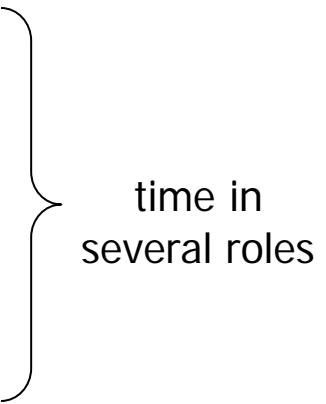
Overview

- Data Warehouse Design Process
- Conceptual Design
- Logical Design
- Details of Logical Design
 - Extended Dimension Table Design
 - Production keys / surrogate keys
 - Roles of Dimensions
 - Hierarchies
 - Slowly changing dimensions
 - Time stamping in large dimensions
 - Large dimensions with frequent changes
 - Many-to-Many Dimensions
 - Time Dimensions
 - Extended Fact Table Design
 - Modeling Events and Coverage (Factless Fact Tables)
 - Multinational Currency Tracking
- Physical Design

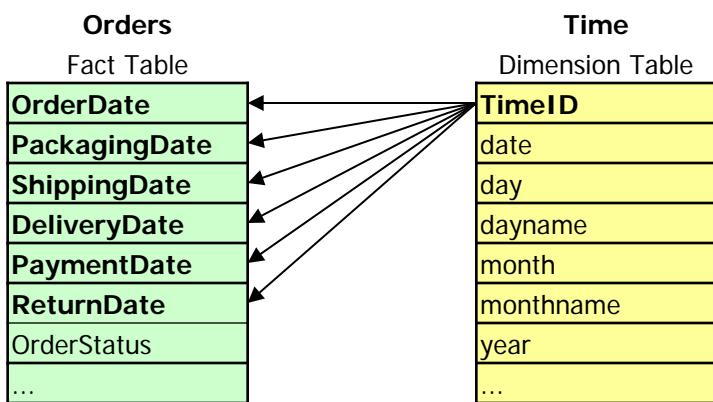
Production Keys / Surrogate Keys

- **Production key:** Attributes that make up a primary key in the production system (source system).
 - Keys may be reused in the production system.
 - Rules for building production keys may change.
 - Production keys are kept although the dimension has changed.
- **Surrogate key:** Single attribute (INTEGER) that is used as primary key in the data warehouse.
 - Results in small keys.
 - Surrogate keys have no meaning.
 - Keys may be changed independent of the source system
(this is especially important for type two slowly changing dimensions)

Roles of Dimensions

- In some situations a single dimension appears several times in the same fact table.
 - **Example:**
A fact table on customer orders may include:
 - Order date
 - Packaging date
 - Shipping date
 - Delivery date
 - Payment date
 - Return date
 - Order status
 - Customer
 - ...
 - Problems:
 - Using the same dimension for each role may result in wrong query results.
 - Result may contain many columns with identical names.
- 
- time in several roles

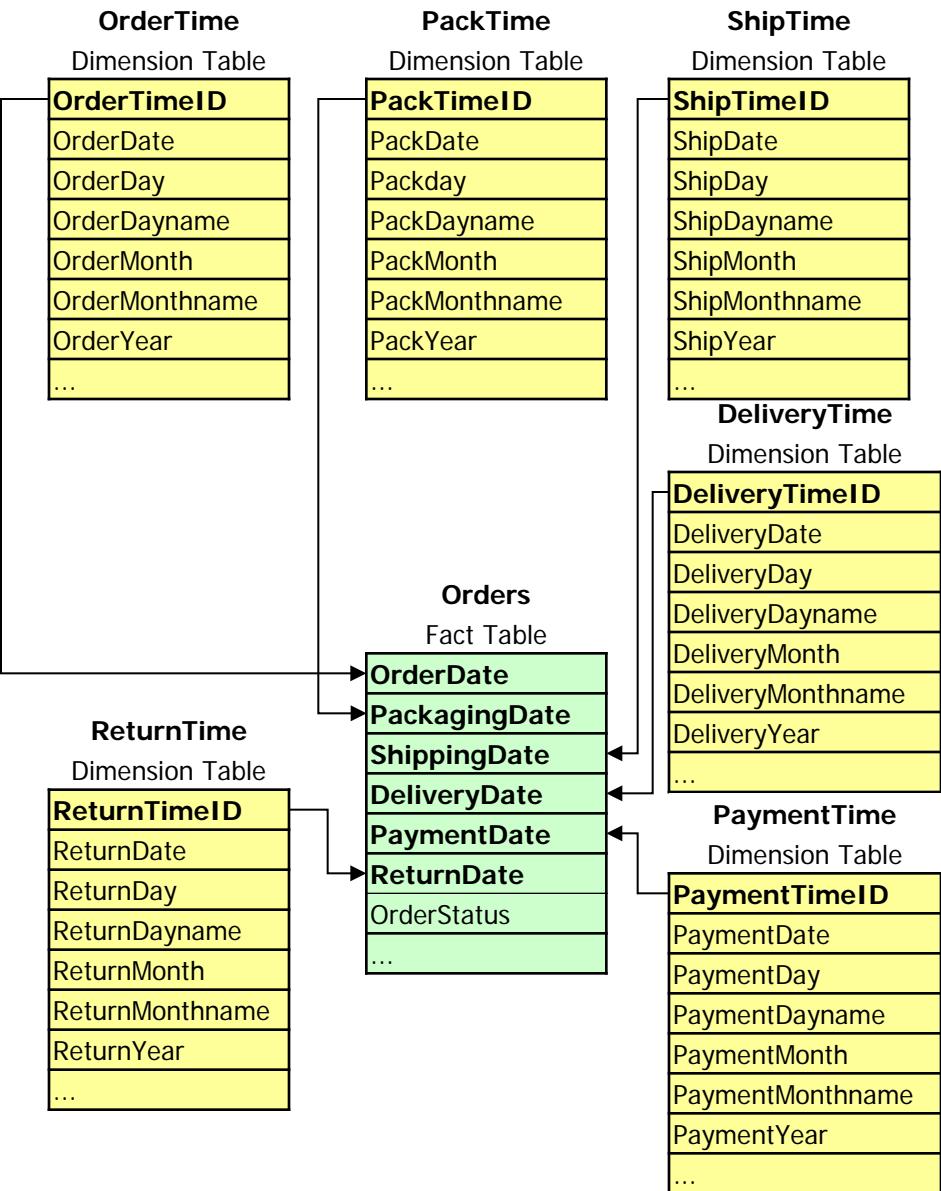
Single Dimension Tables for all Roles



```
SELECT      ...
FROM        Orders AS O,
Time AS T
WHERE       O.OrderDate = T.TimeID
AND         O.PackagingDate = T.TimeID
AND         O.ShippingDate = T.TimeID
AND         O.OrderStatus = 'completed'
GROUP BY   ...
```

```
SELECT      ...
FROM        Orders AS O,
Time AS OT,
Time AS PT,
Time AS ST
WHERE       O.OrderDate = OT.TimeID
AND         O.PackagingDate = PT.TimeID
AND         O.ShippingDate = ST.TimeID
AND         O.OrderStatus = 'completed'
GROUP BY   ...
```

Single Dimension Table for each Role

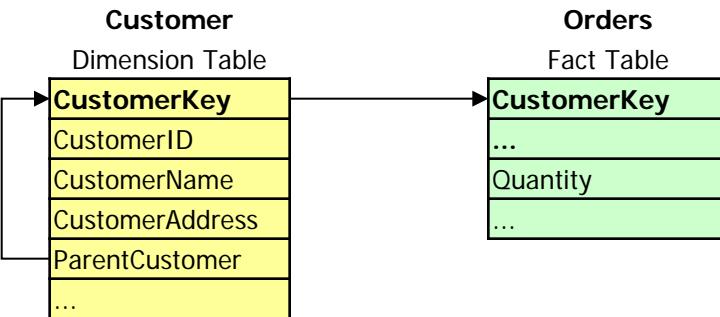


```

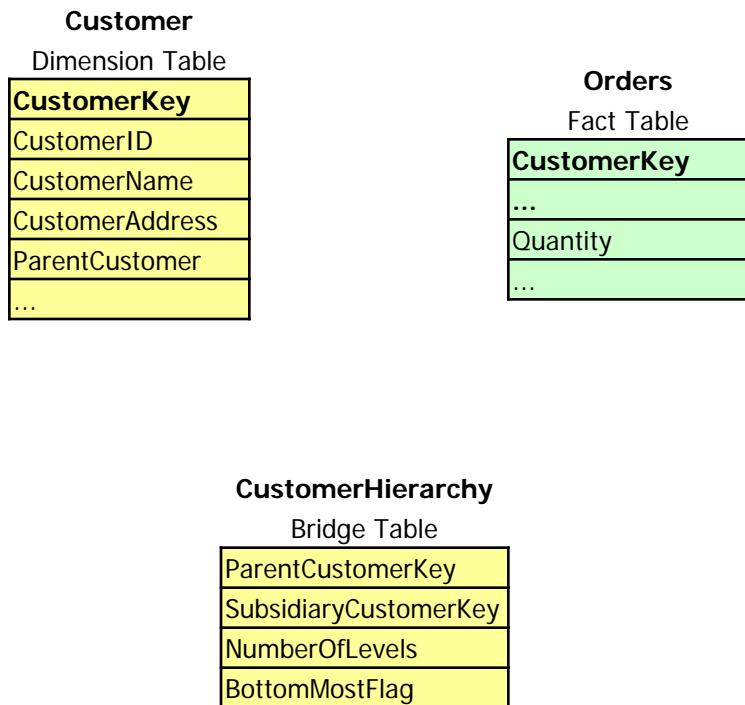
SELECT      ...
FROM        Orders AS O,
          OrderTime AS OT,
          PackTime AS PT,
          ShipTime AS ST
WHERE       O.OrderDate = OT.OrderTimeID
AND         O.PackagingDate =
          PT.PackTimeID
AND         O.ShippingDate = ST.ShipTimeID
AND         O.Orderstatus = ' completed'
GROUP BY   ...
  
```

Role-playing dimension tables can be provided as physical tables or as views.

Hierarchies

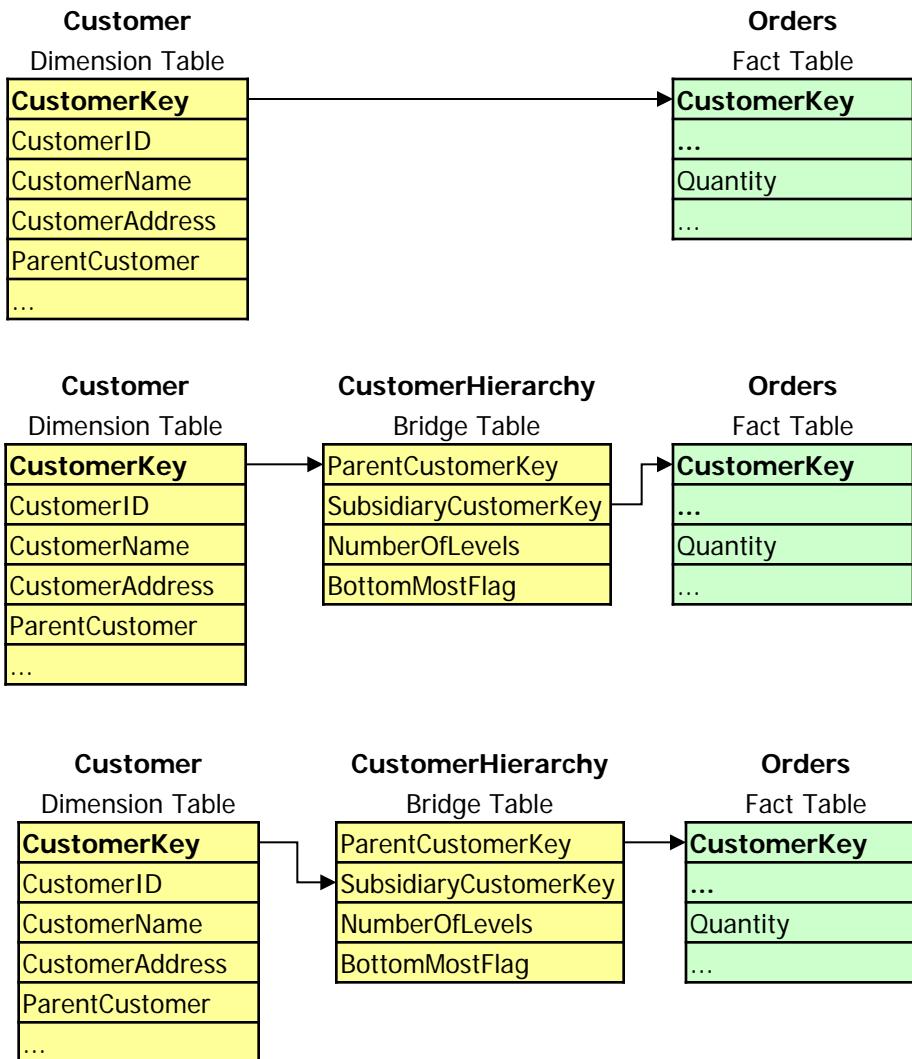
- Organization hierarchies
 - Parts explosion hierarchies
 - Compact way to represent the hierarchy.
 - Structure results in complex queries.
 - Goals:
 - Keep the original grain of the customer dimension.
 - Support aggregation for the entire hierarchy.
 - Support aggregation for the immediate subsidiaries as well as for the lowest-level subsidiaries.
 - Support the efficient search for the immediate parent or the top-most parent.
- 
- The diagram illustrates a data warehouse architecture. On the left, a 'Customer' dimension table is shown as a vertical stack of columns: CustomerKey (highlighted in yellow), CustomerID, CustomerName, CustomerAddress, ParentCustomer, and an ellipsis. An arrow points from this table to the right. On the right, an 'Orders' fact table is shown as a vertical stack of columns: CustomerKey (highlighted in yellow), an ellipsis, Quantity, and another ellipsis. The CustomerKey column in both tables is highlighted in yellow, indicating it is a primary key that links the two entities.

Bridge Table for Hierarchies



- Dimension table and fact table are kept .
- Bridge table contains:
 - All combinations of CustomerKeys and all their SubsidiaryCustomers
 - Path-length of all combinations including paths of length zero.
 - A flag identifying the bottom level of the hierarchy.

Usage of Bridge Tables



- Aggregation ignoring the customer hierarchy.
- Descending the customer hierarchy.
- Ascending the customer hierarchy.

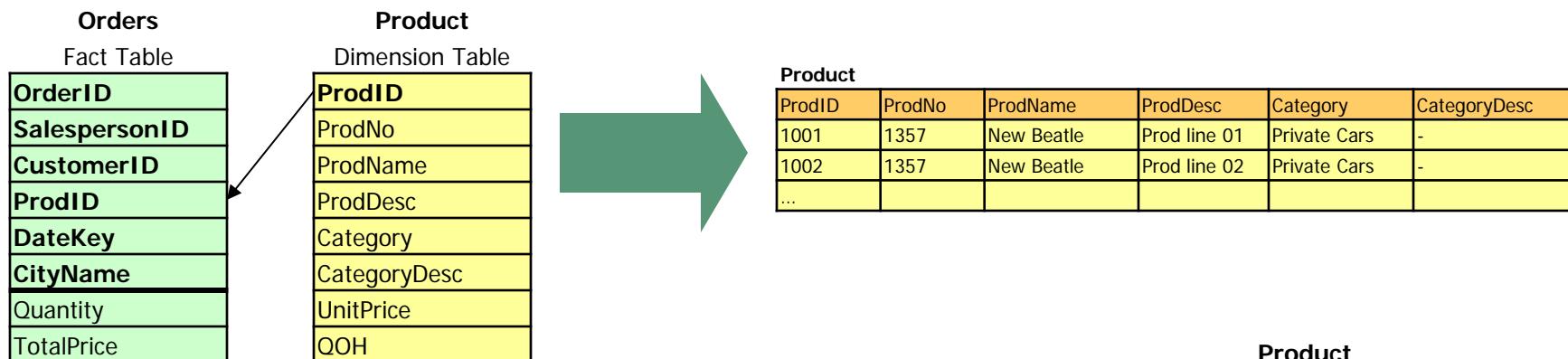
Slowly Changing Dimensions

- **Example:** The description of a product changes.

- Modeling alternatives:

Type One: Substitution of the old description by the new one.

Type Two: Product together with its new description gets a new ID.



Type Three:

Associated product entry has an additional attribute that holds the old description.

ProdID
ProdName
ProdDesc
Category
CategoryDesc
UnitPrice
QOH
ProdDescOld

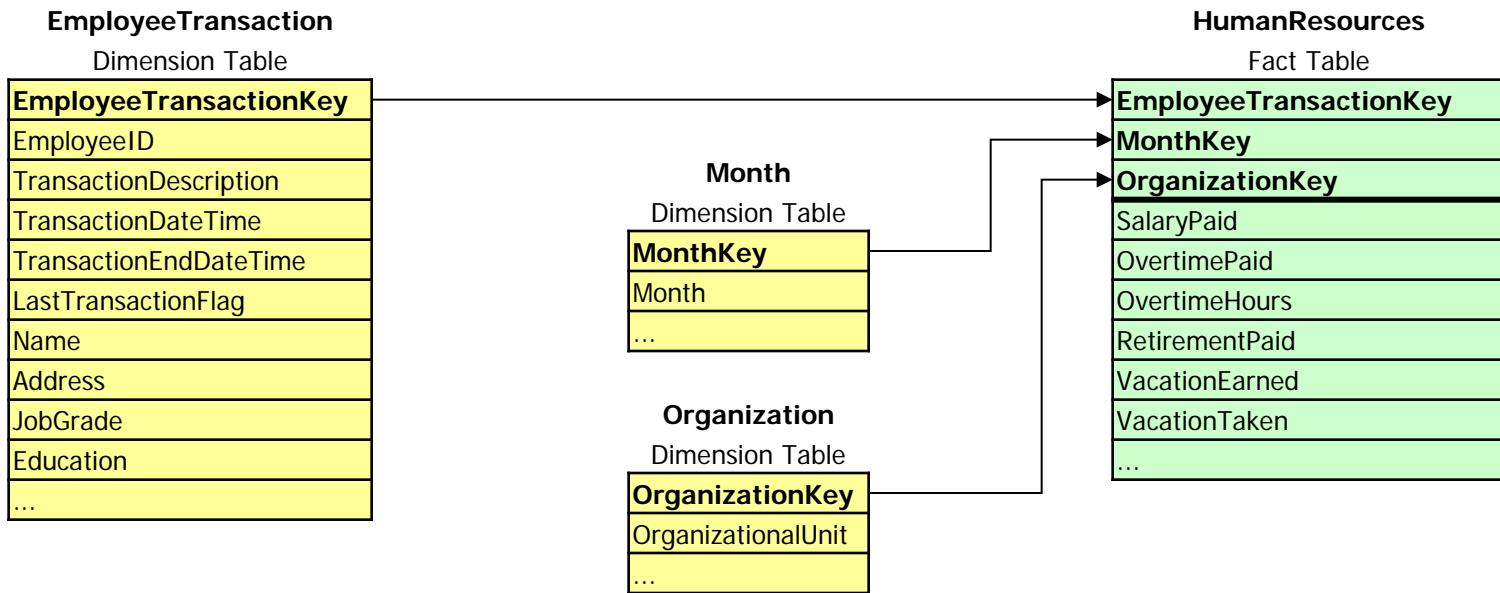
Slowly Changing Dimensions

	Type One	Type Two	Type Three
Keeps history			
Schema changed			
Volume of data changes			
Effect on queries			
Usage			

Time Stamping in Large Dimensions

- **Example:** A human resource department has to store many attributes on many employees and precisely track all the changes on these attributes.
- Reports that should be provided:
 - R1: Summary status of the entire employee base on a regular (monthly) basis.
 - R2: Profile the employee population at any precise instant of time.
 - R3: Provide every action taken on a given employee, with the correct transaction sequence and the correct timing of the transaction.
- Slowly changing dimensions, type 2:
 - R1, R2: Does not provide the valid information for an employee for a certain point in time.
 - R3: Does not track the transaction time for actions taken on employees.

Human Resource Environment



- Employee transaction table contains a complete snapshot of the employee record for each transaction.
- Human resource table contains the regular facts stored for each employee each month.
- Which tables are needed for reports R1, R2 and R3?

Large Dimensions with Frequent Changes

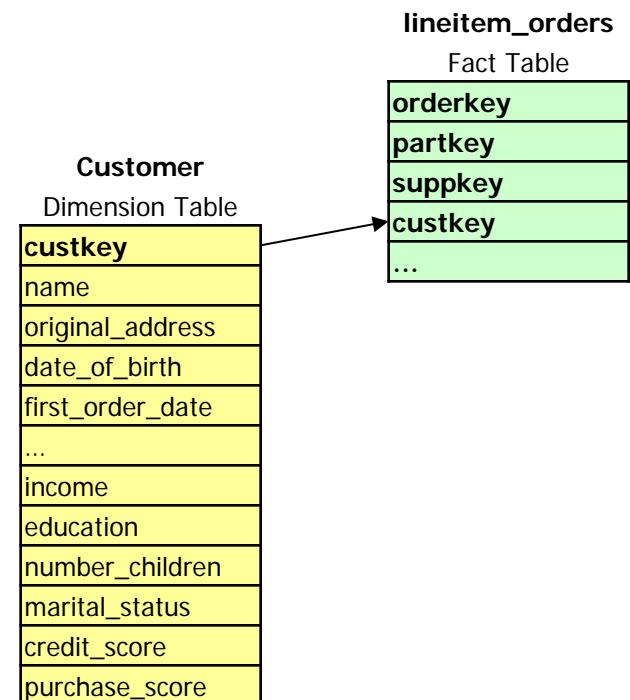
- **Example:**

- For an insurance company the customer dimension might hold millions of tuples; large amounts of demographic data are kept for each customer.
- Customer data change constantly and these changes have to be propagated into the warehouse.

Customer	
Dimension Table	
	custkey
	name
	original_address
	date_of_birth
	first_order_date
	...
	income
	education
	number_children
	marital_status
	credit_score
	purchase_score

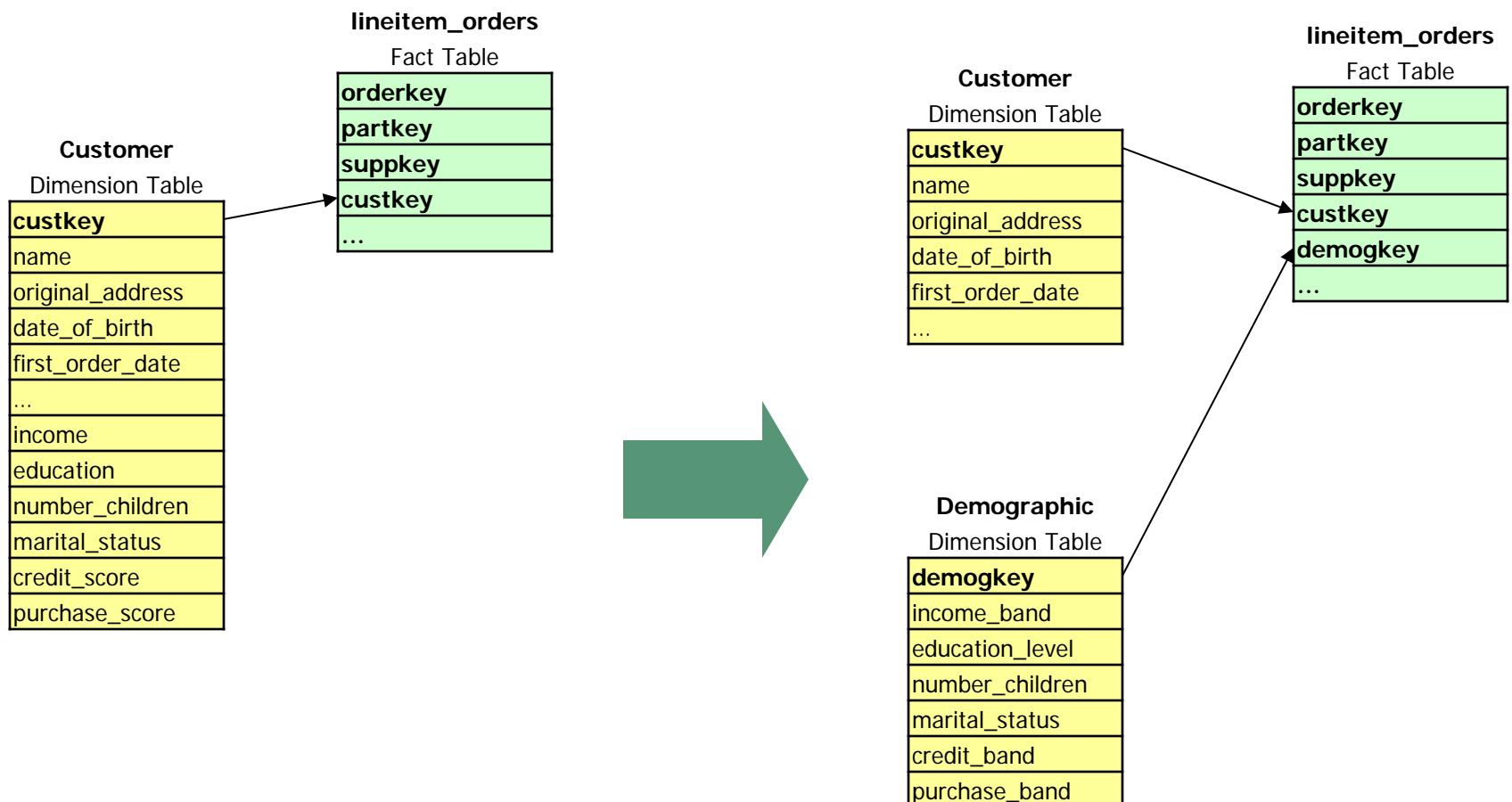
Large Dimensions with Frequent Changes

- Design goals for this kind of dimensions:
 - Support rapid browsing, e.g. of low cardinality attributes.
 - Support efficient browsing of cross-constrained values in the dimension table.
 - Do not penalize the fact table query for using a large dimension.
 - Find and suppress duplicate entries in the dimension.
 - Do not create additional records to handle the changing dimension problem.



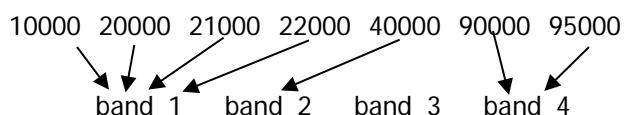
Large Dimensions with Frequent Changes

- Modeling approach:
Break off dimension into several dimension tables.

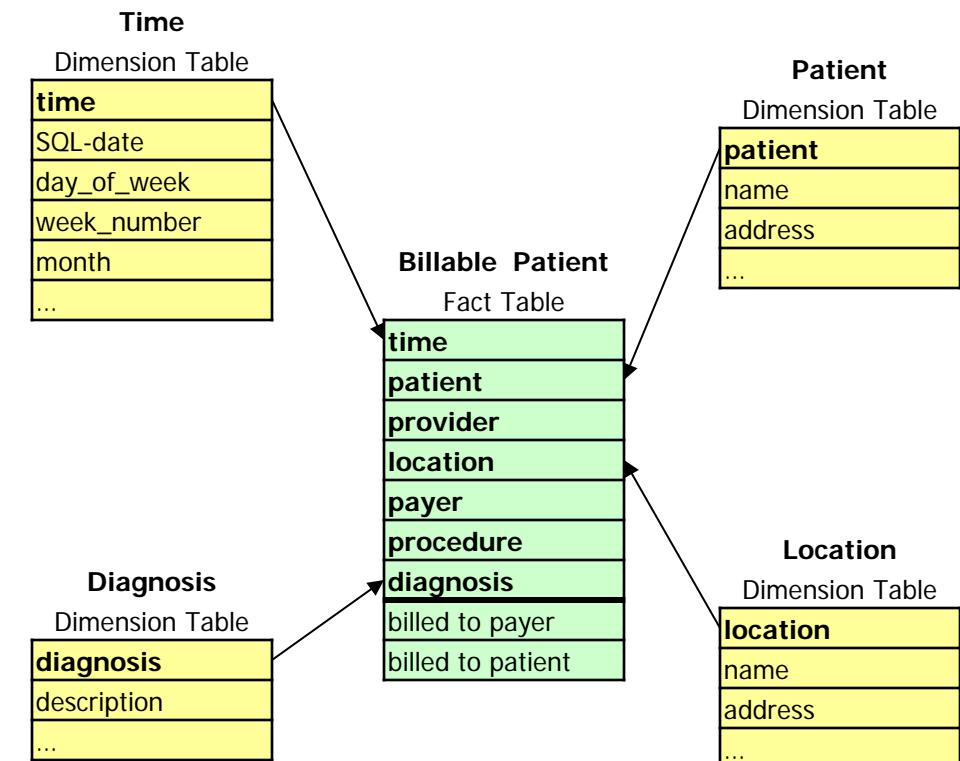


Large Dimensions with Frequent Changes

- One (or several) additional dimension table (Demographics) includes all changing attributes.
- Attributes in this new dimension are forced to have a relatively small number of discrete values:
 - e.g., income is mapped to income bands
- The additional dimension table
 - is populated with all possible discrete attribute combinations, and
 - comes with its own surrogate key.
- The surrogate key of the additional dimension is used in the fact table.
- Changes in the demographics dimension are reflected by entries in the fact table.



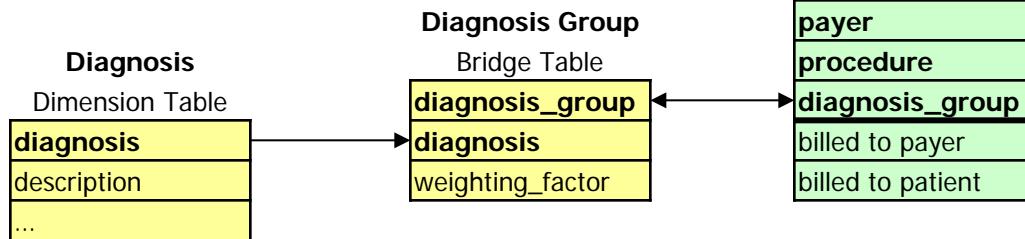
Many-to-Many Dimensions



- Dimensions may have zero, one or many values for a given fact record.
- The number of dimension values is not known in advance.
- **Example:**
A fact table record for each line item of a hospital bill or each line item of a treatment performed in a physician's office.
The design has to cover several diagnoses per patient.

Bridge Tables

- Bridge table between the fact table and the dimension table.
- Use a special diagnosis group key in the fact table.
- The diagnosis group table contains a set of records for each patient. Each record points to a specific diagnosis and provides a weighting factor.
- Over time a patient may have different diagnosis groups.
- Questions that can be answered:
 - Correctly weighted summary of all charges grouped by diagnosis.
 - Impact report that totals the impact each diagnosis has in terms of total amounts associated with that diagnosis.



Time Dimension

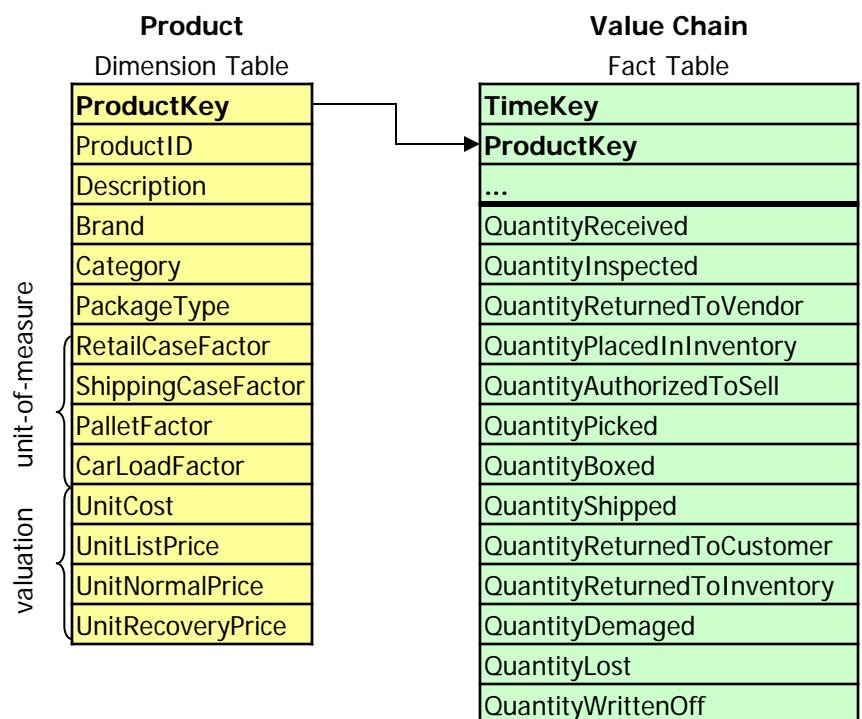
- Some transactions need a fine-scale tracking to the minute or even the second.
- Solution 1:
 - Time dimension with one record for every minute or second.
- Solution 2:
 - Time dimension on a daily basis.
 - Transaction time is treated as a fact.
 - Timestamp may be provided for various time zones.

Sales
Fact Table
DateTimeKey
ProductKey
StoreKey
CustomerKey
ClerkKey
RegisterKey
PromotionKey
DollarsSold
UnitsSold
...

Sales
Fact Table
DateKey
ProductKey
StoreKey
CustomerKey
ClerkKey
RegisterKey
PromotionKey
TimeOfDay
GMTTimeOfDay
DollarsSold
UnitsSold
...

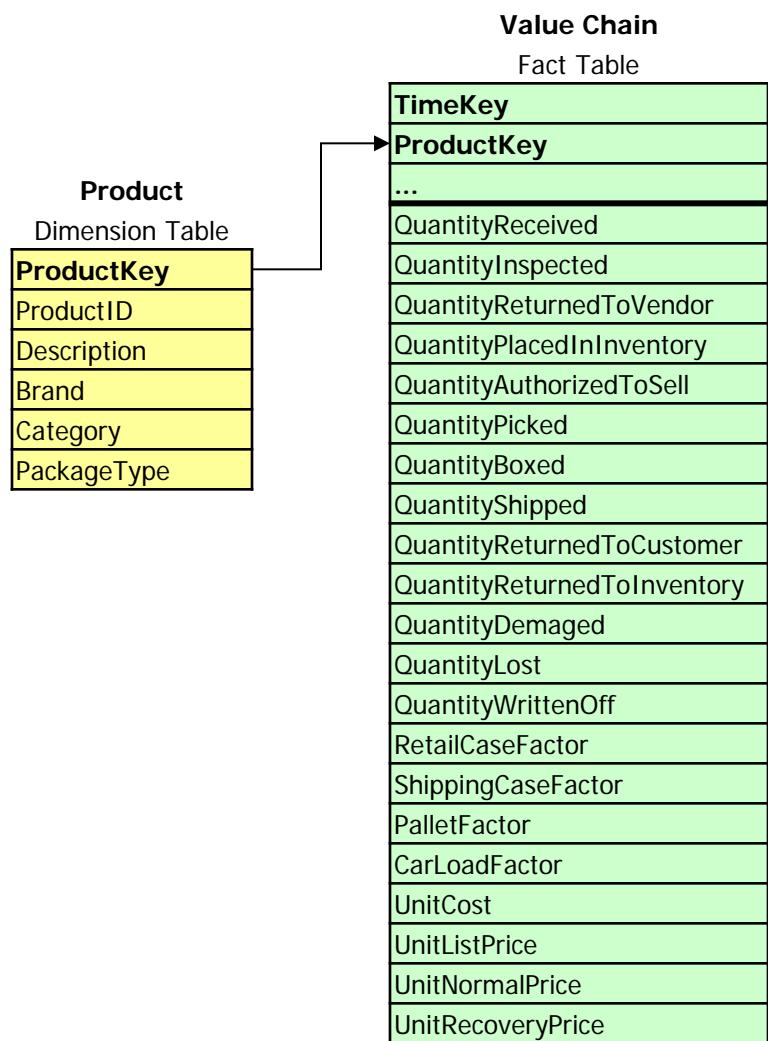
Various Units of Measure

- Several business processes monitor the flow of products or the inventory. For some of these processes numbers should be expressed in different units of measure.
- Example:
In the value chain several quantity facts have five different unit-of-measure interpretations and four valuation schemas.
- Solution 1:
 - The unit-of-measure interpretations as well as the valuation schemas are presented in the dimension table.



Various Units of Measure

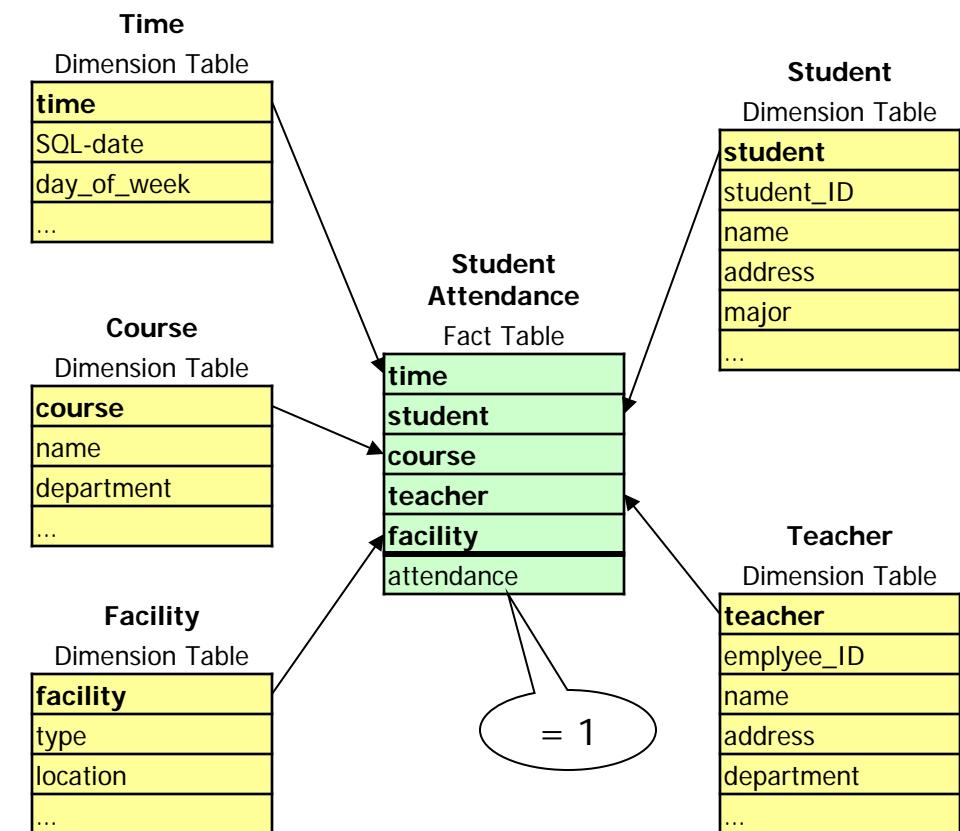
- Solution 2:
 - The unit-of-measure interpretations as well as the valuation schemas are provided in the fact table.
- Advantages of this solution:
 - Eliminates the possibility of choosing the wrong factors.
 - Changes in factors do not lead to changes in the dimension table.



Modeling Events and Coverage

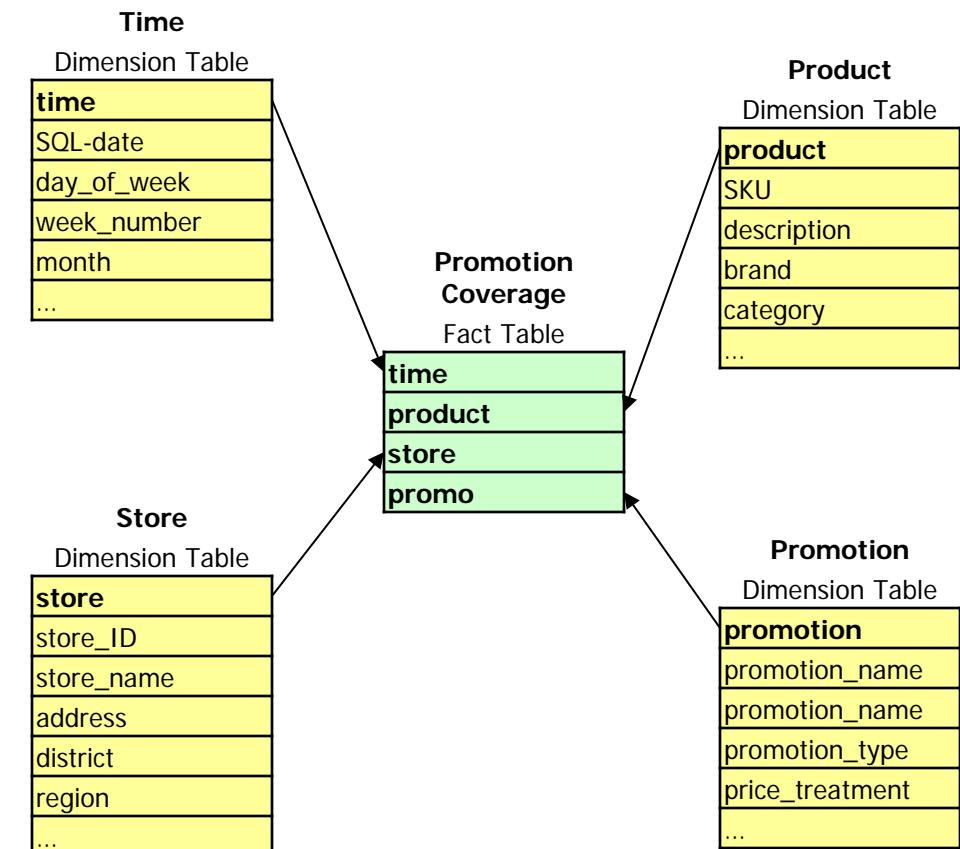
- How to model events?
- **Example:**
 - A fact table should express the fact that students participate in courses and exercises (student's attendance event).
- What are the dimensions?
What are the facts?
- Dimensions: Time, Student, Course, Teacher, Facility
- No obvious fact:
Combination of all relevant keys express the fact.
- How to model coverage?
- **Example:**
 - A fact table should record the sales of products in stores on particular days under each promotion condition.
 - Sales fact table may be sparse.
How to handle products that did not sell?
 - Dimensions: Time, Store, Product, Promotion
 - No obvious fact:
Combination of all relevant keys express the fact.

Factless Fact Tables



- Dummy fact 'attendance' is optional but makes SQL more readable.
- Many questions can be answered:
 - Which classes were the most heavily attended?
 - Which classes were the most consistently attended?
 - Which teachers taught the most students?
 - Which teachers taught classes in facilities belonging to other departments?
 - Which facilities where the most lightly used?

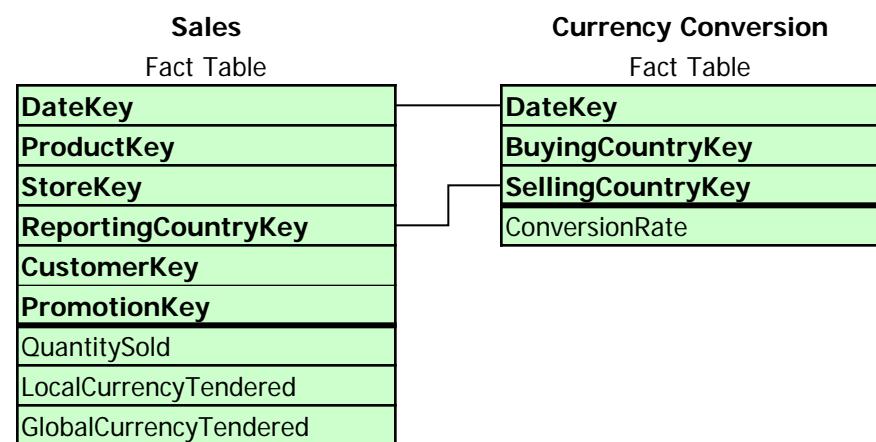
Factless Fact Tables



- Answering the question of which products were on promotion but did not sell requires a two-step application.
 - first, consult coverage table
 - second, consult sales table
- Use sales fact table instead and fill in records representing zero sales for all possible products?
 - The coverage factless fact table can be made much smaller.

Multinational Currency Tracking

- Transactions have to be expressed in a multitude of currencies.
- Solution:
 - Value is reported in the local currency as well as in the global currency for a multinational enterprise (e.g. US dollars).
 - Conversion table provides conversion rates on a daily basis in both directions.



Overview

- Data Warehouse Design Process
- Conceptual Design
- Logical Design
- Details of Logical Design
- Physical Design

Physical Design

Physical implementation of the logical schemata with respect to the individual properties of the target database system.

- What to consider in physical design:
 - indexing
 - partitioning
 - denormalization
 - pre-aggregation
 - ...
- See also 'Database Support for Data Warehousing'

Summary

- Main steps of the Data Warehouse Design Process:
 - req. analysis, conceptual, logical and physical design
- Different conceptual models are available. Common elements:
 - attributes, dimensions, hierarchies, facts, measures
- Conceptual models usually provide a graphical notation.
- Logical design is based on the multidimensional model or the relational model.
- Conceptual schema may be mapped to different logical schema types (relational model):
 - star schema, snowflake schema, vendor-specific schemes
- Physical design deals with optimization steps for the target database system.

Papers

- [GMR98] M. Golfarelli, D. Maio, S. Rizzi: The Dimensional Fact Model: A Conceptual Model for Data Warehouses. International Journal of Cooperative Information Systems, Vol. 7, No. 2&3, 1998.
- [HLV00] B. Hüsemann, J. Lechtenbörger, G. Vossen: Conceptual Data Warehouse Design. Proc. of the Second International Workshop on Design and Management of Data Warehouses, Stockholm, 2000.
- [LTS06] S. Luján-Mora, J. Trujillo, I.-Y. Song: A UML profile for multidimensional modeling in data warehouses. Data & Knowledge Engineering 59 (2006) 725-769.
- [PJ99] T. B. Pedersen, C. S. Jensen: Multidimensional Data Modeling for Complex Data. Proc. of the 15th International Conference on Data Engineering, Sydney, Australia, 1999.
- [TBC99] N. Tryfona, F. Busborg, J. G. Christiansen: starER: A Conceptual Model for Data Warehouse Design. Proc. of the ACM Second International Workshop on Data Warehousing and OLAP, Kansas City, Missouri, USA, 1999.

Appendix: starER

- Combines the star structure with the semantically rich constructs of the ER model.
- Adds special types of relationships to support hierarchies.
- starER vs. DFM:
 - starER allows many-to-many relationships between dimensions and facts.
 - starER allows objects participating in the data warehouse, but not in the form of a dimension.
 - Specialized relationships on dimensions are permitted in starER (specialization/generalization).

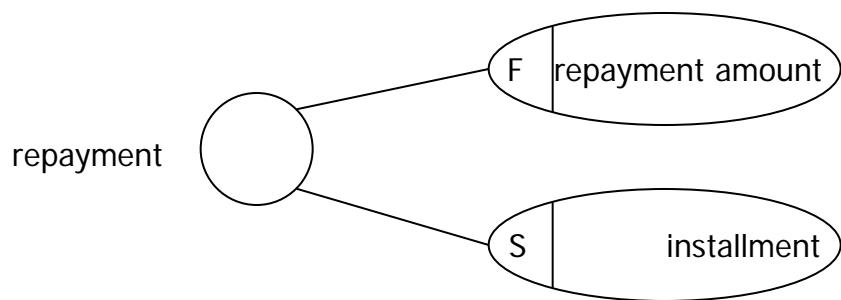
Appendix: starER: Fact Sets

- **Fact set:**
 - Represents a set of real-world facts sharing the same characteristics or properties.
 - Semantically, a fact set points to the process of generating data over time, i.e., data is generated in terms of facts, each time an event related to the fact takes place.
- Graphical representation:



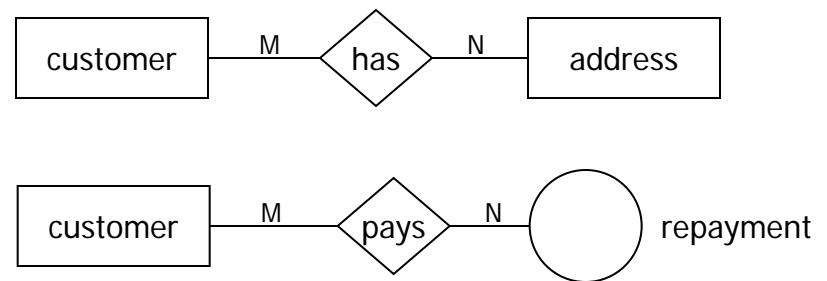
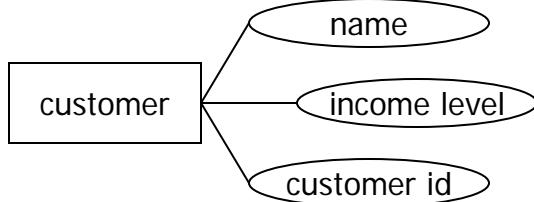
Appendix: starER: Fact Properties

- **Fact properties:**
 - Properties that characterize a fact.
 - Usually numerical data that can be summarized.
 - There are three different types of properties:
 - stock (S): records the state of something at a specific point in time.
 - flow (F): records the commutative effect over a period of time.
 - value-per-unit (V): like 'stock', but the unit of the property is different
- Graphical representation:



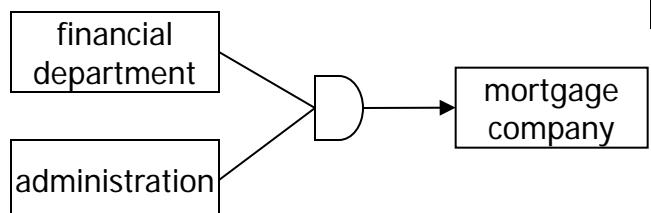
Appendix: starER: Entity sets and relationship sets

- **Entity sets:**
 - Represent a set of real-world objects with similar properties.
 - It has the same meaning as in traditional application modeling.
- **Relationship sets:**
 - Represents a set of associations among entity sets or among entity sets and fact sets.
 - Its cardinality can be many-to-many, many-to-one or one-to-many.
- **Attributes:**
 - Static properties of entity sets, relationship sets, and fact sets.
- Graphical representation:

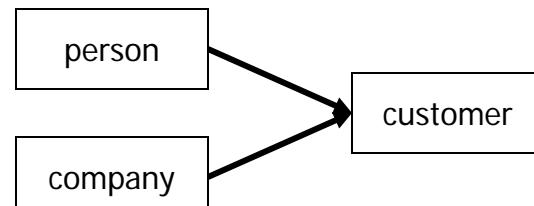


Appendix: starER: Relationships

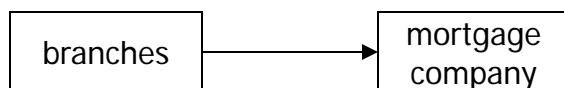
- Specialization / generalization



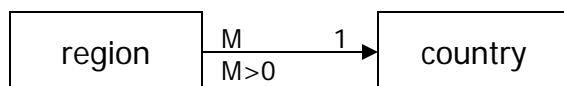
- Aggregation



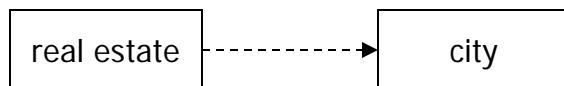
- Complete membership



- Non-complete membership

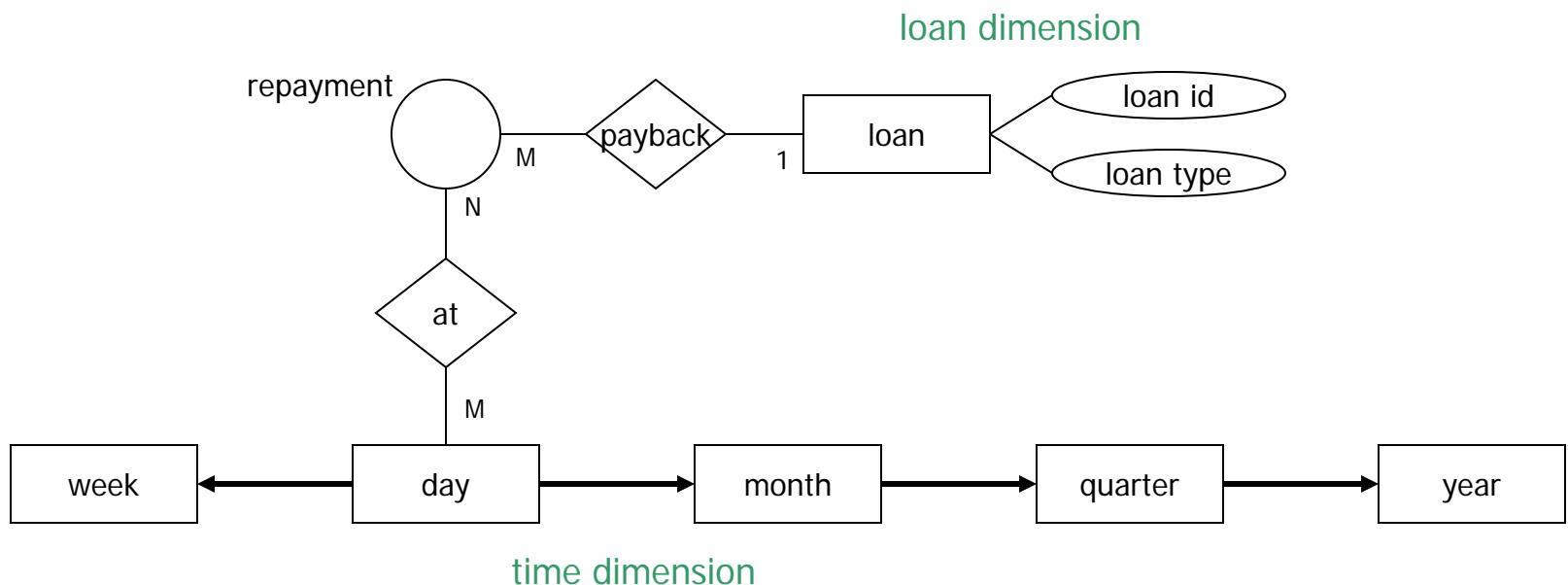


- Strict membership

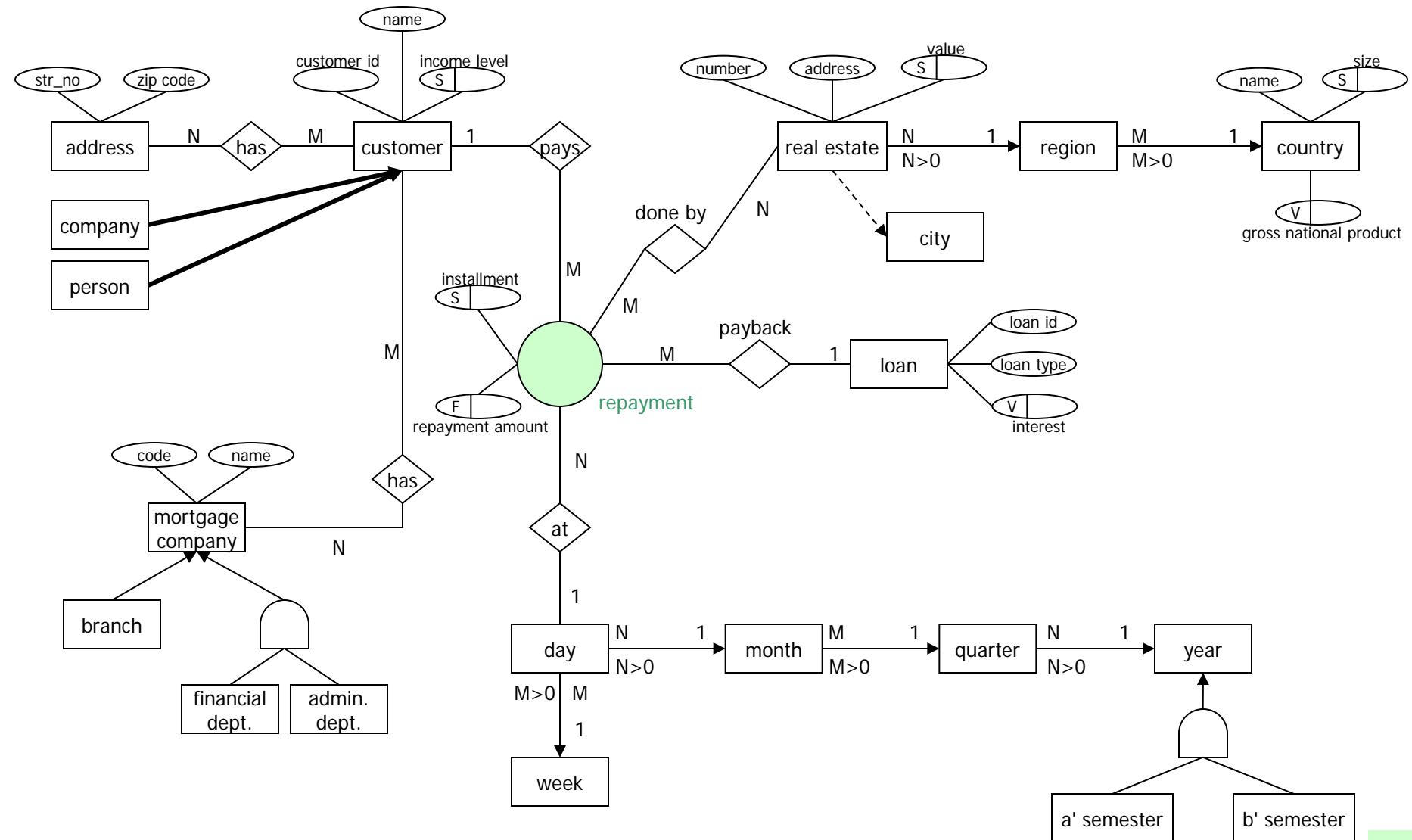


Appendix: starER: Dimensions and Hierarchies

- **Dimension:**
 - Entity sets associated to a fact are the dimensions of that fact.
- **Hierarchies:**
 - Dimensions consist of hierarchies and other relationships among other entity sets.



Appendix: Mortgage Company DW



Data-Warehouse-, Data-Mining- und OLAP-Technologien

Chapter 4: Extraction, Transformation, Load

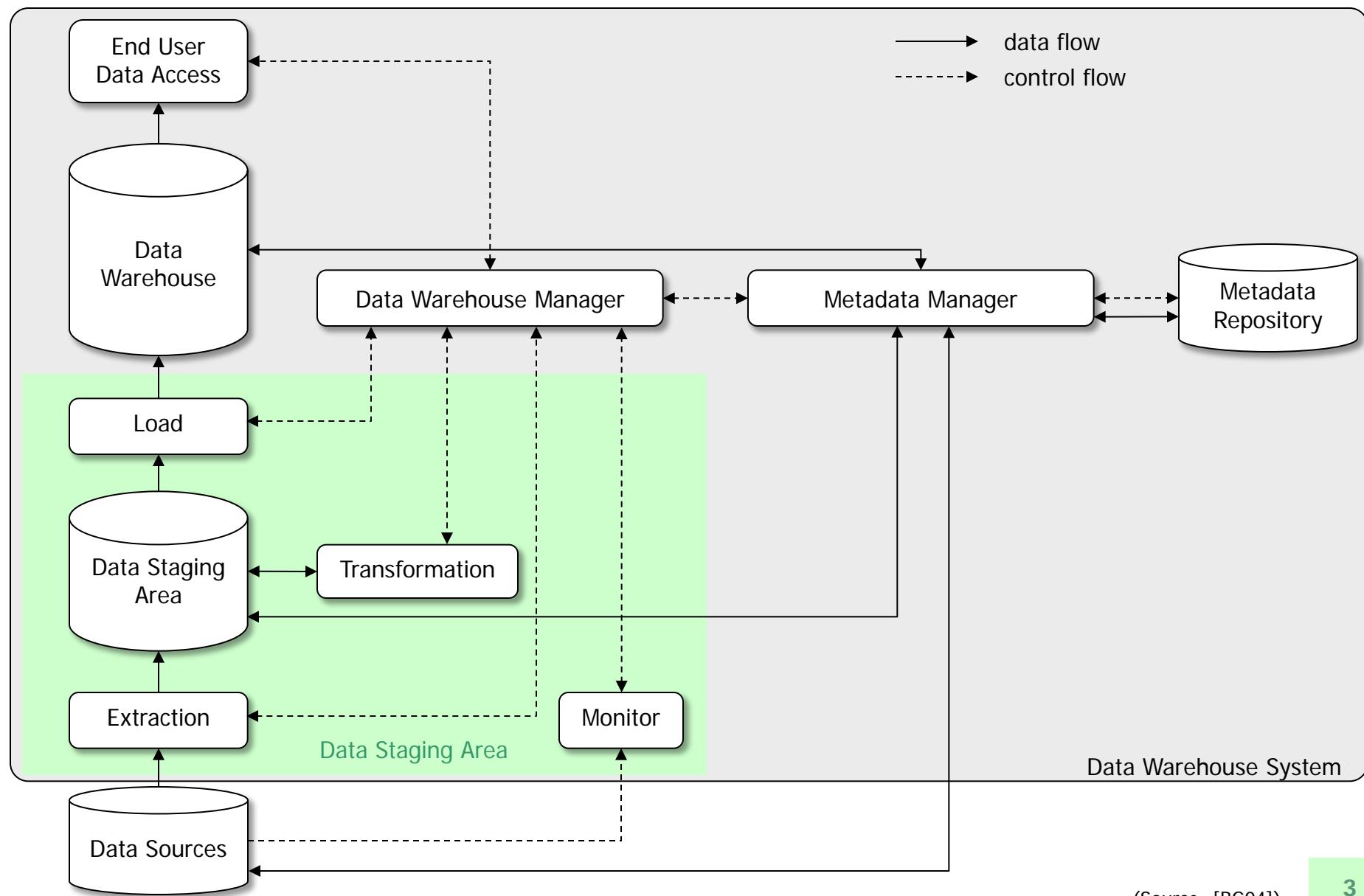
Bernhard Mitschang
Universität Stuttgart

Winter Term 2015/2016

Overview

- Monitoring
- Extraction
 - Export, Import, Filter, Load
 - Direct Integration
- Load
 - Bulk Load
 - Replication
 - Materialized Views
- Transformation
 - Schema Integration
 - Data Integration
 - Data Cleansing
- Tools

Architecture

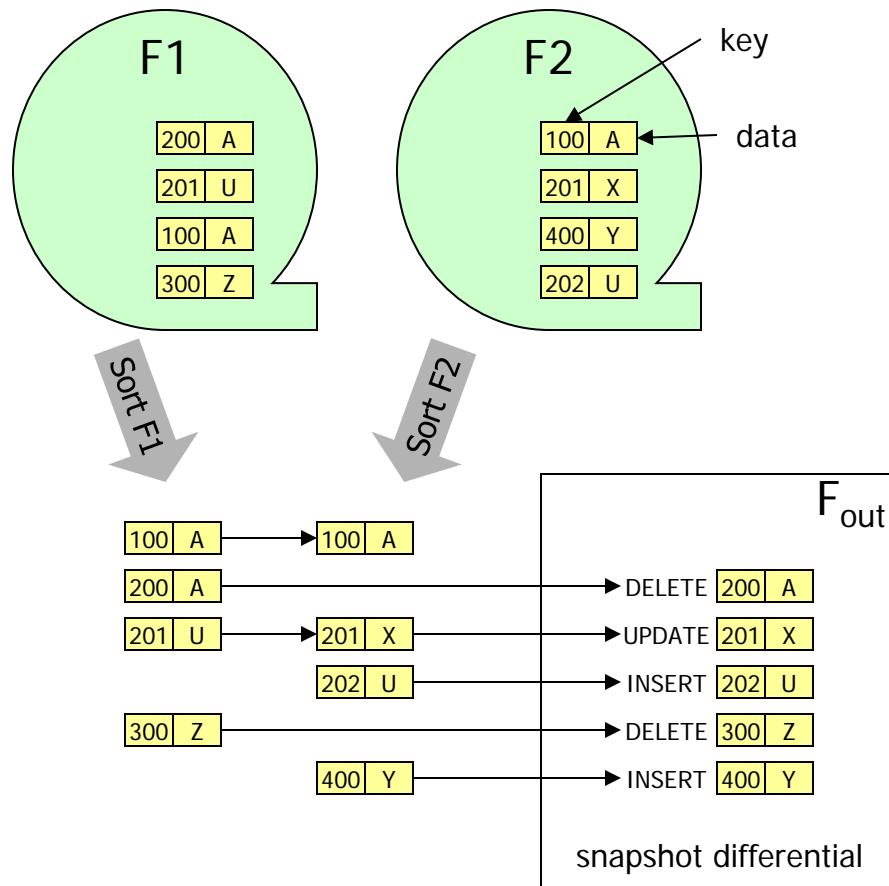


Monitoring

- Goal: Discover changes in data source incrementally
- Approaches:

	Based on ...	Changes identified by ...
Trigger	triggers defined in source DBMS	trigger writes a copy of changed data to files
Replica	replication support of source DBMS	replication provides changed rows in a separate table
Timestamp	timestamp assigned to each row	use timestamp to identify changes (supported by temporal DBMS)
Log	log of source DBMS	read log
Snapshot	periodic snapshot of data source	compare snapshots

Snapshot Differentials



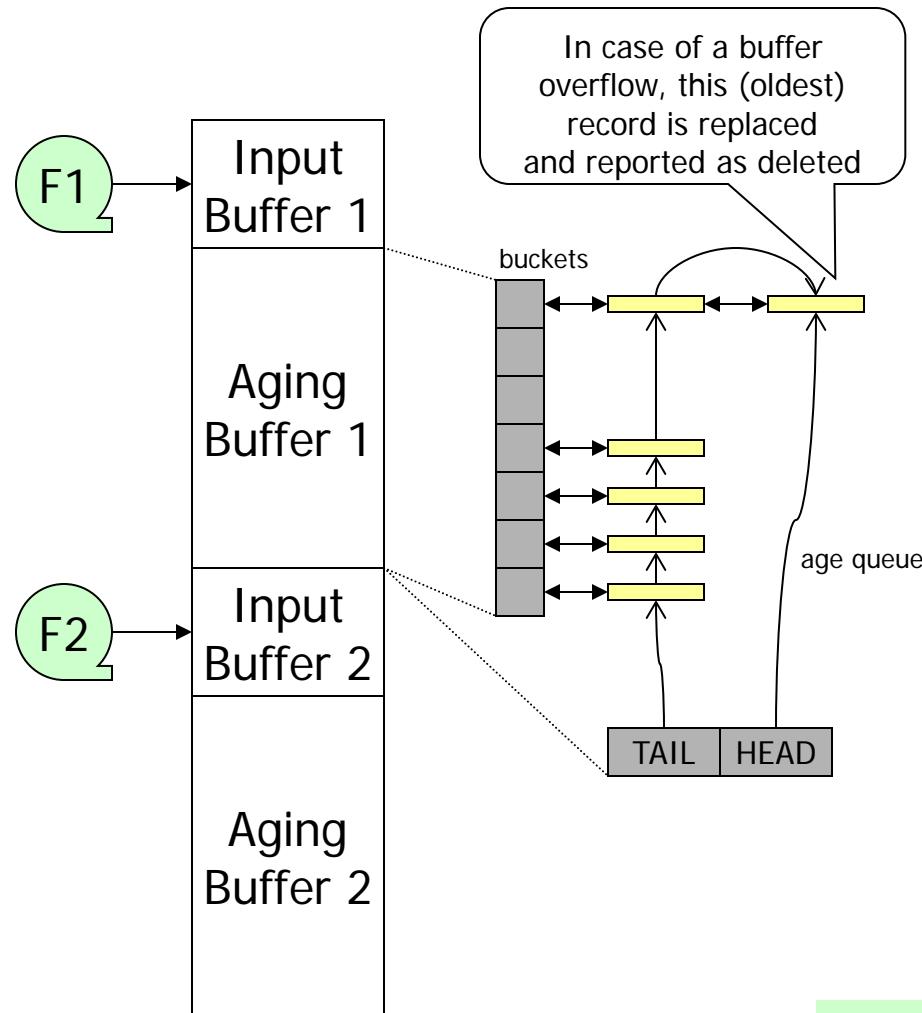
- Two snapshot files:
F1 was taken before F2
- Records contain key fields and data fields
- **Goal:** Provide UPDATES/ INSERTS/ DELETES in a snapshot differential .
- Sort Merge Outerjoin:
 - Sort F1 and F2 on their keys
 - Read F1' and F2' and compare records
 - Snapshot files may be compressed
 - Snapshots are read multiple times
- Window Algorithm:
 - Maintain a moving window of records in memory for each snapshot (aging buffer)
 - Assumes that matching records are "physically" nearby
 - Read snapshots only once

Window Algorithm

INPUT: F_1, F_2, n

OUTPUT: F_{out} /* the snapshot differential */

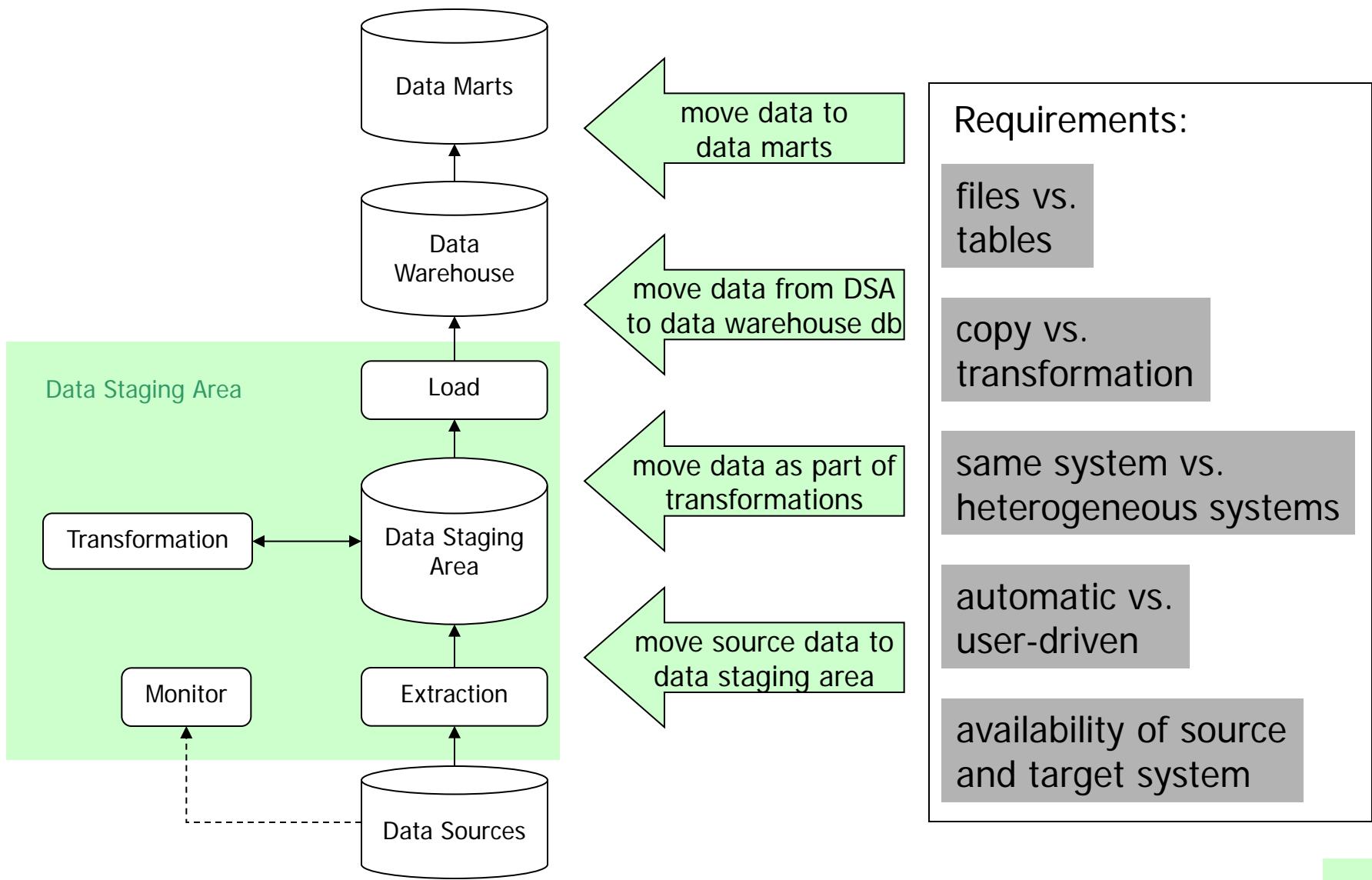
- (1) Input Buffer₁ \leftarrow Read n blocks from F_1
- (2) Input Buffer₂ \leftarrow Read n blocks from F_2
- (3) while ((Input Buffer₁ \neq EMPTY) and
 (Input Buffer₂ \neq EMPTY))
- (4) Match Input Buffer₁ against Input Buffer₂
- (5) Match Input Buffer₁ against Aging Buffer₂
- (6) Match Input Buffer₂ against Aging Buffer₁
- (7) Put contents of Input Buffer₁
 to Aging Buffer₁
- (8) Put contents of Input Buffer₂
 to Aging Buffer₂
- (9) Input Buffer₁ \leftarrow Read n blocks from F_1
- (10) Input Buffer₂ \leftarrow Read n blocks from F_2
- (11) Report records in Aging Buffer₁ as deletes
- (12) Report records in Aging Buffer₂ as inserts



Overview

- Monitoring
- Extraction
 - Export, Import, Filter, Load
 - Direct Integration
- Load
 - Bulk Load
 - Replication
 - Materialized Views
- Transformation
 - Schema Integration
 - Data Integration
 - Data Cleansing
- Tools

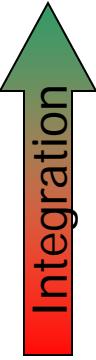
ETL Processing



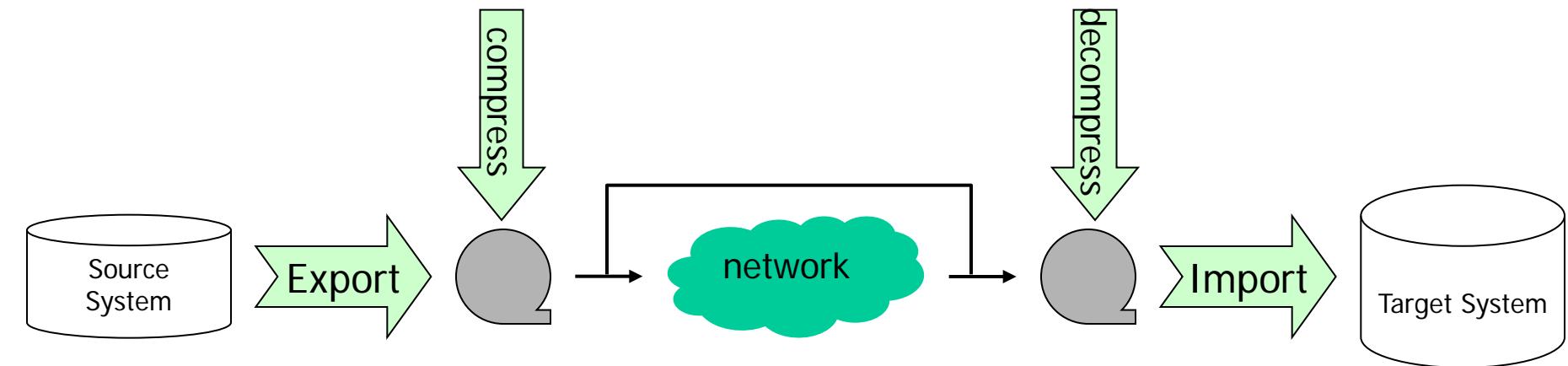
Extraction

heterogeneous source systems

performance

- support of monitoring and extraction:
 - replica
 - active db / trigger
 - snapshot
 - export / db dump
 - logging
 - no support
 - accessing data sources
 - application / application API
 - database API
 - log files
- 
- The diagram features a vertical arrow pointing upwards, divided into two colored segments: green at the top and red at the bottom. The word "Integration" is written vertically along the center of the arrow.

Export and Import



```
EXPORT TO      c:\cust_berlin_1\cust.data
OF              DEL
MODIFIED BY   COLDEL |
MESSAGES      c:\cust_berlin_1\msg1.txt
SELECT        *
FROM          customer_data
WHERE         new_customer = true
                           (DB2)
```

- Export:
 - ASCII files
 - proprietary format
- Import:
 - import command
 - bulk load

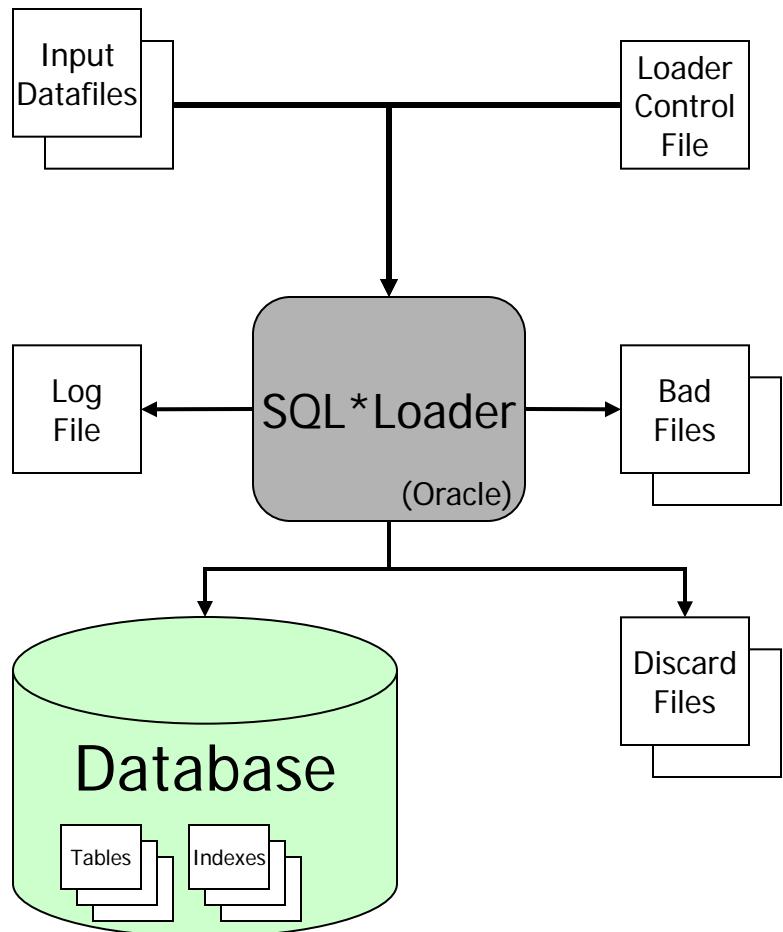
Import vs. Load

```
IMPORT FROM c:\cust_berlin_1\cust.data  
OF DEL  
MODIFIED BY COLDEL |  
COMMITCOUNT 1000  
MESSAGES c:\cust_berlin_1\msg2.txt  
INSERT INTO cust_berlin_1  
          (DB2)
```

```
LOAD FROM c:\cust_berlin_1\cust.data  
OF DEL  
MODIFIED BY COLDEL |  
SAVECOUNT 1000  
MESSAGES c:\cust_berlin_1\msg3.txt  
REPLACE INTO cust_berlin_1  
STATISTICS YES  
           (DB2)
```

	IMPORT	LOAD
COMMIT	explicit	automatic
Logging	complete, mandatory	optional
Integrity	check all constraints	check local constraints only
Trigger	all	none
Locking	read access possible	table lock

Filter and Load



- Bulk load tools provide some filter and transformation functionality:
 - Load data from multiple datafiles during the same load session.
 - Load data into multiple tables during the same load session.
 - Specify the character set of the data.
 - Selectively load data, i.e. load records based on the records' values.
 - Manipulate the data before loading it, using SQL functions.
 - Generate unique sequential key values in specified columns.

Direct Integration

external tables

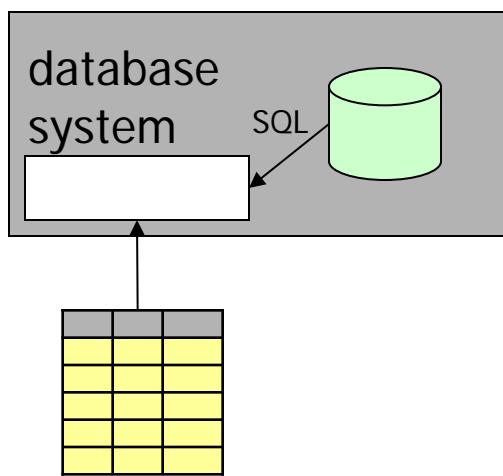
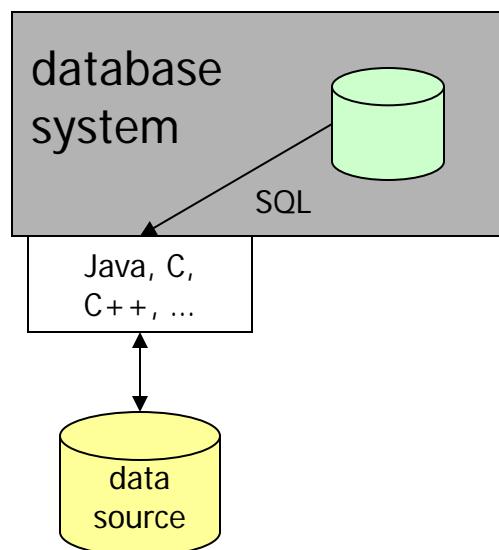
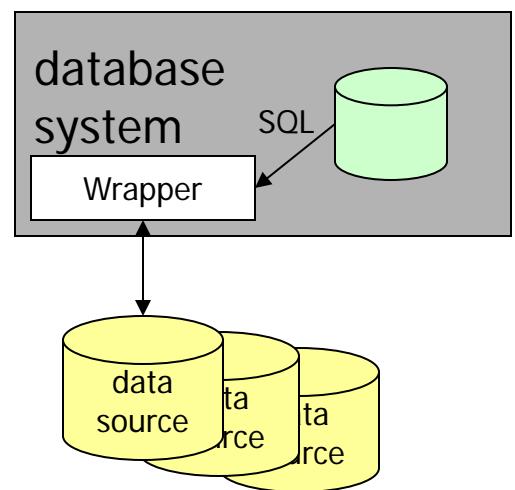


table functions



federated database



- register external data as tables
- allows to read external data
- no query optimization

- user-defined functions provides a tables as result
- function reads data from external sources

- register external data as tables
- define wrapper for access to external data
- exploit capabilities of external source for query optimization

Overview

- Monitoring
- Extraction
 - Export, Import, Filter, Load
 - Direct Integration
- Load
 - Bulk Load
 - Replication
 - Materialized Views
- Transformation
 - Schema Integration
 - Data Integration
 - Data Cleansing
- Tools

Load

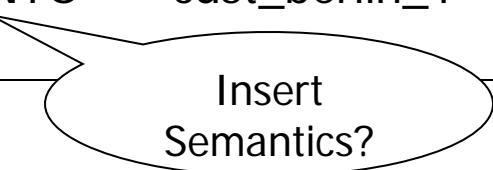
Transfer data from the data staging area into the data warehouse and data marts.

- Bulk load is used to move huge amounts of data.
- Data has to be added to existing tables:
 - add rows
 - replace rows or values
- Flexible insert mechanism is needed to:
 - add and update rows based on a single data source.
 - add rows for a single data source to multiple tables in the data warehouse.
- Consider complex criteria in load processing:
 - write application program
 - use procedural extensions of SQL

Update and Insert

```
IMPORT FROM c:\cust_berlin_I\cust.data  
OF DEL  
MODIFIED BY COLDEL |  
COMMITCOUNT 1000  
MESSAGES c:\cust_berlin_I\msg2.txt  
INSERT INTO cust_berlin_1
```

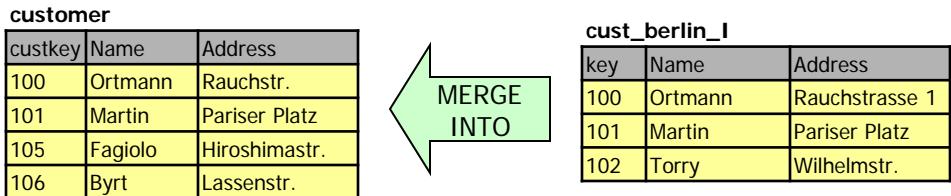
(DB2)



- **INSERT:**
 - Adds the imported data to the table without changing the existing table data.
- **INSERT_UPDATE:**
 - Adds rows of imported data to the target table, or updates existing rows (of the target table) with matching primary keys.

- **REPLACE:**
 - Deletes all existing data from the table by truncating the data object, and inserts the imported data. The table definition and the index definitions are not changed.
- **REPLACE_CREATE:**
 - If the table exists, deletes all existing data from the table by truncating the data object, and inserts the imported data without changing the table definition or the index definitions.
 - If the table does not exist, creates the table and index definitions, as well as the row contents.

MERGE INTO



```
MERGE INTO      customer AS c1
USING ( SELECT  key, name, address, ...
           FROM    cust_berlin_I
           WHERE   ...) AS c2
ON ( c1.custkey = c2.key )
WHEN MATCHED THEN
        UPDATE SET c1.address = c2.address
WHEN NOT MATCHED THEN
        INSERT (custkey, name, address, ...)
        VALUES (key, name, address ,...)
```

- 'transaction table' (*cust_berlin_I*) contains updates to existing rows in the data warehouse and/or new rows that should be inserted.
- MERGE Statement of SQL:2003 allows to
 - update rows that have a matching counterpart in the master table
 - insert rows that do not have a matching counterpart in the master table



Multiple Inserts

- Insert rows (partly) into several target tables.
- Allows to insert the same row several times.
- Allows to define conditions to select the target table.
- **INSERT FIRST** defines that the rows is inserted only once.

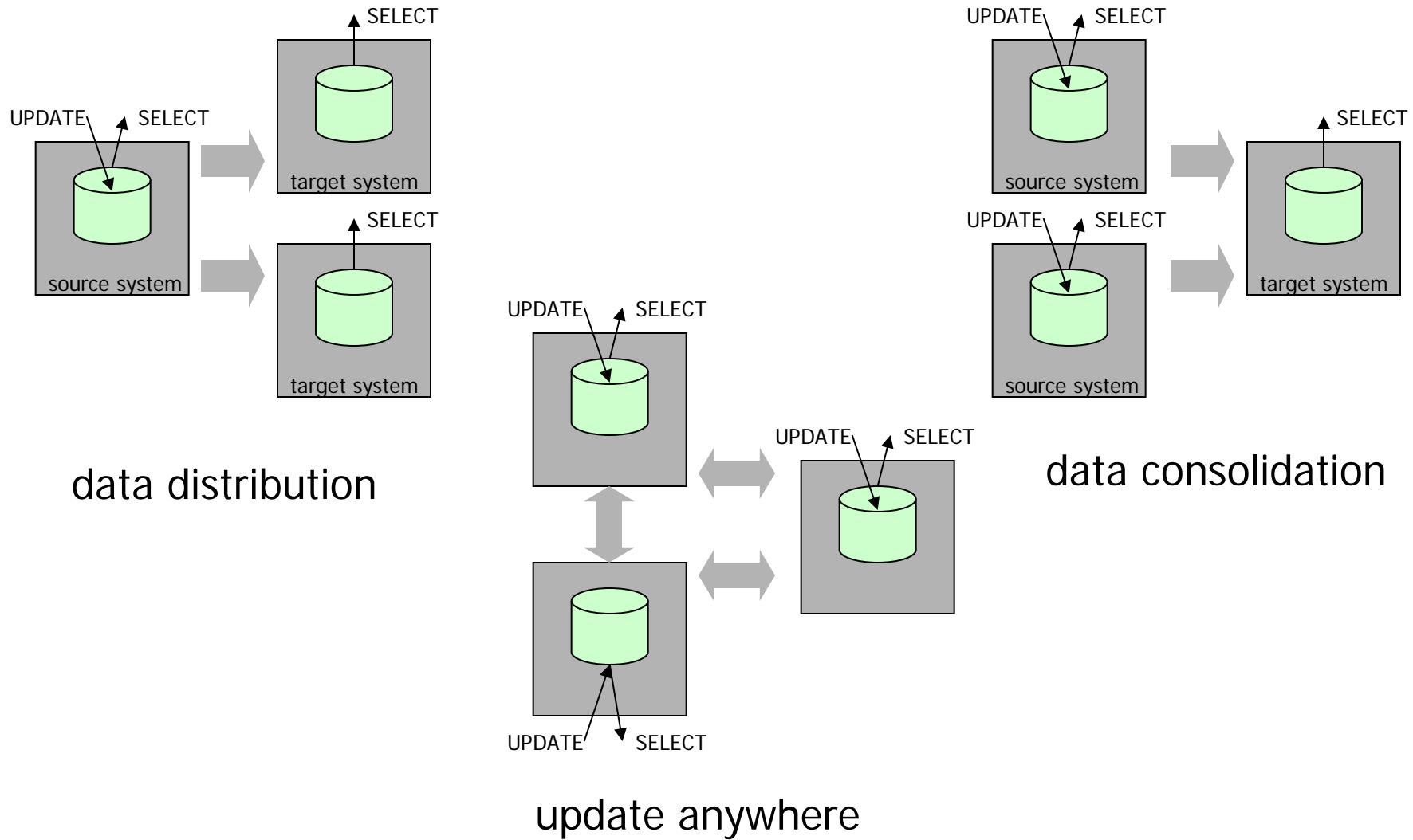
INSERT ALL

```
INTO customer
VALUES (key, name, address)
INTO location
VALUES (address, 'Berlin')
SELECT * FROM cust_berlin_I WHERE ...
(Oracle)
```

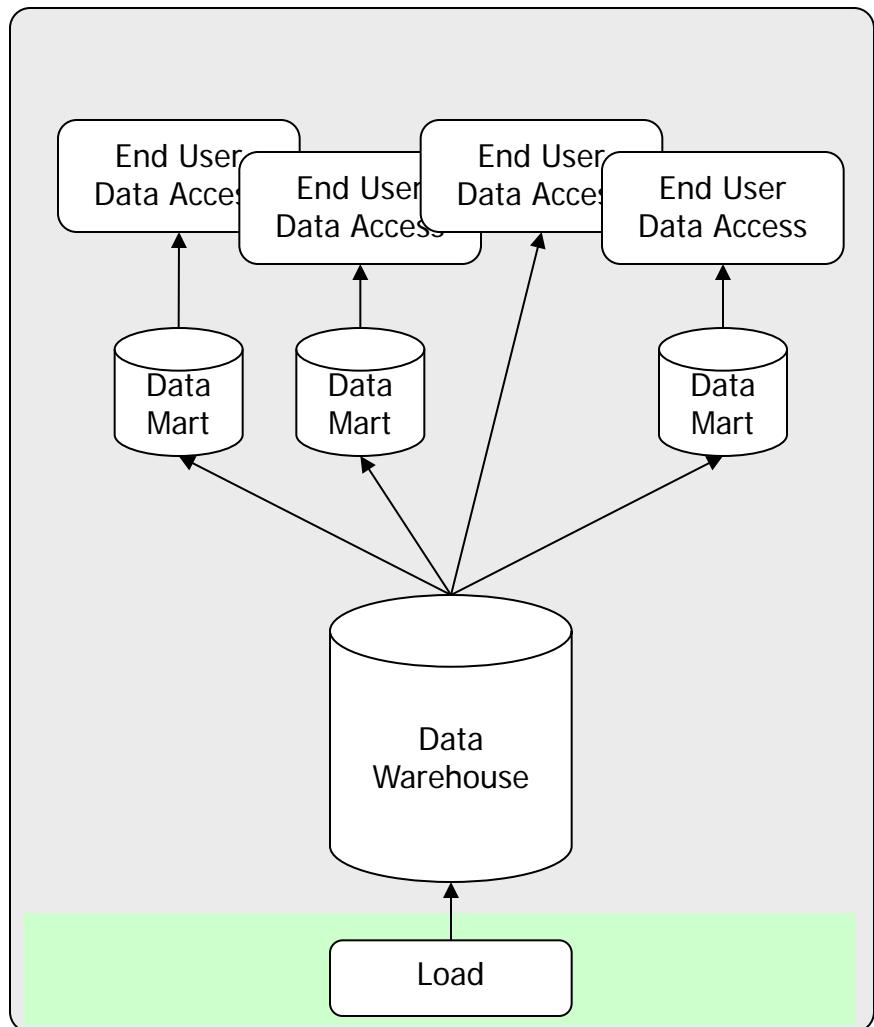
INSERT ALL /* INSERT FIRST */

```
WHEN key < 100
    INTO customer
    VALUES (key, name, address)
WHEN key < 1000
    INTO location
    VALUES (address, 'Berlin')
SELECT * FROM cust_berlin_I WHERE ...
(Oracle)
```

Replication



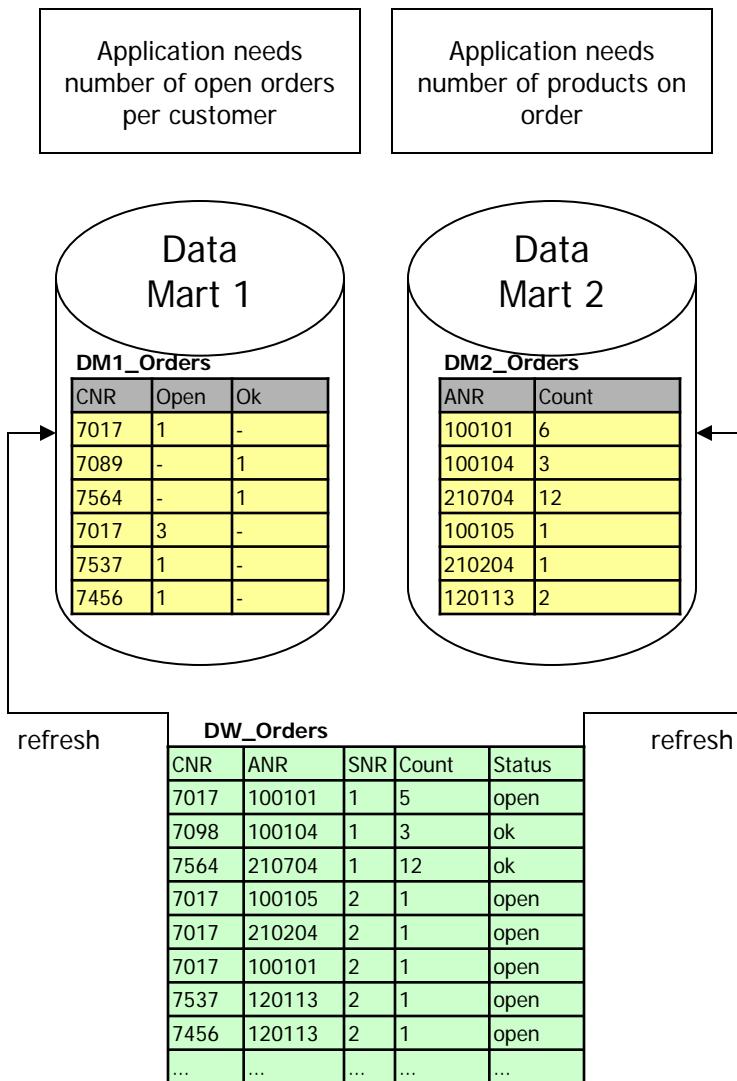
Materialized Views



dependent data marts

- Data marts provide extracts of the data warehouse for a specific application.
- Applications often need aggregated data.
- Materialized Views (MV) allow to:
 - define the content of each data mart as views on data warehouse tables
 - automatically update the content of a data mart
- Important Issues:
 - MV selection
 - MV refresh
 - MV usage

Materialized Views



```

CREATE TABLE DM2_Orders AS (
  SELECT ANR, SUM(Count)
  FROM DW_Orders
  GROUP BY ANR )
DATA INITIALLY DEFERRED
REFRESH DEFERRED;

REFRESH TABLE DM2_Orders;
  
```

(DB2)

- Materialized views are created like views
- A strategy for refreshing has to be specified:
 - DEFERRED: Use REFRESH TABLE statement
 - IMMEDIATE: As part of the update in the source table

Overview

- Monitoring
- Extraction
 - Export, Import, Filter, Load
 - Direct Integration
- Load
 - Bulk Load
 - Replication
 - Materialized Views
- Transformation
 - Schema Integration
 - Data Integration
 - Data Cleansing
- Tools

Transformation

Convert the data into something representable to the users and valuable to the business.

- Transformation of structure **and** content:
 - Semantics → identify proper semantics
 - Structure → schema integration
 - Data → data integration and data cleansing

Transformation: Semantics

- Information on the same object is covered by several data sources.
- E.g., customer information is provided by several source systems.
- Identify synonyms

car	automobile
student	pupil
baby	infant

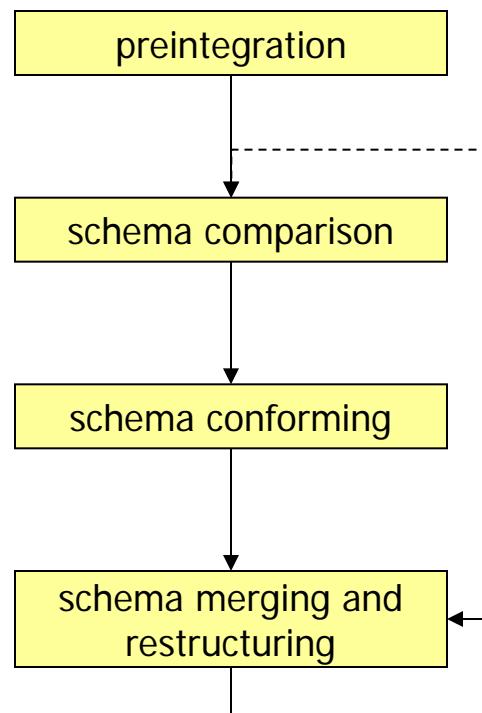
Identify homonyms

cash	cache
bare	bear
sight	site

- Identifying the proper semantics depends on the context.
- Users have to define the proper semantics for the data warehouse.
- Describe semantics in the metadata repository.

Schema Integration

- Schema integration is the activity of integrating the schemata of various sources to produce a homogeneous description of the data of interest.
- Properties of the integrated schema:
 - completeness
 - correctness
 - minimality
 - understandability
- Steps of schema integration:

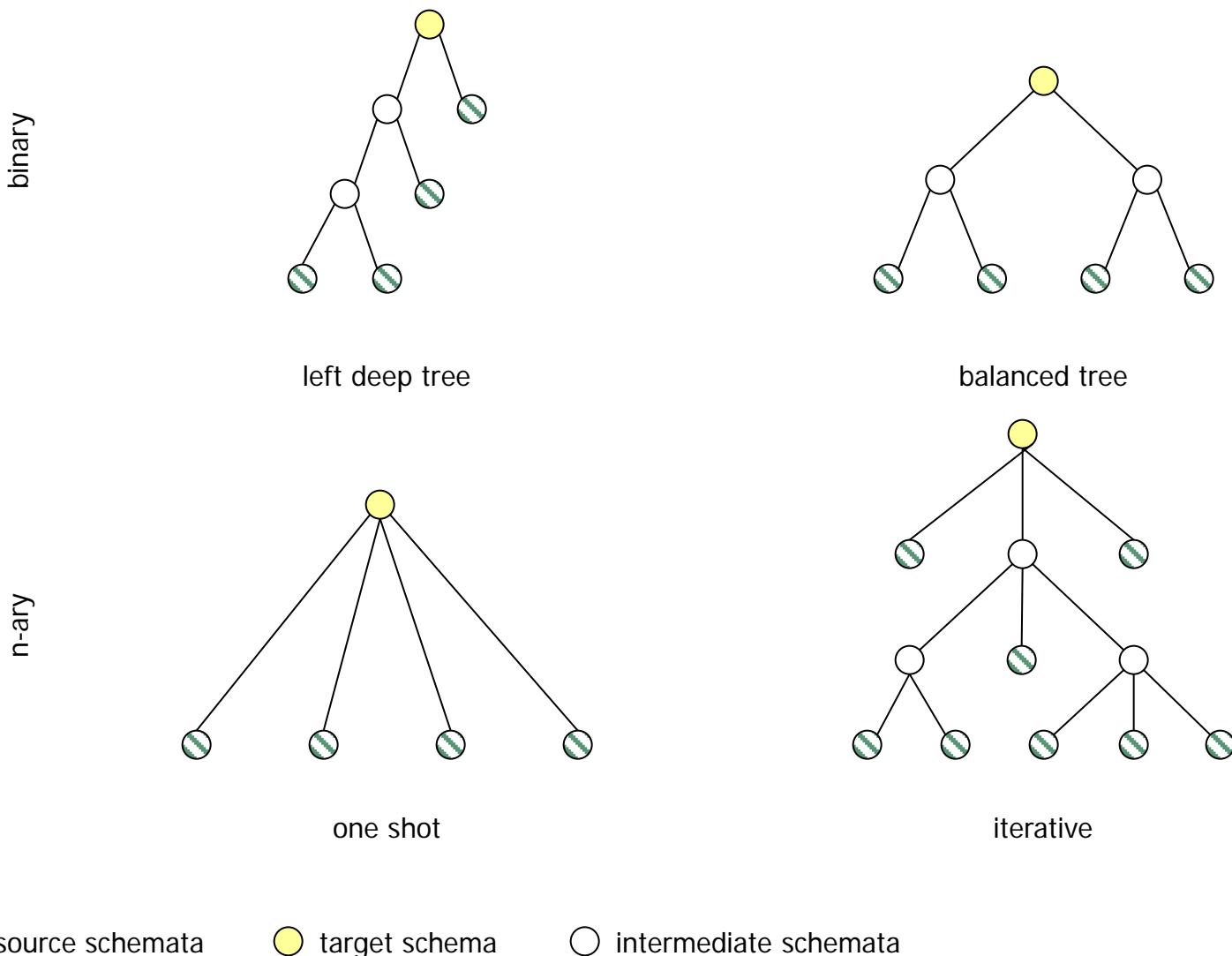


Pre-Integration

Analysis of the schemata to decide on the general integration policy.

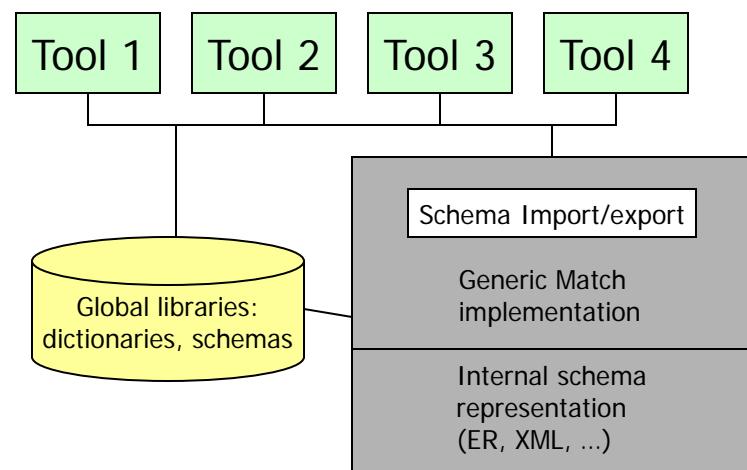
- Decide on:
 - schemata to be integrated
 - order of integration / integration process
 - preferences

Schema Integration Process



Schema Matching

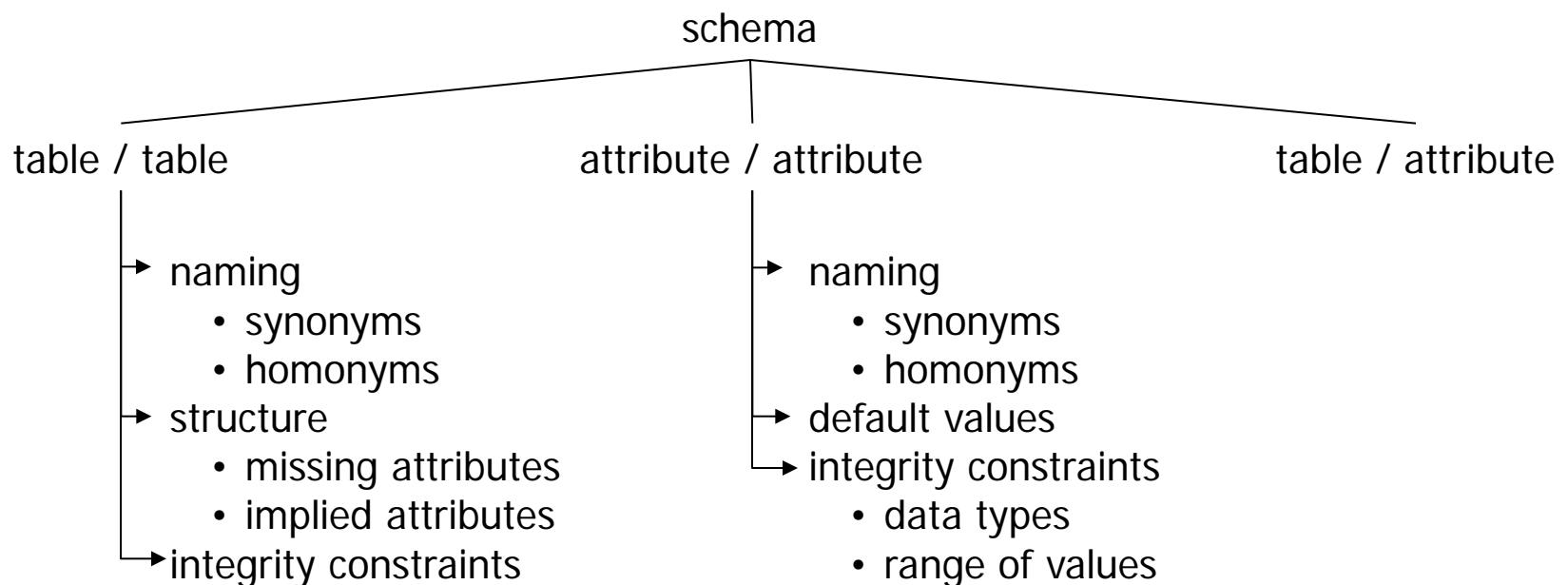
- **Goal:** Take two schemas as input and produce a mapping between elements of the two schemas that correspond semantically to each other.
- Typically performed manually, supported by a graphical user interface.
 - tedious, time-consuming, error-prone, expensive
- General architecture of generic match:
 - tools = schema-related apps.
 - internal schema representation + import and export needed
 - use libraries to help find matches
 - only determine match candidates
 - user may accept or reject



Schema Comparison

Analysis to determine the correlations among concepts of different schemata and to detect possible conflicts.

- Schema Matching is part of this step.
- Types of conflicts in relational systems:

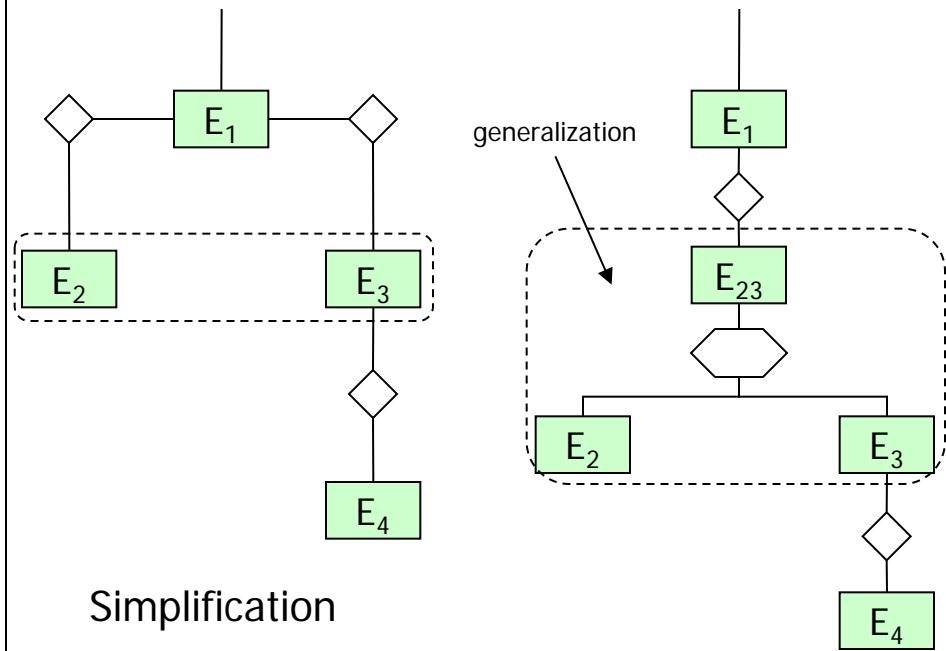
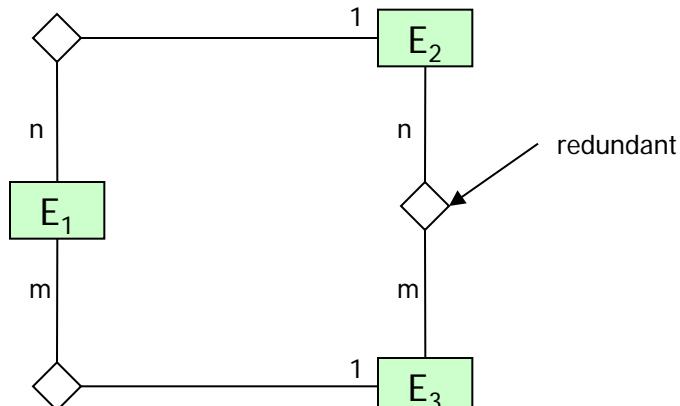
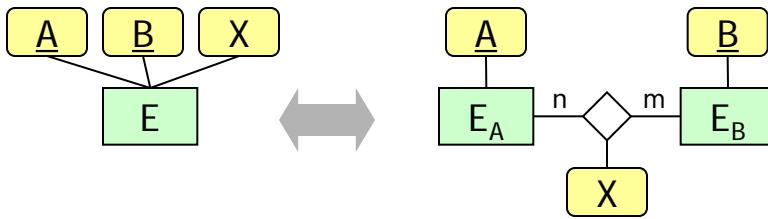
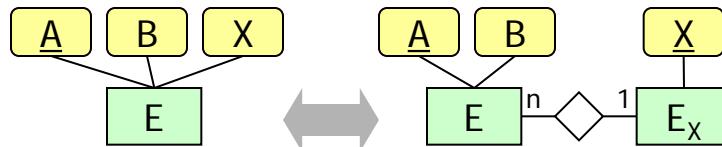


Schema Conforming

Conform and align schemata to make them compatible for integration.

- Conflict resolution
 - based on the application context
 - cannot be fully automated
 - human intervention supported by graphical interfaces
- Sample steps:
 - Attributes vs. Entity Sets
 - Composite Primary Keys
 - Redundancies
 - Simplification

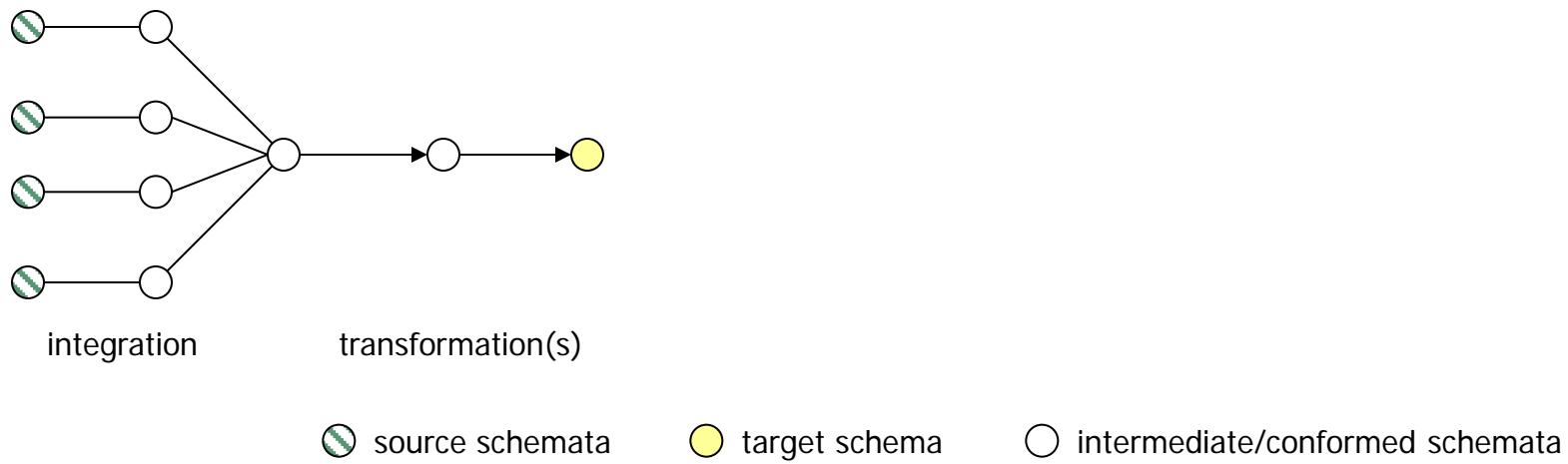
Schema Conforming



Schema Merging and Restructuring

Conformed schemata are superimposed, thus obtaining a global schema.

- Main steps:
 - superimpose conformed schemata
 - quality tests against quality dimensions (completeness, correctness, minimality, understandability, ...)
 - further transformation of the obtained schema



Schema Integration in Data Warehousing

Integration in data warehousing

- schema integration and data integration
- schema integration is a prerequisite for data integration
- schema integration is mainly used for the data staging area
- final data warehouse schema is defined from a global point of view,
i.e., it is more than only integrating all source schemata
- schema matching between source schema and data warehouse schema provides the basis for defining transformations

Integration in federated systems

- focus on schema integration
- integrated schema is used

Data Integration

- Normalization / denormalization:
 - depending on the source schema and the data warehouse schema
- Surrogate keys:
 - keys should not depend on the source system
- Data type conversion:
 - if data type of source attribute and target attribute differ
- Coding:
 - text → coding; coding → text;
coding A → coding B

examples

customer		
system	local key	global key
1	107	5400345
1	109	5401340
2	107	4900342
2	214	5401340

character → date
character → integer
'MM-DD-YYYY' → 'DD.MM.YYYY'

gross sales → 1
net sales → 2
3 → price
2 → GS

Data Integration

- Convert strings:
 - standardization
- Convert date to date format of the target system
- Convert measures
- Combine / separate attributes
- Derived attributes
- Aggregation

e x a m p l e s

'Video' → ' video'
'VIDEO' → 'video'
'Miller, Max' → 'Max Miller'

2004, 05, 31 → 31.05.2004
04, 05, 31 → 31.05.2004
'05/31/2004' → 31.05.2004

inch → cm
km → m
mph → km/h

2004, 05, 31 → 31.05.2004
'video', 1598 → 'video 1598'

net sales + tax → gross sales
on_stock - on_order → remaining

sales_per_day → sales_per_month

Data Cleansing

- Elementizing
 - identify fields
- Standardizing
 - format, coding

David and Clara Miller
Ste. 116
13150 Hiway 9
Box 1234 Boulder Crk
Colo 95006

first name 1: David
last name 1: Miller
first name 2: Clara
last name 2: Miller
suite: 116
number: 13150
street: Hiway 9
post box: 1234
city: Boulder Crk
state: Colo
zip: 95006

first name: David
last name: Miller
first name 2: Clara
last name 2: Miller
suite: 116
number: 13150
street: Highway 9
post box: 1234
city: Boulder Creek
state: Colorado
zip: 95006

- Verification
 - contradictions?
 - should lead to corrections in source system(s)
- Matching
 - is 'David Miller' and/or 'Clara Miller' already present in data warehouse?
 - if so, are there changed fields?
- Householding
 - 'David Miller' and 'Clara Miller' constitute a household
- Documenting

first name: David
last name: Miller
first name 2: Clara
last name 2: Miller
suite: 116
number: 13150
street: Highway 9
post box: 1234
city: Boulder Creek
state: California
zip: 95006

Dimensions of Data Cleansing

	single source	multiple sources
single record	<ul style="list-style-type: none">• attribute dependencies (contradictions)• spelling mistakes• missing values• illegal values	<ul style="list-style-type: none">• duplicates / matching• householding• contradictions• standardization, coding
multiple records	<ul style="list-style-type: none">• primary key foreign key• duplicates / matching• householding	

Data Quality

consistency

correctness

completeness

exactness

reliability

understandability

relevance

- Are there contradictions in data and/or metadata?
- Do data and metadata provide an exact picture of the reality?
- Are there missing attributes or values?

- Are exact numeric values available?
Are different objects identifiable? Homonyms?
- Is there a Standard Operating Procedure (SOP) that describes the provision of source data?
- Does a description for the data and coded values exist?
- Does the data contribute to the purpose of the data warehouse?

Improving Data Quality

- Assumption:
 - Various projects can be undertaken to improve the quality of warehouse data.
 - **Goal:** Identify the data quality enhancement projects that maximize value to the users of data.
- Tasks for the data warehouse manager:
 - Determine the organizational activities the data warehouse will support;
 - Identify all sets of data needed to support the organizational activities;
 - Estimate the quality of each data set on each relevant data quality dimension;
 - Identify a set of potential projects (and their cost) that could be undertaken for enhancing or affecting data quality;
 - Estimate for each project the likely effect of that project on the quality of the various data sets, by data quality dimension;
 - Determine for each project, data set, and relevant data quality dimension the change in utility should a particular project be undertaken.

Improving Data Quality

- I: Index of organizational activities supported by a data warehouse
- J: Index for data sets
- K: Index for data quality attributes or dimensions
- L: Index for possible data quality projects P(1) ... P(S)

Current quality: CQ(J, K)
 Required quality: RQ(I, J, K)
 Anticipated quality: AQ(J, K, L)
 Priority of organizational activities: Weight(I)
 Cost of data quality enhancement: Cost(L)
 Value added: Utility(I, J, K, L)

$$\text{Value of Project L} = \sum_{\text{All } I} \text{Weight}(I) \cdot \sum_{\text{All } J} \sum_{\text{All } K} \text{Utility}(I, J, K, L)$$

$$X(L) = \begin{cases} 1, & \text{if Project L is selected} \\ 0, & \text{otherwise} \end{cases}$$

$$\text{Maximize: total value from all projects} \quad \sum_{\text{All } L} X(L) * \text{Value}(L)$$

$$\text{Resource Constraint:} \quad \sum_{\text{All } L} X(L) * \text{Cost}(L) \leq \text{Budget}$$

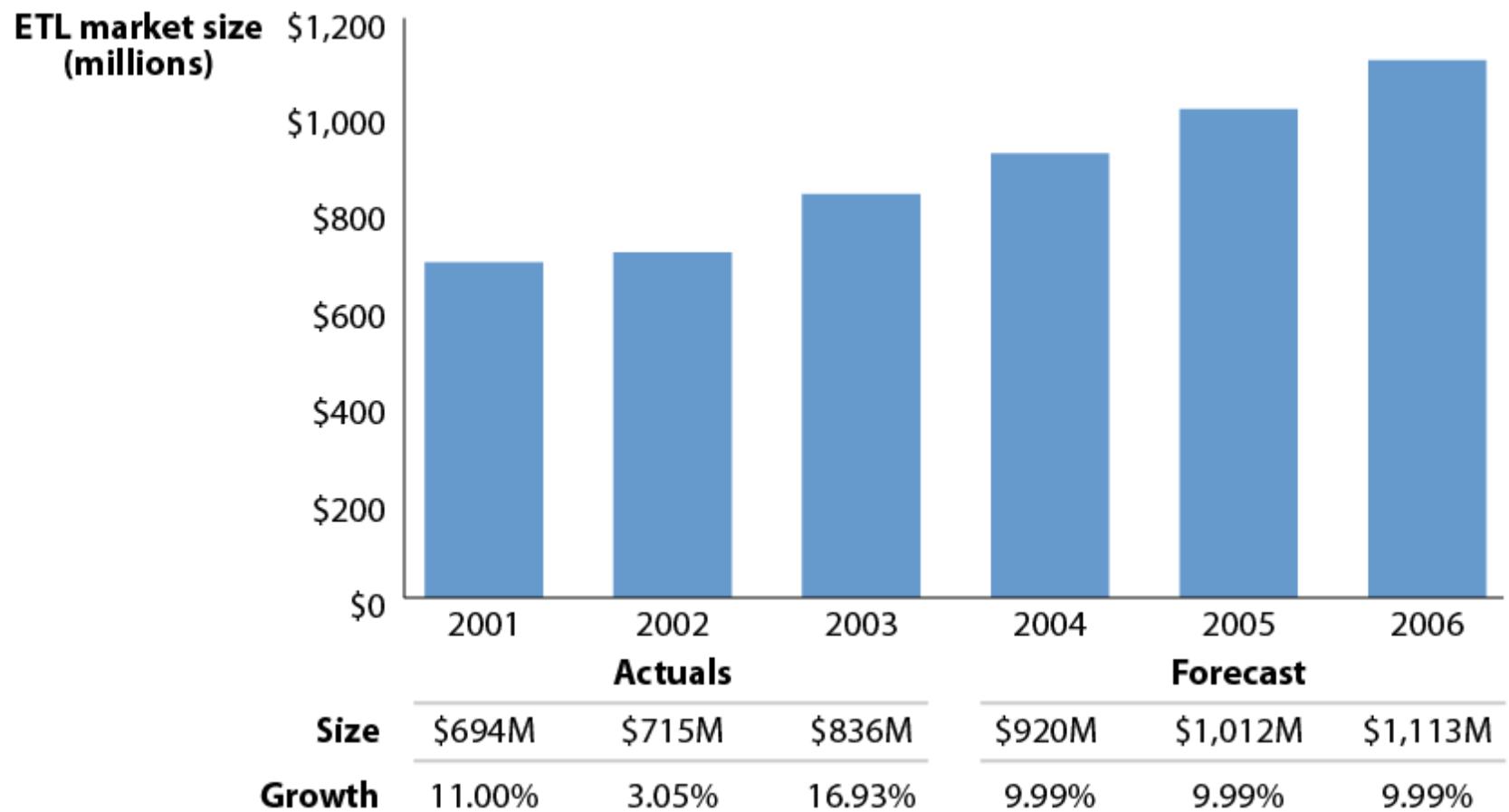
$$\text{Exclusiveness Constraint:} \quad X(P(1)) + X(P(2)) + \dots + X(P(S)) \leq 1$$

$$\text{Interaction Constraint:} \quad X(P(1)) + X(P(2)) + X(P(3)) \leq 1$$

Overview

- Monitoring
- Extraction
 - Export, Import, Filter, Load
 - Direct Integration
- Load
 - Bulk Load
 - Replication
 - Materialized Views
- Transformation
 - Schema Integration
 - Data Integration
 - Data Cleansing
- Tools

ETL Market Size 2001-2006



ETL Market

- Vendors coming from several different backgrounds and perspectives:

(A) "Pure Play" Vendors

- ETL represents a core competency
- ETL accounts for most of the license revenue
- This class of vendors is driving the bulk of innovation and "mind share" in the ETL market

(C) Database Management System (DBMS) Vendors

- database vendors have an increasing impact on this market as they continue to bundle ETL functionality closer to the relational DBMS

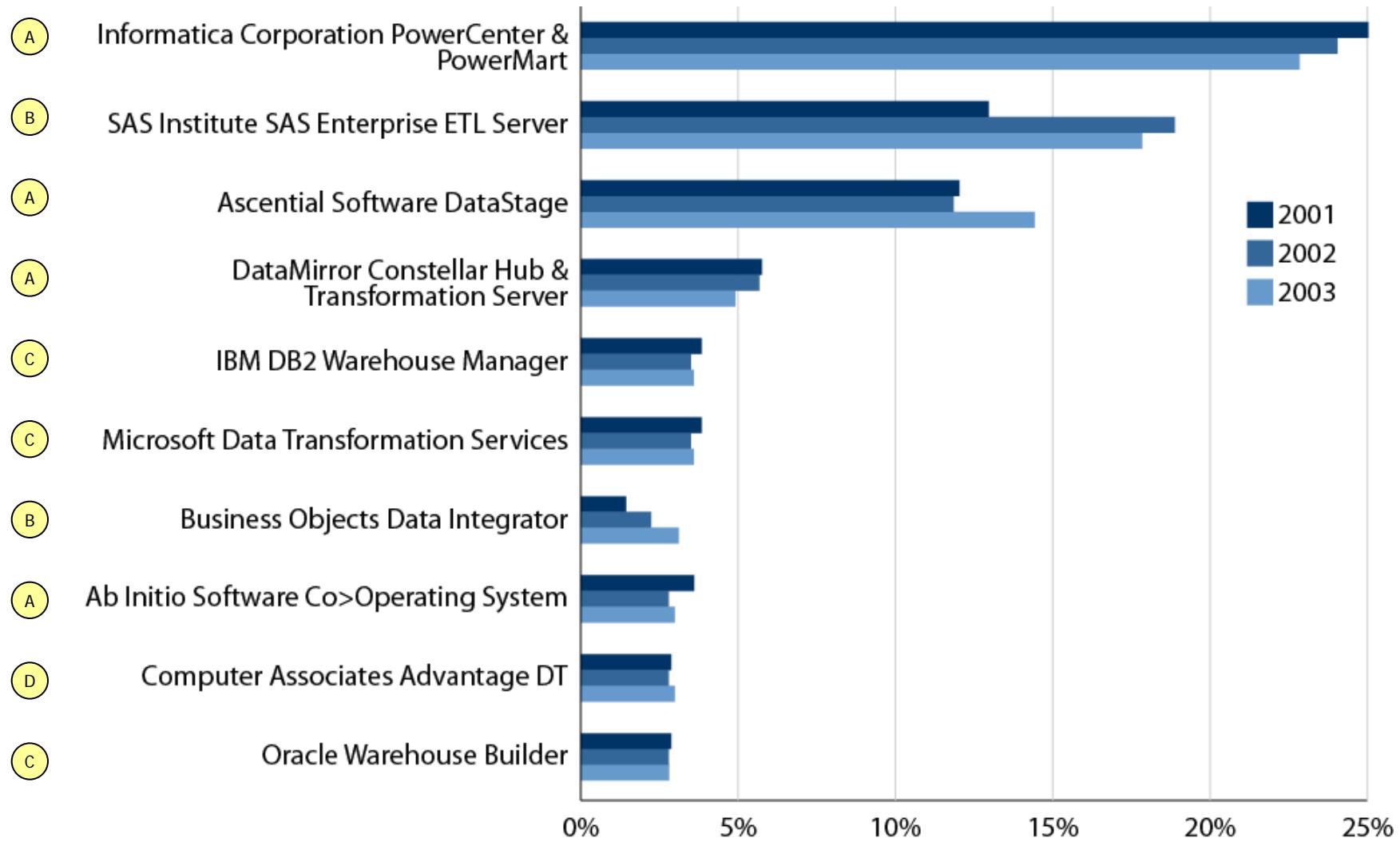
(B) Business Intelligence Vendors

- Business intelligence tools and platforms are their core competency
- For most of these vendors, ETL technology plays a supporting role to their flagship business intelligence offerings, or is one component of a broad offering including business intelligence and ETL

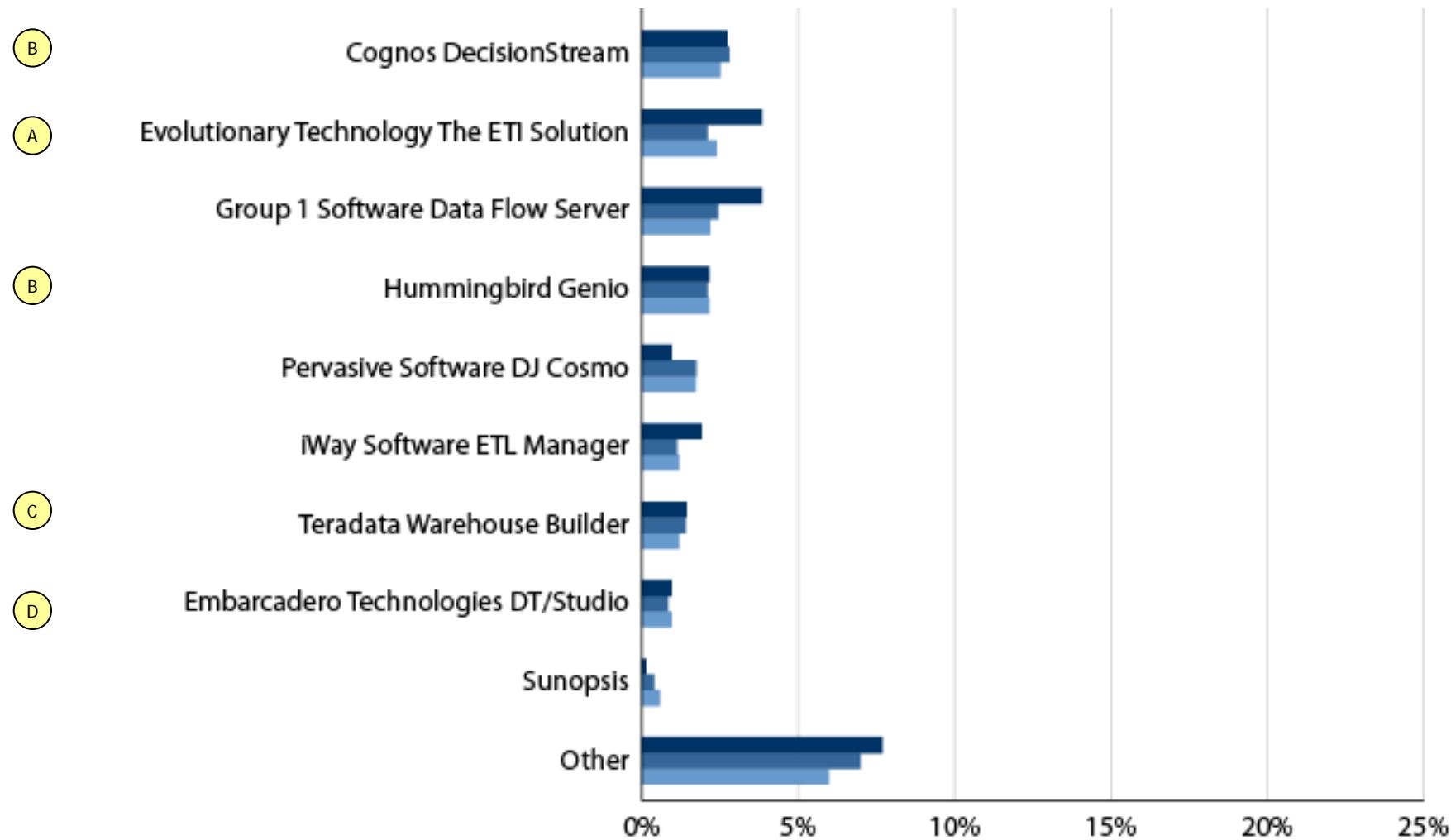
(D) Other Infrastructure Providers

- They provide various types of technical infrastructure components beyond the DBMS
- ETL is typically positioned as yet another technical toolset in their portfolios

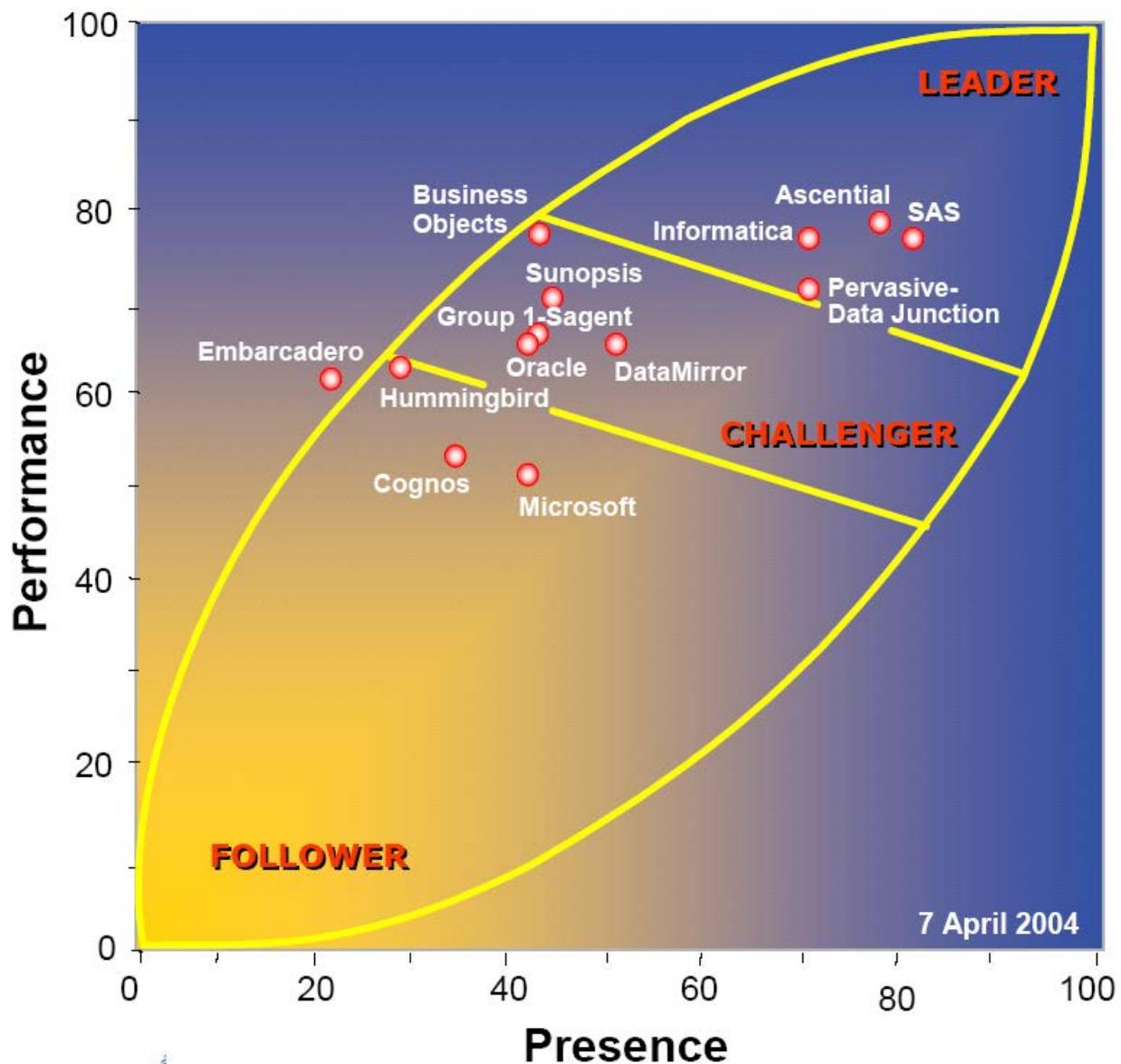
ETL Vendors/Products Ranked By Market Share Percentages



ETL Vendors/Products Ranked By Market Share Percentages



ETL Tools



Summary

- Moving data is part of most steps of the ETL process:
Extraction, Transformation, Loading Data Warehouse and Data Marts
- Several approaches available:
 - export, import, load
 - direct integration
 - replication
 - materialized views
- Transformation steps include
 - semi-automatic schema matching and integration
 - data integration steps
 - data cleansing

Papers

- [BT99] D. Ballou, G. K. Tayi: Enhancing Data Quality in Data Warehouse Environments. Communications of the ACM, Vol. 42, No. 1, 1999.
- [LG96] W. Labio, H. Garcia-Molina: Efficient Snapshot Differential Algorithms for Data Warehousing. Proc. of 22th International Conference on Very Large Data Bases, Mumbai (Bombay), India, 1996.
- [RB01] E. Rahm, P. Bernstein: A survey of approaches to automatic schema matching. VLDB Journal 10:334-350 (2001)

Data-Warehouse-, Data-Mining- und OLAP-Technologien

Chapter 5: Online Analytic Processing

Bernhard Mitschang
Universität Stuttgart

Winter Term 2015/2016

Overview

- OLAP
 - Introduction
 - Operations
 - Characteristics
- Storage of OLAP cubes
 - Relational vs. Multidimensional
 - Multidimensional Arrays
 - Sparse Cubes
 - Multidimensional Query Language
- Architecture
 - MOLAP, ROLAP, HOLAP

OLAP

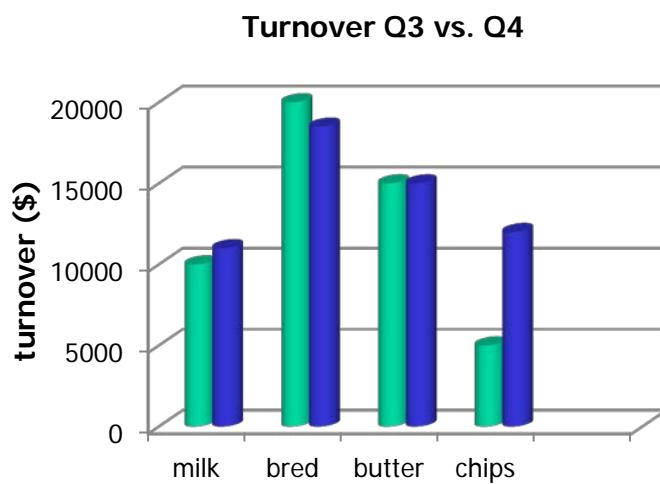
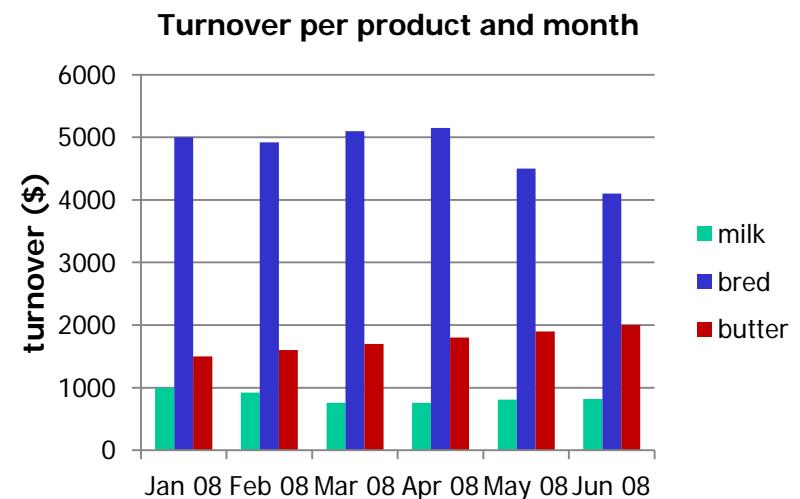
- Online Analytic Processing
- Technologies and tools that support (ad-hoc) analysis of multi-dimensionally aggregated data
- Individual analysis is supported, i.e., the user is not restricted to available standard reports/analysis
- Graphical user interface is available for analysis specification
- Knowledge of a query language or programming language is not required
- Result information is given graphically and made available for incorporation into other applications
- Users: Analysts, Manager, "knowledge worker"
- Typical analysis scenarios:
 - Multi-dimensional views, e.g. turnover per product group and month
 - Comparisons, e.g. turnover in Q4 compared to that of Q3
 - Ranking, e.g. top 10 product in a certain group ranked by turnover

OLAP

- Defining OLAP reports/Analysis
 - select facts
 - select dimensions
 - define filters
 - define presentation

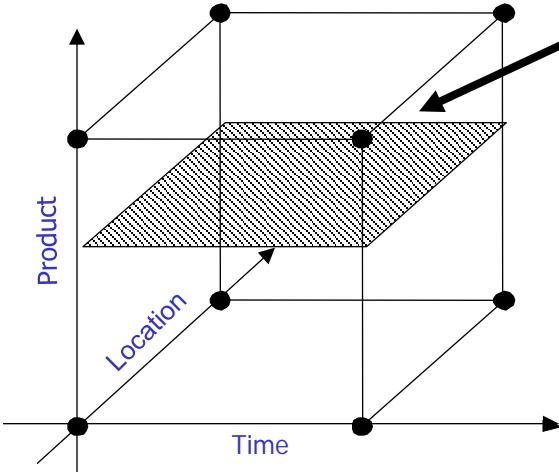
Top 10 fruit and vegetables

Rank	Produkt	Turnover (\$)
1	potatoes	210000
2	carrots	205000
3	celery	190000
3	tomatoes	190000
5	kiwi fruit	150000
6	strawberry	145000
7	spinach	142000
8	zucchini	95500
9	lettuce	94000
10	blackberry	92000



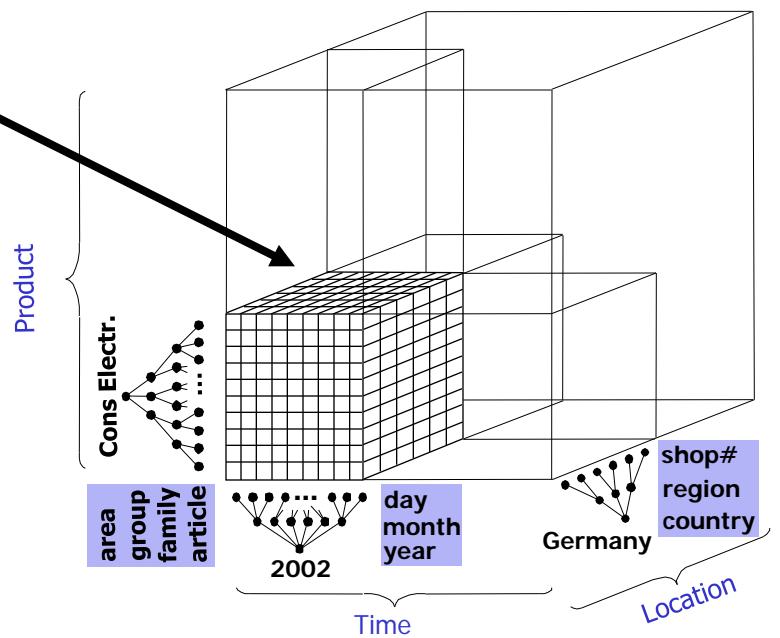
Multidimensional Model

Multidimensional Model

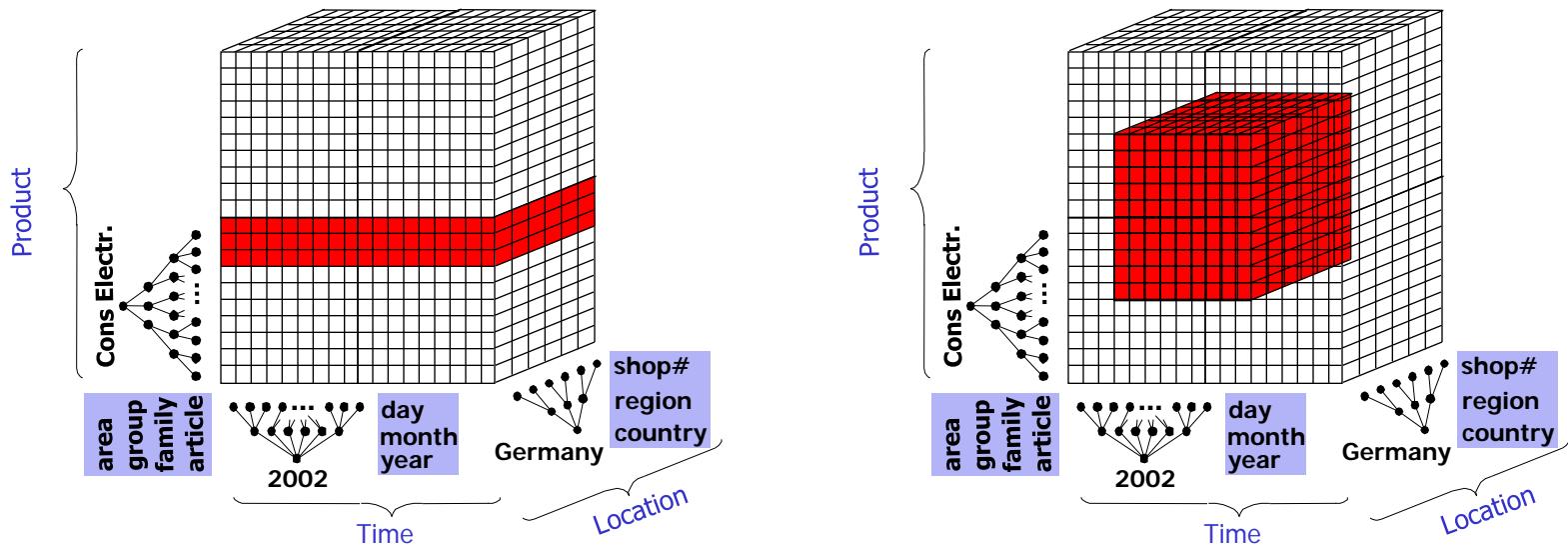


Fact Data
(sales)

„Cube“ Metaphor

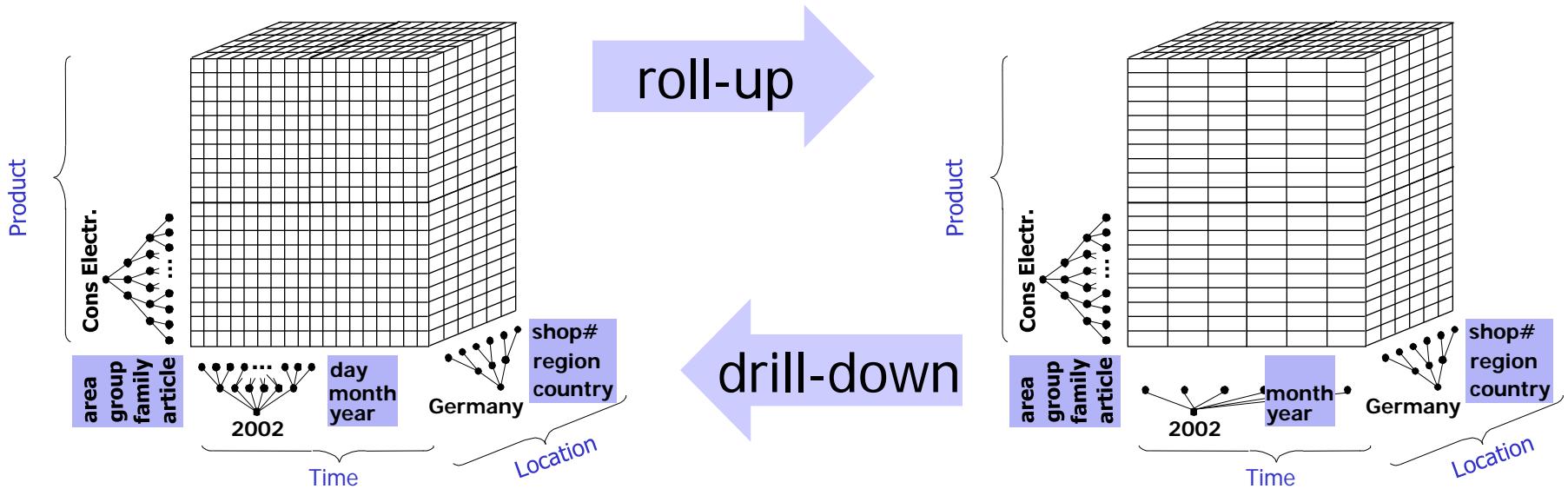


Slice and Dice



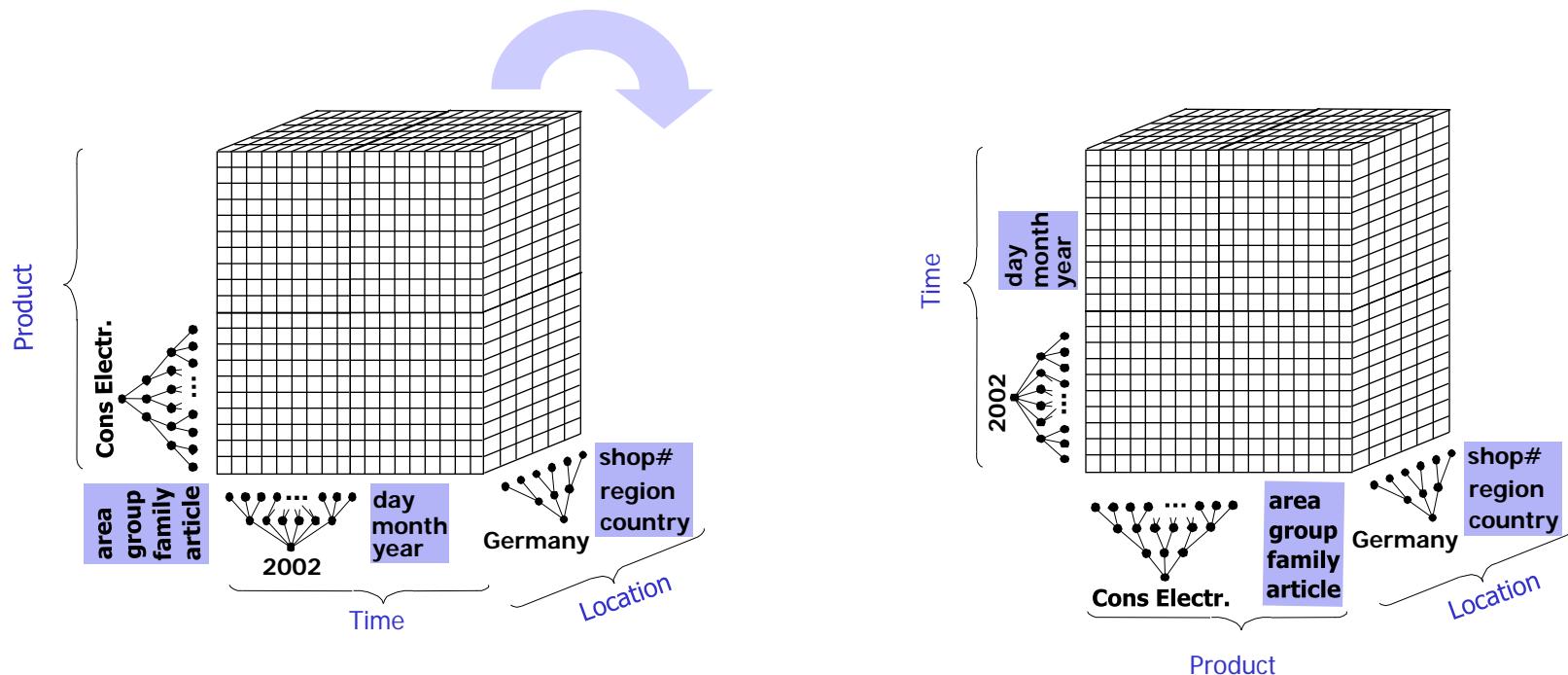
- **Slice:**
 - restrict one dimension to a range of values
- **Dice:**
 - restrict several dimensions to a range of values
 - results in a sub-cube
- Example: Analysis of a certain product family.

Roll-up and Drill-down



- Roll-up (drill-up):
 - summarize data by climbing up hierarchy or by dimension reduction
- Drill-down (roll-down):
 - reverse of roll-up
 - from higher level summary to lower level summary or detailed data, or introducing new dimensions

Pivot and Rotate



- Pivot:
 - reorient the cube
 - visualization
 - 3D to series of 2D planes

OLAP Operations: Overview

- Typical OLAP operations (explained in a general manner):
 - **Roll up** (drill-up): summarize data
 - by climbing up hierarchy or by dimension reduction
 - **Drill down** (roll down): reverse of roll-up
 - from higher level summary to lower level summary or detailed data, or introducing new dimensions
 - **Slice and dice**:
 - project and select
 - **Pivot** (rotate):
 - reorient the cube, visualization, 3D to series of 2D planes.
 - Other operations
 - **drill across**: involving (across) more than one fact table
 - **drill through**: through the bottom level of the cube to its back-end relational tables (using SQL)

OLAP Product Evaluation Rules

Basic Features

- R1: multi-dimensional conceptual view
- R10: intuitive data manipulation
- R3: accessibility
- N: batch extraction vs. interpretive
- N: OLAP analysis models
- R5: client-server architecture
- R2: transparency
- R8: multi-user support

Reporting Features

- R11: flexible reporting
- R4: consistent reporting performance
- R7: dynamic sparse matrix handling

Dimension Control

- R6: generic dimensionality
- R12: unlimited dimensions and aggregation levels
- R9: unrestricted cross-dimensional operations

Special Features

- N: treatment of non-normalized data
- N: storing OLAP results: keeping them separate from source data
- N: extraction of missing values
- N: treatment of missing values

R1 - R12: original rules

N: additional rules

FASMI Test

FAST	<ul style="list-style-type: none">• deliver most responses within about five seconds• simplest analysis taking no more than one second• very few taking more than 20 seconds
ANALYSIS	<ul style="list-style-type: none">• cope with any business logic and statistical analysis that is relevant for applications and users• allow users to define new ad-hoc calculations without programming
SHARED	<ul style="list-style-type: none">• confidentiality• concurrent update locking if multiple write access is needed
MULTIDIMENSIONAL	<ul style="list-style-type: none">• multidimensional conceptual view of data• support for hierarchies and multiple hierarchies
INFORMATION	<ul style="list-style-type: none">• handle huge amounts of input data

(Nigel Pendse: What is OLAP? www.olapreport.com, 2004)

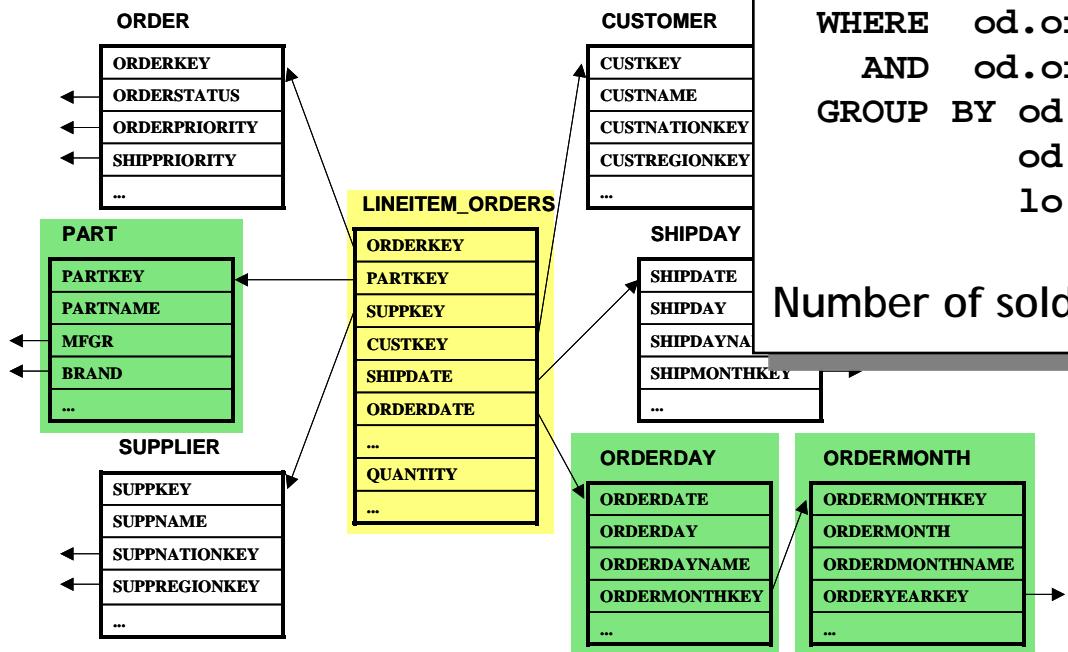
Overview

- OLAP
 - Introduction
 - Operations
 - Characteristics
- Storage of OLAP cubes
 - Relational vs. Multidimensional
 - Multidimensional Arrays
 - Sparse Cubes
 - Multidimensional Query Language
- Architecture
 - MOLAP, ROLAP, HOLAP

Relational Storage of OLAP Cubes

- Mapping the cube view to a star- or snowflake-schema
- Information requests of the users have to be mapped to the relational schema (see 'sequence of typical star queries')
- Result tables have to be mapped to the cube structure before they are presented to the user

Sequence of typical star queries (1)



```
INSERT INTO A1 (orderyearkey, ordermonthkey,
                partkey, sumquantity)
```

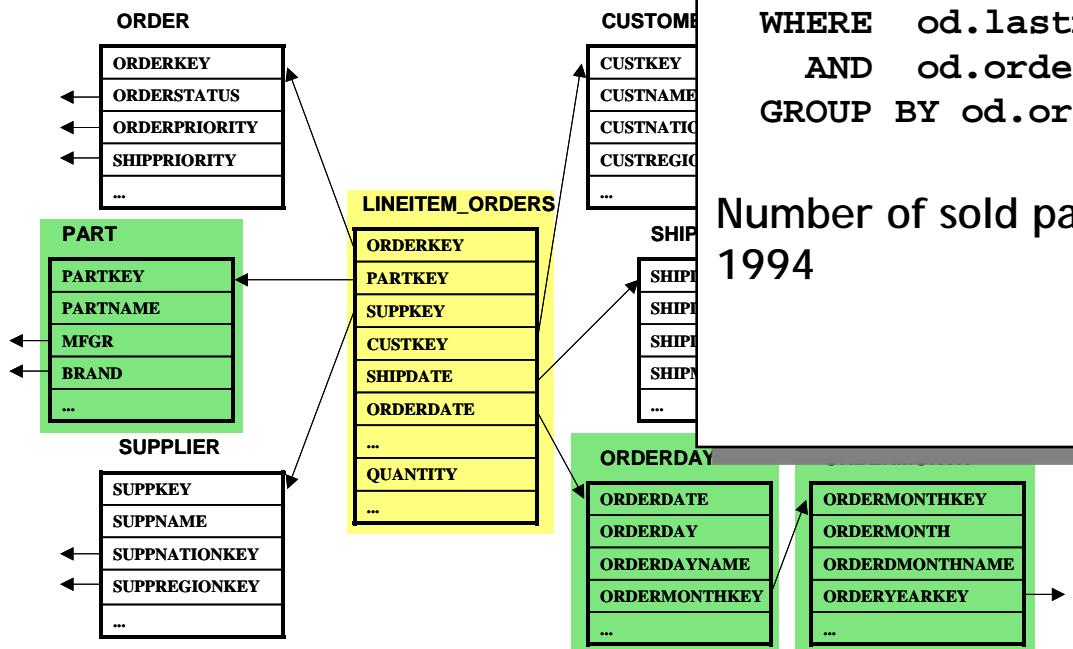
```
SELECT od.orderyearkey, od.ordermonthkey,
       lo.partkey, SUM(lo.quantity)
  FROM lineitem_orders lo, orderday od
 WHERE od.orderdate = lo.orderdate
   AND od.ordermonthkey IN (199401,199402)
 GROUP BY od.orderyearkey,
          od.ordermonthkey,
          lo.partkey;
```

Number of sold parts in January and February 1994

Information request:

Which are the top products whose number of sold pieces in the months chosen by the user compared to the respective month ago has increased most?

Sequence of typical star queries (2)



```
INSERT INTO A2 (ordermonthkey, partkey,
sumquantity)
```

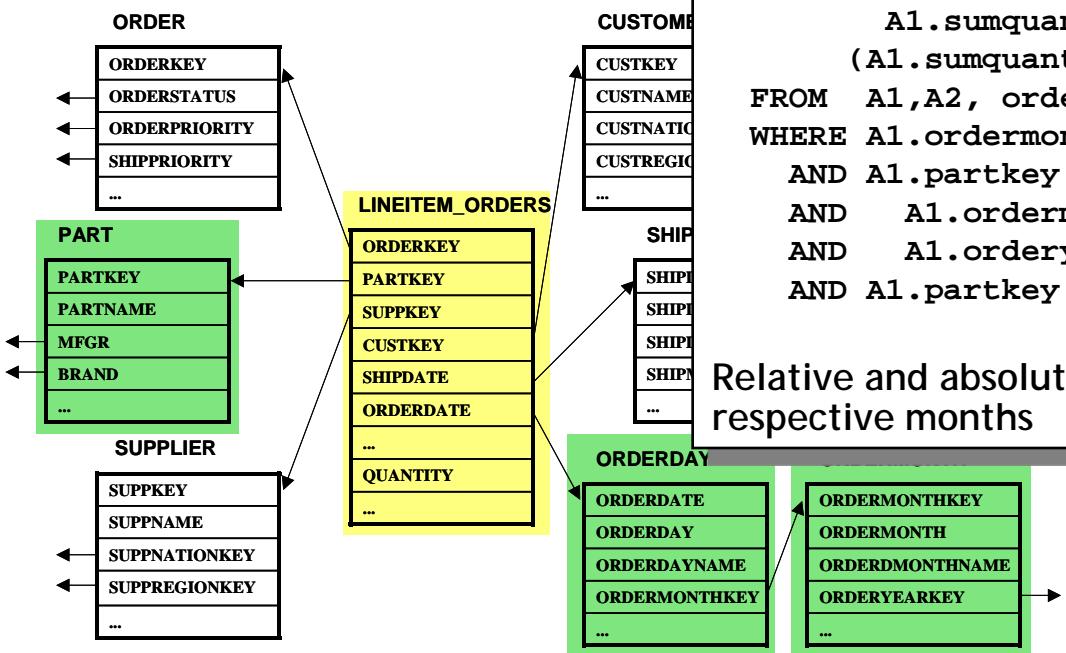
```
SELECT od.ordermonthkey, lo.partkey,
SUM(lo.quantity)
FROM lineitem_orders lo, orderday od
WHERE od.lastmonthdate = lo.orderdate
AND od.ordermonthkey IN (199401, 199402)
GROUP BY od.ordermonthkey, lo.partkey;
```

Number of sold parts in December 1993 and January 1994

Information request:

Which are the top products whose number of sold pieces in the months chosen by the user compared to the respective month ago has increased most?

Sequence of typical star queries (3)



```

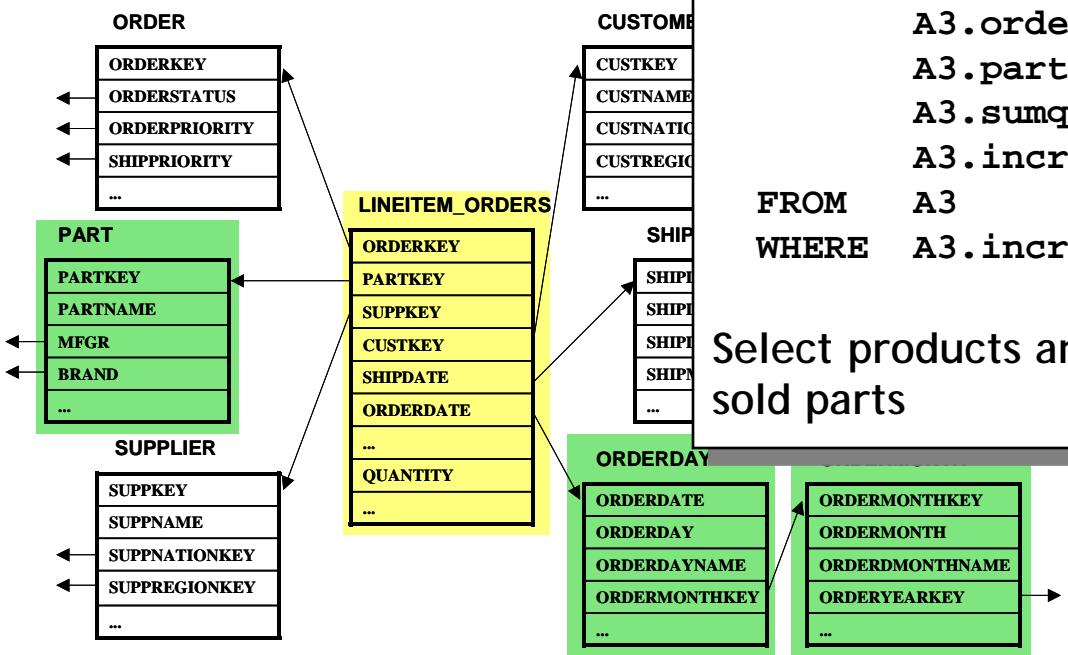
INSERT INTO A3 (ordermonthkey, ordermonthname,
                orderyearkey, orderyear, partkey,
                partname, sumquantity, lmsumquantity,
                incrquantity, incrquantity2)
SELECT om.ordermonthkey, om.ordermonthname,
       oy.orderyearkey, oy.orderyear, pa.partkey,
       pa.partname, A1.sumquantity, A2.sumquantity,
       A1.sumquantity - A2.sumquantity,
       (A1.sumquantity - A2.sumquantity)/A2.sumquantity
FROM   A1,A2, ordermonth om, orderyear oy, part pa
WHERE  A1.ordermonthkey = A2.ordermonthkey
       AND A1.partkey = A2.partkey
       AND A1.ordermonthkey = om.ordermonthkey
       AND A1.orderyearkey = oy.orderyearkey
       AND A1.partkey = pa.partkey;
  
```

Relative and absolute increase of sold parts compared to respective months

Information request:

Which are the top products whose number of sold pieces in the months chosen by the user compared to the respective month ago has increased most?

Sequence of typical star queries (4)



```

INSERT INTO A4 (ordermonthkey, ordermonthname,
                orderyearkey, orderyear,
                partkey, partname,
                sumquantity, lmsumquantity,
                incrquantity, incrquantity2)

SELECT A3.ordermonthkey, A3.ordermonthname,
       A3.orderyearkey, A3.orderyear,
       A3.partkey, A3.partname,
       A3.sumquantity, A3.lmsumquantity,
       A3.incrquantity, A3.incrquantity2
FROM   A3
WHERE  A3.incrquantity2 >= 98;

```

Select products and months with highest increase of sold parts

Information request:

Which are the top products whose number of sold pieces in the months chosen by the user compared to the respective month ago has increased most?

Multidimensional Storage of OLAP Cubes

- Allows to directly store the cells of a data cube in a n-dimensional array
- Avoids mapping between cube view and relational schema
- May result in sparse cubes
- Multidimensional query language needed

Multidimensional Database Systems

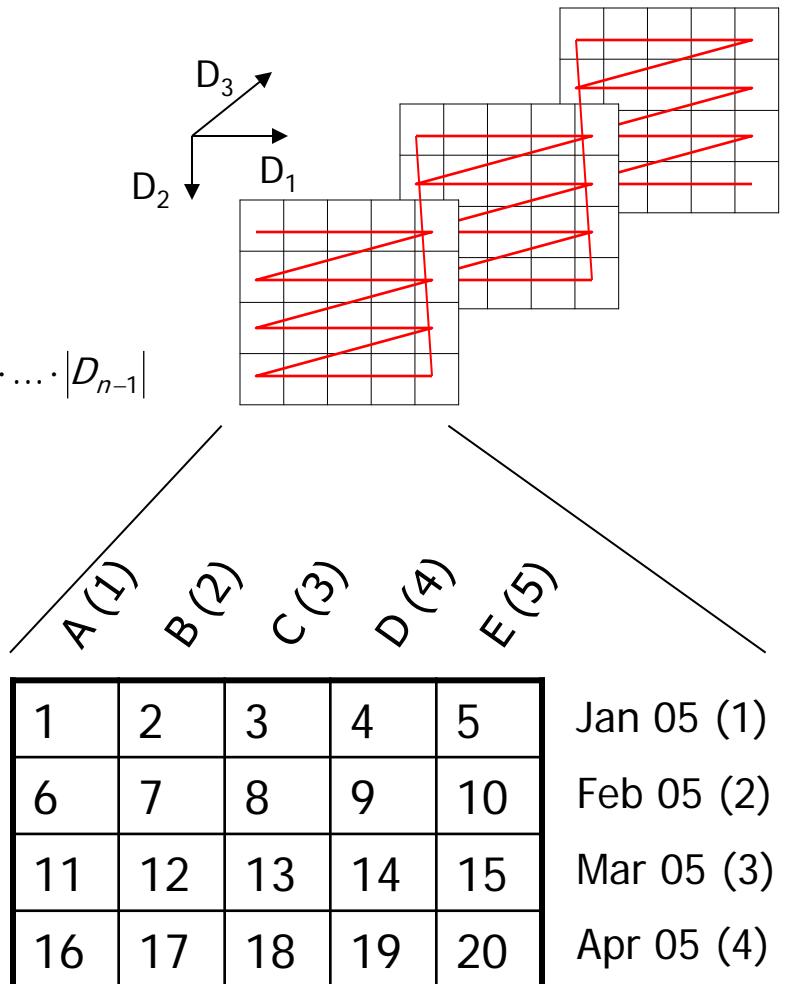
- Allow to directly store the cells of a data cube in a n-dimensional array.

	single cube	many cubes
single measure per cube		<ul style="list-style-type: none">relevant dimensionality for each measure
multiple measures per cube	<ul style="list-style-type: none">sparse dimensions likely	<ul style="list-style-type: none">direct mapping of the conceptual model

- Many proprietary implementations of storage structure:
 - similar to common index structures

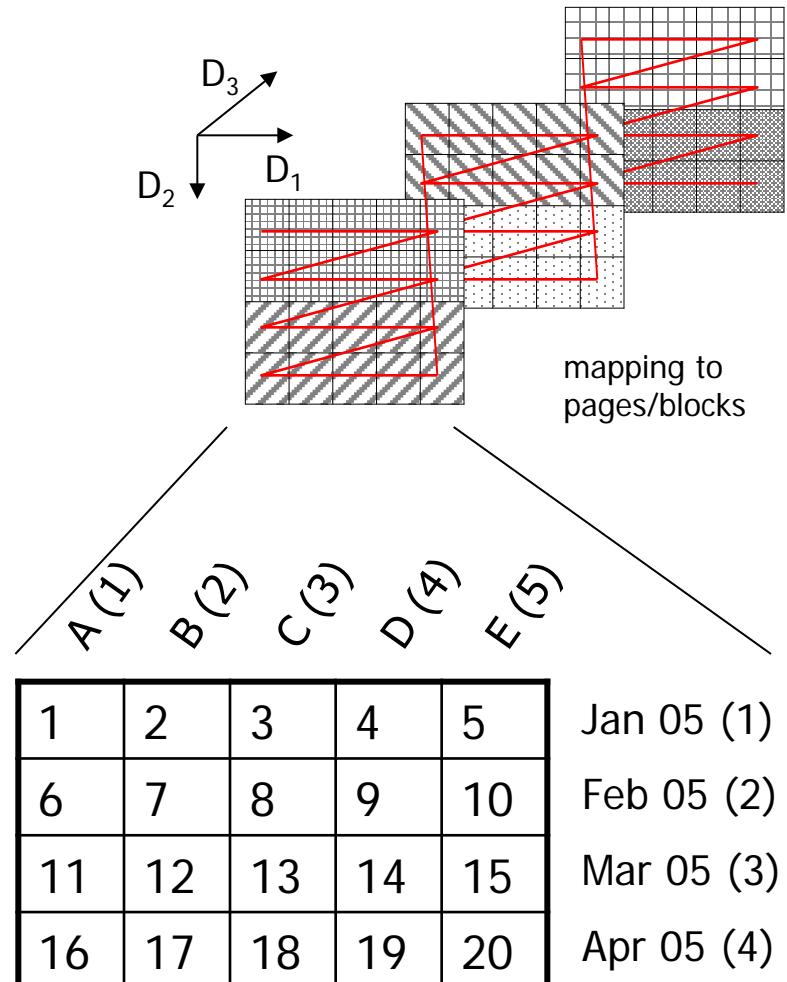
Multidimensional Arrays

- Dimensions D_1, \dots, D_n
- Data cube with $|D_1| * |D_2| * \dots * |D_n|$ cells
- Index of cell (x_1, x_2, \dots, x_n)
$$= x_1 + (x_2 - 1) \cdot |D_1| + (x_3 - 1) \cdot |D_1| \cdot |D_2| + \dots + (x_n - 1) \cdot |D_1| \cdot \dots \cdot |D_{n-1}|$$
$$= 1 + \sum_{i=1}^n (x_i - 1) \cdot \prod_{j=1}^{i-1} |D_j|$$
- Example:
 - Dimension 1: Product
 - Dimension 2: Month
 - Which cell stores data for product C in April 2005?



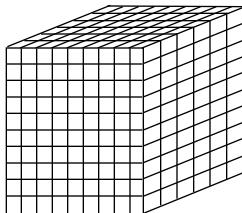
Query Processing in Multidimensional Arrays

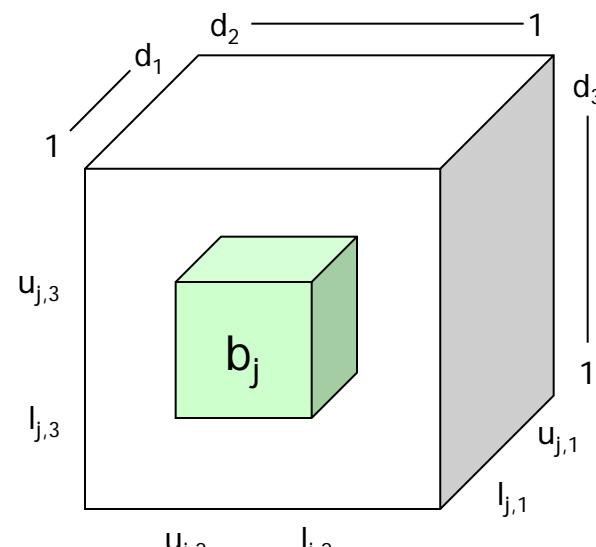
- Query processing:
 - determine index of cells
 - read pages/blocks for these cells into main memory
- Query performance depends on the number of pages to be read.
- Example:
 - How many blocks need to be read to get all cells on product A?
 - How many blocks need to be read to get all cells for February 2005?
- Order of dimensions is significant for query performance.

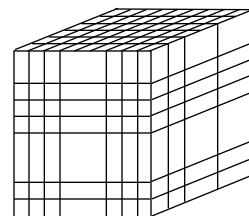


Multidimensional Partitioning

- Dimensions D_1, \dots, D_n
- Dimension values $1 \dots d_i$ for each dimension D_i .
- Partition b_1, \dots, b_m as
$$b_1 = [l_{1,1}:u_{1,1}, \dots, l_{1,n}:u_{1,n}]$$
$$\dots$$
$$b_m = [l_{m,1}:u_{m,1}, \dots, l_{m,n}:u_{m,n}]$$
- regular partitioning:
same value range in dimension D_i for each partition b_j .



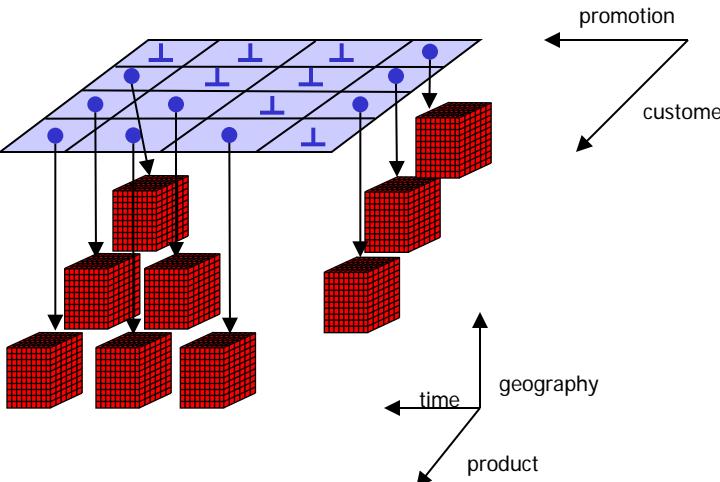
- 
- irregular partitioning:
partition-specific value ranges



Multidimensional Partitioning

- Automatic partitioning:
 - system automatically defines the partitioning
 - goals:
 - identify sparse dimensions
 - efficient query processing
- Partitioning based on dimension semantics:
 - e.g. partitioning according to time series
- user-defined partitioning:
 - explicit specification based on
 - value ranges
 - dimensions
- Storage of partitions:
 - relational: coordinates of cells are stored as primary key in a table
 - array: cells are stored in an array (as shown before)

Sparse Cubes

- A cube may contain empty cells.
 - Density of a cube:
$$= \frac{\text{number of defined cells}}{\text{number of all cells}}$$
 - N-dimensional array is efficient for dense cubes.
 - Sparse cubes need further optimizations:
 - don't store empty pages/blocks
 - multidimensional partitioning + two storage levels
 - Two storage levels:
 - first level:
 - index structure for sparse dimensions
 - index structures like B-trees, Grid, Hashing
 - second level:
 - n-dim. array for dense dimensions
 - compressed arrays
- 

Multidimensional Query Language

- Query language that includes specific features for multidimensional data:
 - access to cubes
 - access to dimensions
 - aggregation of measure
 - restrictions on dimensions
 - selection of subcubes
 - set of functions for the manipulation of data
- No standard available
- Most tools provide queries based on the information users requested by means of a graphical user interface
- Example: MDX (MultiDimensional Expression)
 - published in 1998
 - part of Microsofts OLE DB for OLAP
 - OLE DB provides COM interfaces for access to various data sources
 - supports the definition and manipulation of multidimensional objects and data (DML and DDL statements)

MDX

- Basic syntax:

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

<axis_specification> ::= <set> ON <axis_name>

<axis_name> ::= COLUMNS | ROWS | PAGES | SECTIONS | CHAPTERS | AXIS(<index>)

- SELECT clause:
 - determines the axis dimensions of an MDX SELECT statement
- FROM clause:
 - determines which multidimensional data source is to be used when extracting data to populate the result set

- WHERE clause:

- determines which dimension or member to use as a slicer dimension
- slicer dimension = dimension that is not assigned to an axis
- restricts the extracting of data to a specific dimension or member

MDX: Examples

```
SELECT { [Measures].[Unit Sales], [Measures].[Store Sales] } ON COLUMNS,  
       { [Time].[1997], [Time].[1998] } ON ROWS  
FROM   Sales  
WHERE  ( [Store].[USA].[CA] )
```

- Specifies that:
 - two measures should be presented in columns
 - values for two years should be presented in rows
 - only stores in CA should be included
- WHERE clauses

tuple

```
WHERE ( [Route].[All], [Time].[1st half] )  
WHERE { ([Time].[1st half], [Route].[nonground]),  
        ([Time].[1st half], [Route].[ground])}
```

set

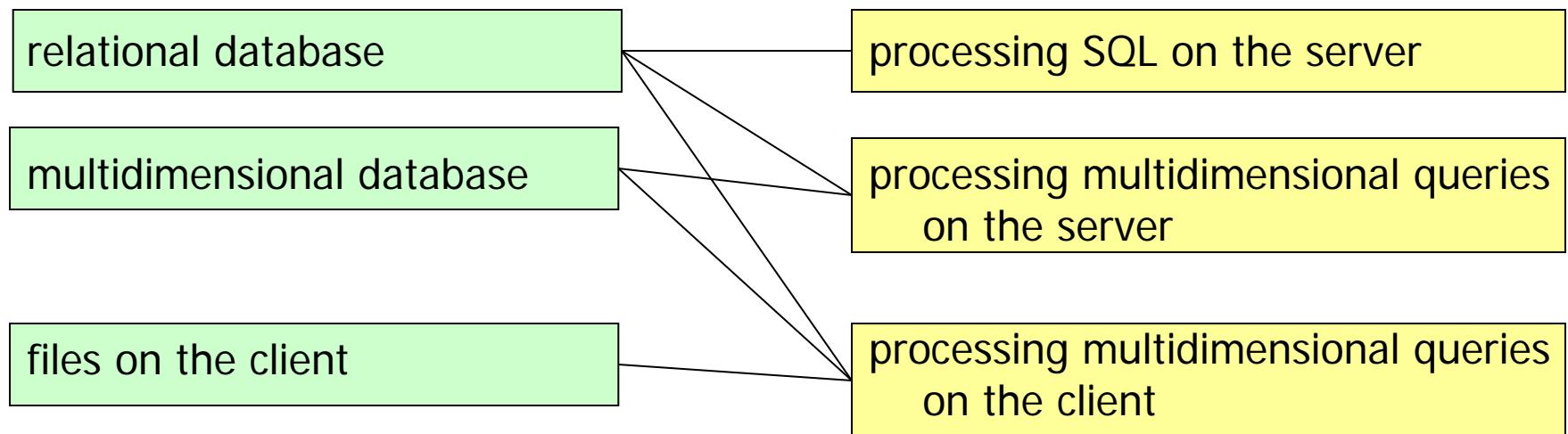
- tuple: uniquely identifies a section in the cube (subcube)
- if multiple tuples are specified (set) result cells in every tuple along the set will be aggregated

Overview

- OLAP
 - Introduction
 - Operations
 - Characteristics
- Storage of OLAP cubes
 - Relational vs. Multidimensional
 - Multidimensional Arrays
 - Sparse Cubes
 - Multidimensional Query Language
- Architecture
 - MOLAP, ROLAP, HOLAP

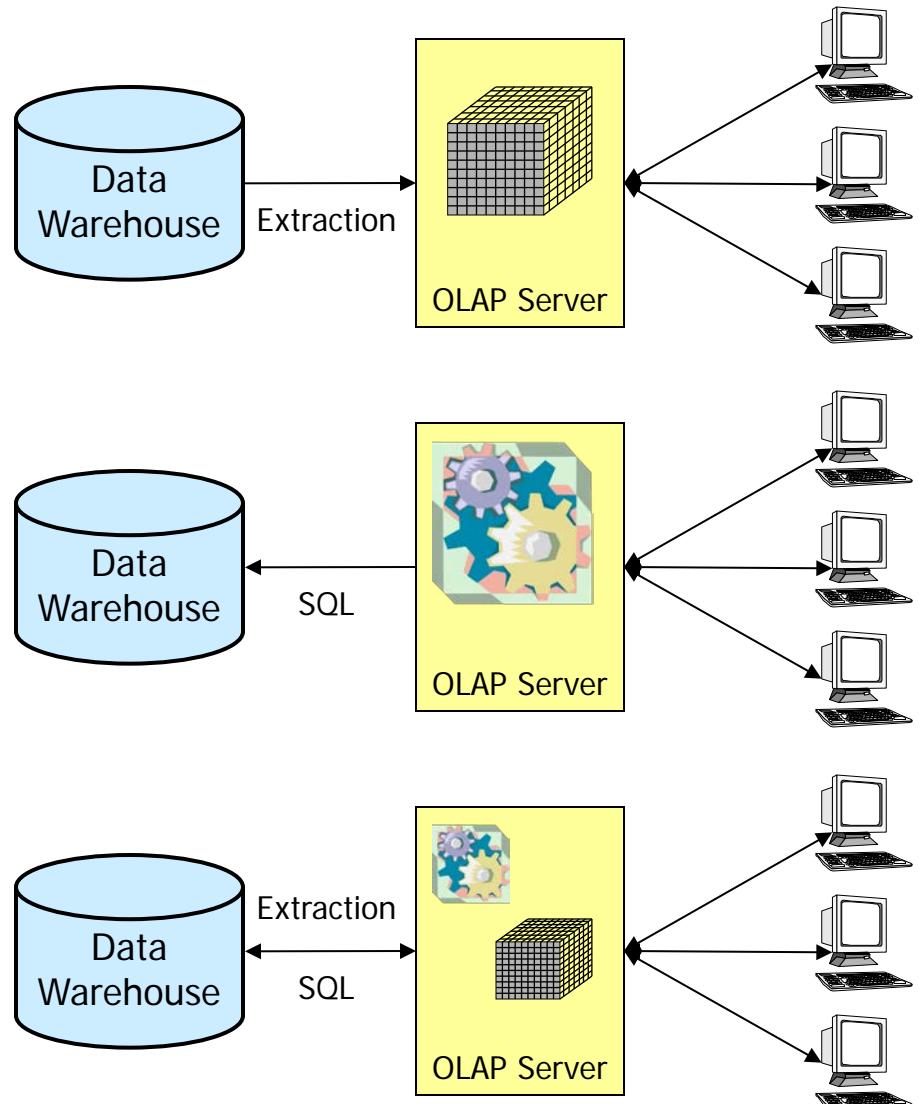
Architecture

- Different options based on:
 - storage of OLAP data
 - processing of OLAP data



MOLAP, ROLAP, HOLAP

- MOLAP
 - data resides in a multidimensional DBMS
 - multidimensional engine (OLAP server) provides access
- ROLAP
 - data resides in a relational DBMS
 - OLAP server provides SQL queries
- HOLAP
 - detailed data resides in a relational DBMS
 - aggregated data resides in a multidimensional DBMS



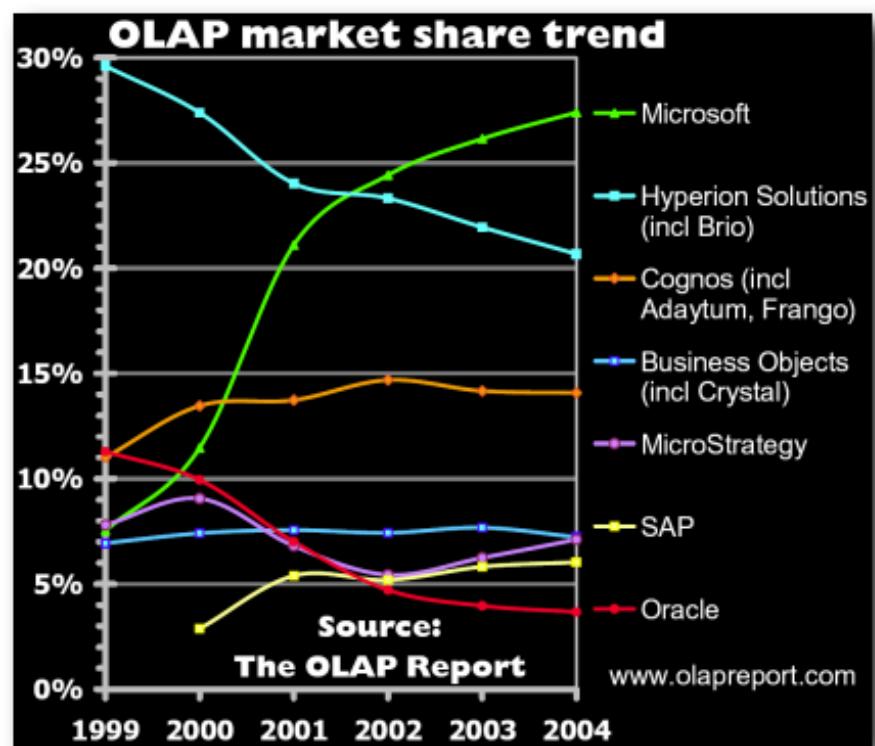
Architecture: Comparison

	MOLAP	ROLAP	HOLAP
Pros	<ul style="list-style-type: none">• short response time• efficient storage structure	<ul style="list-style-type: none">• mature relational technology• no limits on volumes of data	<ul style="list-style-type: none">• short response time for aggregated data• efficient storage structure for aggregated data• no limits on volumes of data
Cons	<ul style="list-style-type: none">• limited performance for large volumes of data• large volumes of data on OLAP server (detailed and aggregated)• preprocessing to provide OLAP cubes	<ul style="list-style-type: none">• increased response time	<ul style="list-style-type: none">• increased response time for detailed data• administration

Product Overview

Vendor	Product	Market Coverage in 2004
Microsoft	SQL Server 2000 Analysis Services	27,4%
Hyperion Solutions	Hyperion Essbase, Hyperion Intelligence (former BRIO)	20,7%
Cognos	PowerPlay, Planning (former Adaytum Planning)	14,1%
Business Objects	BusinessObjects and Webintelligence	7,2%
MicroStrategy	MicroStrategy7i	7,1%
SAP	SAP Business Information Warehouse	6,0%
Oracle	Oracle Express, Oracle10g OLAP Option	3,7%
Applix	Applix TM1	3,1%
Cartesis	Cartesis Magnitude	3,1%

Market Overview



Summary

- OLAP: Technologies and tools that support (ad-hoc) analysis of multi-dimensionally aggregated data
- Basic Operations:
 - Slice and Dice, Roll-up and Drill-down, Pivot
- Main characteristics of OLAP:
 - Fast, Analysis, Shared, Multidimensional, Information
- Storage options:
 - relational database system
 - multidimensional db (n-dimensional arrays, m-dim. query language)
- Architectural options:
 - ROLAP, MOLAP, HOLAP

Papers

- [CCS93] E. Codd, S. Codd, C. Salley: Providing OLAP (On-Line Analytical Processing) to User Analysts: An IT Mandate. White Paper, Arbor Software Cooperation, 1993.

Data-Warehouse-, Data-Mining- und OLAP-Technologien

Chapter 6: Data Mining

Bernhard Mitschang, Christoph Gröger
Universität Stuttgart

Winter Term 2015/2016

Overview

- Introduction
 - Terms & Definitions
 - Disciplines & Applications
- Data Mining Techniques
 - Overview
 - Association Rule Discovery
 - Clustering
 - Classification
 - Regression
- Data Mining Systems
 - Tools
 - Trends

Motivation

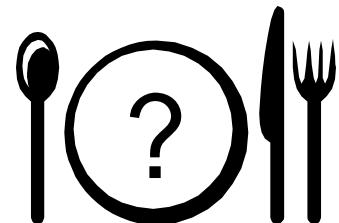
- Automated data collection tools, mature database technology and Internet lead to tremendous amounts of data stored in databases and other information repositories.



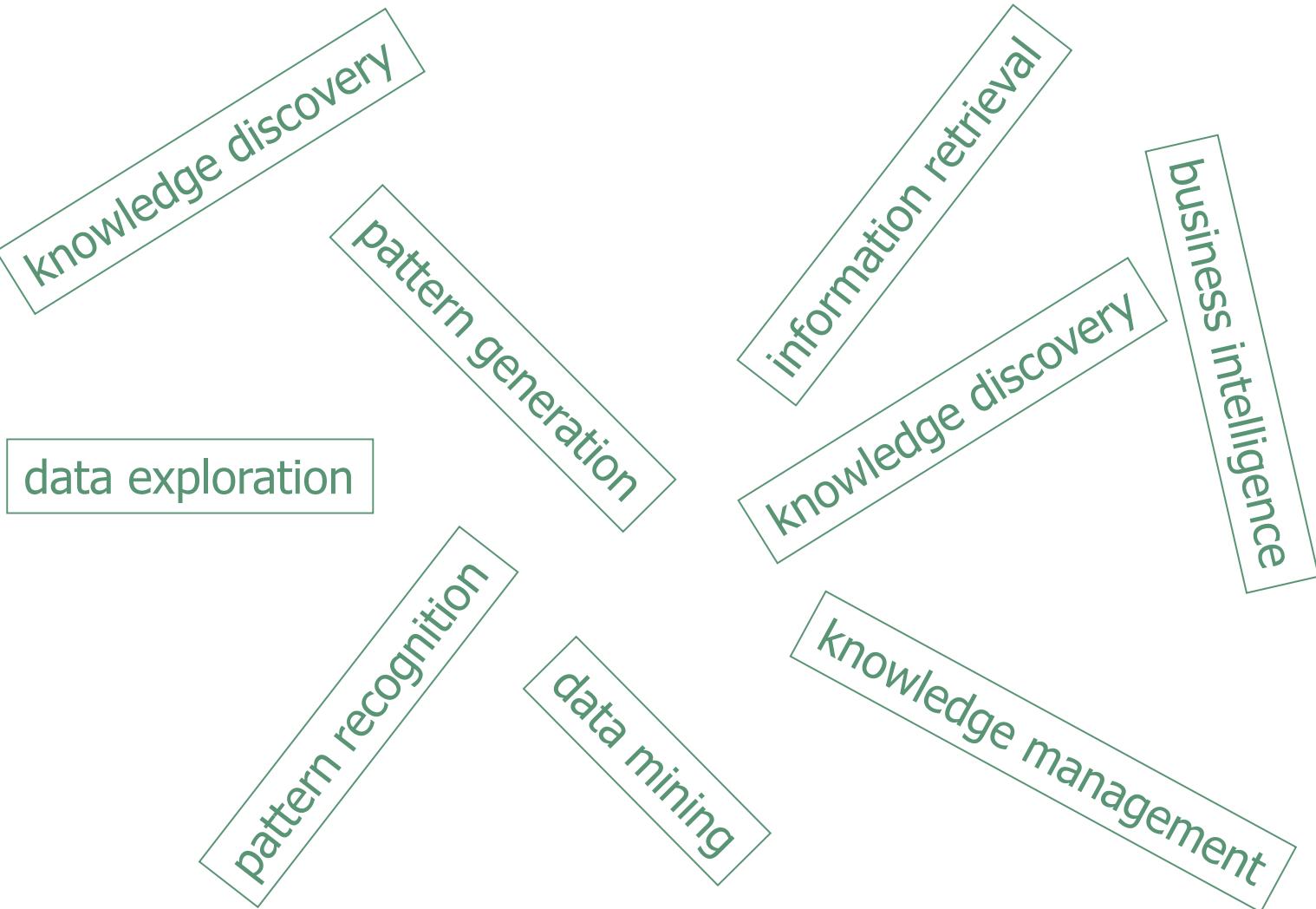
- *We are drowning in information, but starving for knowledge! [John Naisbett]*



- Business Intelligence:
 - OLAP: On-line Analytical Processing
 - Data Mining: Extraction of interesting knowledge (rules, regularities, patterns, constraints) from large data sets.



Terms & Explanations



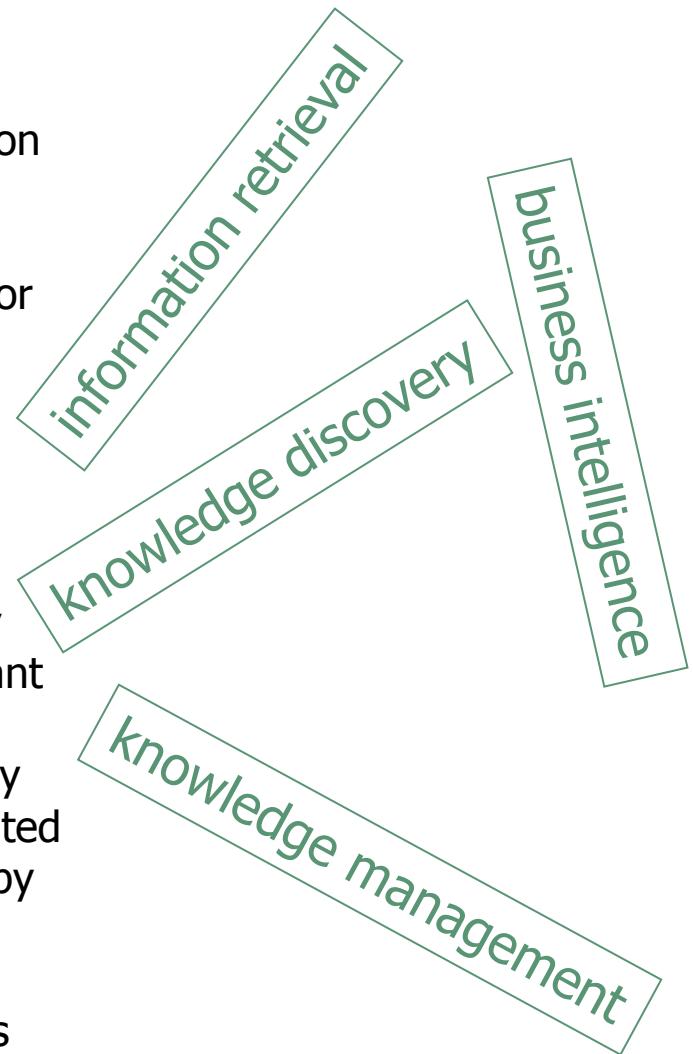
Terms & Explanations



- Here's what's behind these terms:
 - **Knowledge discovery** is a process for the search and generation of knowledge from large data sets. Sometimes the equivalent term **data exploration** is used.
 - One phase of the **knowledge discovery** process, called **pattern generation**, generates relevant information. In our case, this phase is synonymous to **data mining**. However, this phase can also be represented by e.g. on-line analytical processing (OLAP).
 - The term **pattern recognition** is more frequently used in other fields of research, like AI (image understanding) and natural science, e.g. for biochemistry or geographic (GIS) information systems. However, in principle, pattern recognition is equivalent to **knowledge discovery**.

Terms & Explanations

- Here's what's behind these terms:
 - **Information retrieval** is equivalent to data exploration but is primarily used when text-oriented data is explored by search engines. Some authors and companies call this text mining. The user's query for relevant documents are often vague.
 - **Business intelligence** describes the application of **knowledge discovery** in companies/commercial organizations to create economic benefit (\$).
 - **Knowledge management** comprises, among other things, **knowledge discovery**. However, the primary aim is storage, maintenance, and retrieval of relevant information in a way that allows all members of an organization to access this information appropriately (speed, quality, cost, etc.). This approach is motivated by the fact that much valuable information is kept by individuals and is not shared by others who require this information. Even worse, this information is seldom externalized but only memorized in people's brains.



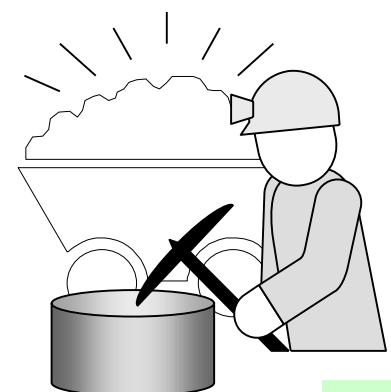
Definitions

- Data mining is the process of discovering hidden, previously unknown and usable information from a large amount of data. The data is analyzed without any expectation on the result. Data mining delivers knowledge that can be used for a better understanding of the data.

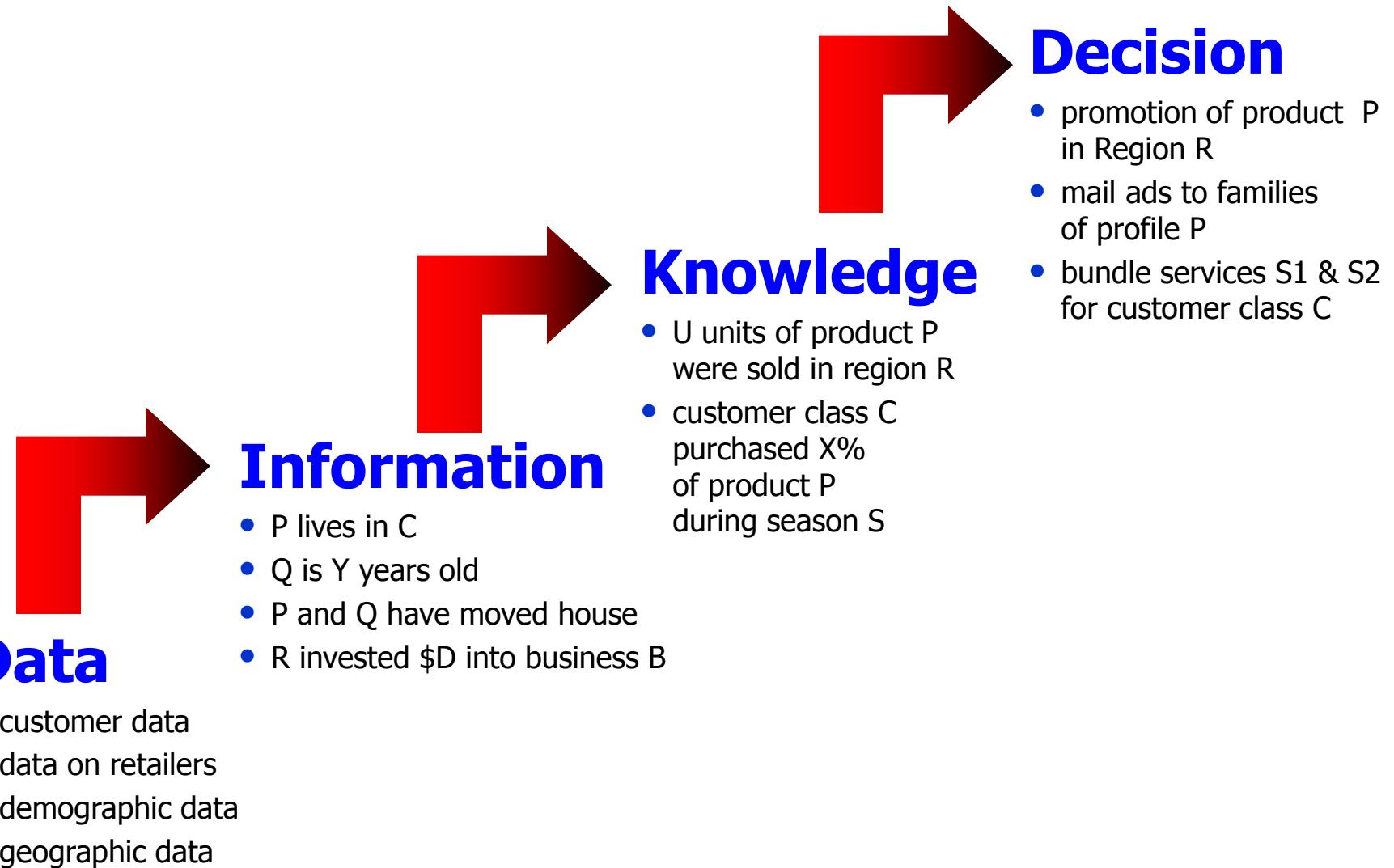
(ISO/IEC JTC1/SC32 WG4 SQL/MM Part 6 WD, 2000)

- Knowledge discovery in databases (KDD) is the non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data.

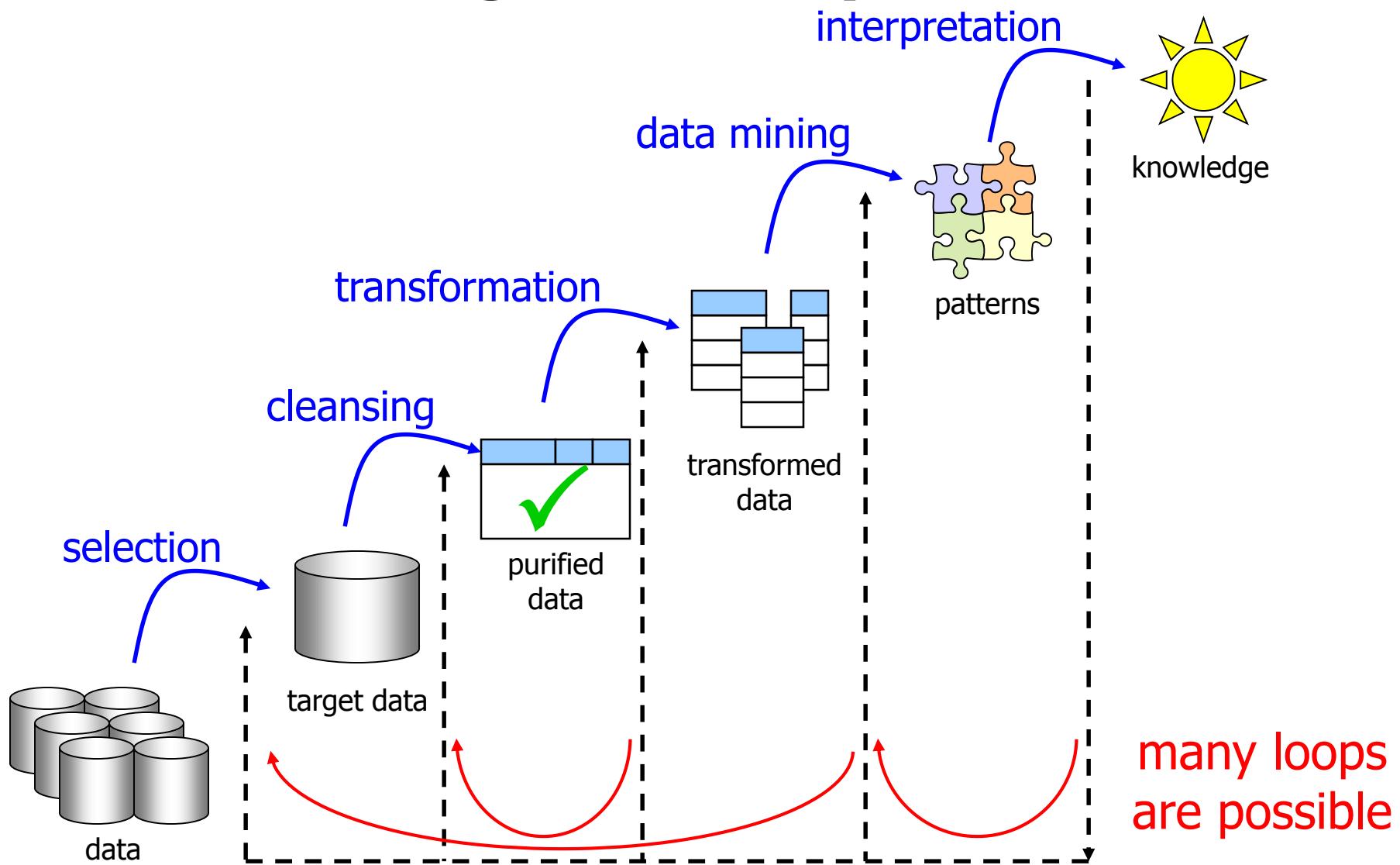
(U. Fayyad et al.: The KDD Process for Extracting ... [FP+96])



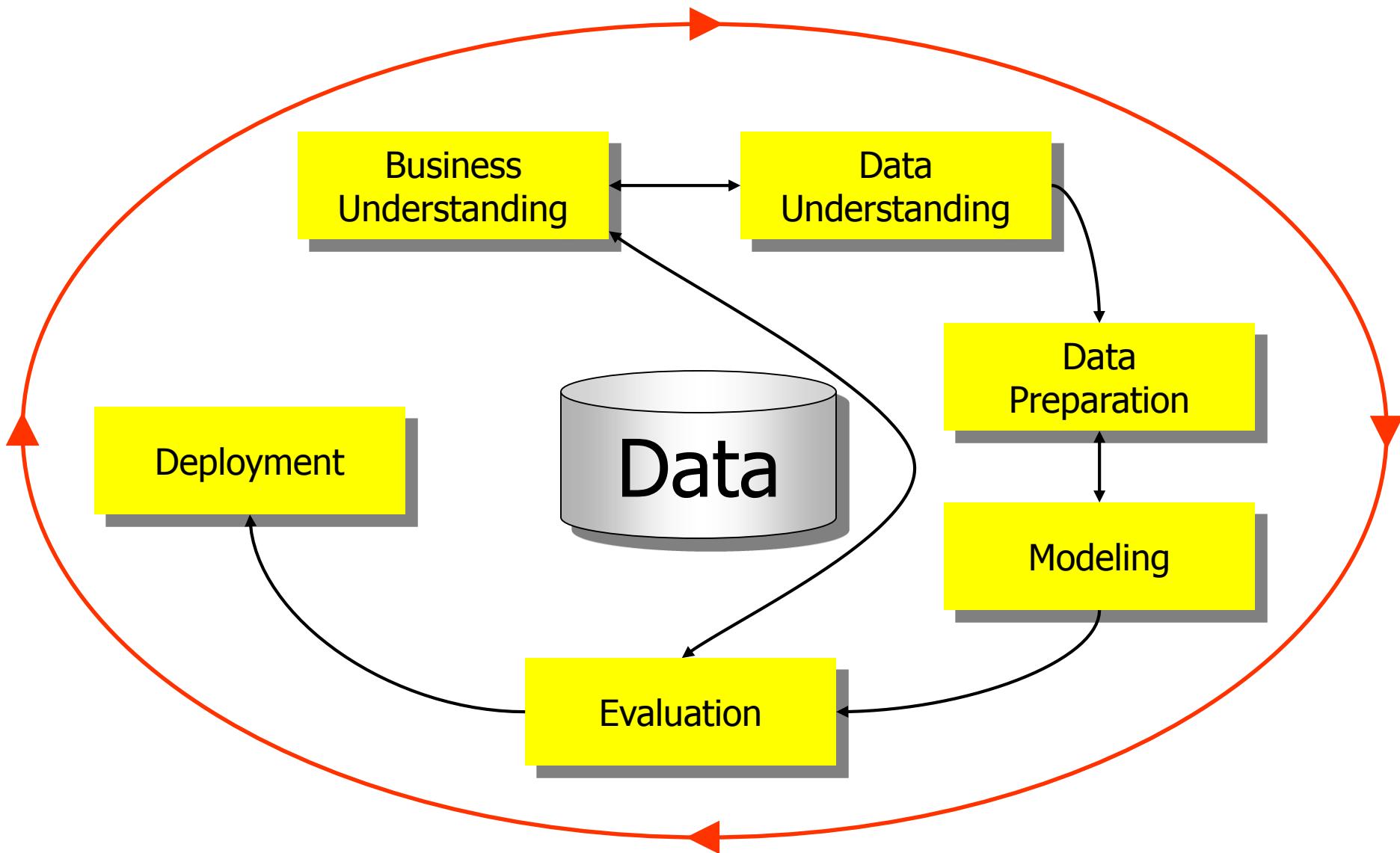
Value Chain of Information



Knowledge Discovery Process



Phases of the CRISP-DM Reference Model

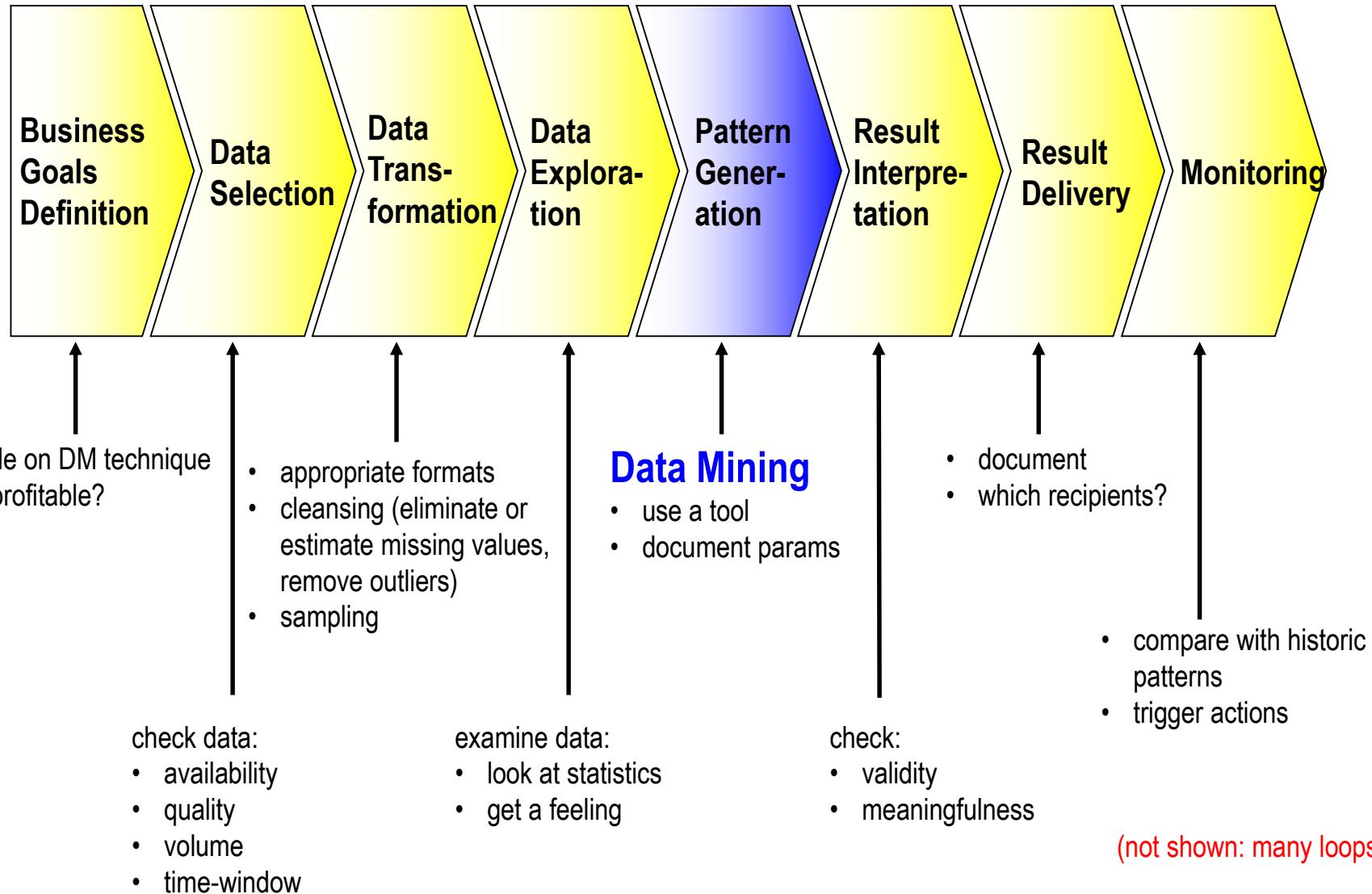


CRISP-DM Reference Model: Phases, Tasks, and Outputs

Business Understanding	Data Understanding	Data Preparation	Modeling	Evaluation	Deployment
Determine Business Objectives <ul style="list-style-type: none"> • Background • Business Objectives • Business Success Criteria Assess Situation <ul style="list-style-type: none"> • Inventory of Resources • Requirements, Assumptions, and Constraints • Risks and Contingencies • Terminology • Costs and Benefits Determine Data Mining Goals <ul style="list-style-type: none"> • Data Mining Goals • Data Mining Success Criteria Produce Project Plan <ul style="list-style-type: none"> • Project Plan • Initial Assessment of Tools and Techniques 	Collect Initial Data <ul style="list-style-type: none"> • Initial Data Collection Report Describe Data <ul style="list-style-type: none"> • Data Description Report Explore Data <ul style="list-style-type: none"> • Data Exploration Report Verify Data Quality <ul style="list-style-type: none"> • Data Quality Report 	Data Set <ul style="list-style-type: none"> • Data Set Description Select Data <ul style="list-style-type: none"> • Rationale for Inclusion/Exclusion Clean Data <ul style="list-style-type: none"> • Data Cleaning Report Construct Data <ul style="list-style-type: none"> • Derived Attributes • Generated Records Integrate Data <ul style="list-style-type: none"> • Merged Data Format Data <ul style="list-style-type: none"> • Reformatted Data 	Select Modeling Technique <ul style="list-style-type: none"> • Modeling Technique • Modeling Assumptions Generate Test Design <ul style="list-style-type: none"> • Test Design Build Model <ul style="list-style-type: none"> • Parameter Settings Model • Model Descriptions Assess Model <ul style="list-style-type: none"> • Model Assessment • Revised Parameter Settings 	Evaluate Results <ul style="list-style-type: none"> • Assessment of Data Mining Results w.r.t. Business Success Criteria • Approved Models Review Process <ul style="list-style-type: none"> • Review of Process Determine Next Steps <ul style="list-style-type: none"> • List of Possible Actions • Decisions 	Plan Deployment <ul style="list-style-type: none"> • Deployment Plan Plan Monitoring and Maintenance <ul style="list-style-type: none"> • Monitoring and Maintenance Plan Produce Final Report <ul style="list-style-type: none"> • Final Report • Final Presentation Review Project <ul style="list-style-type: none"> • Experience Documentation



Knowledge Discovery Process in Detail





Overview

- Introduction
 - Terms & Definitions
 - Disciplines & Applications
- Data Mining Techniques
 - Overview
 - Association Rule Discovery
 - Clustering
 - Classification
 - Regression
 - ...
- Data Mining Systems
 - Tools
 - Trends

Fields of Application

- Typical
 - **Marketing**: segmentation, customer targeting, etc.
 - **Finance**: investment support, portfolio management
 - **Banking & insurance**: credit and policy approval
 - **Security**: fraud detection
 - **Science & medicine**: hypothesis discovery, prediction, classification, diagnosis
 - **Manufacturing**: process modeling, quality control, resource allocation
 - **Internet**: smart search engines, web marketing
- Exotics
 - **Sports**

IBM Advanced Scout analyzed NBA match statistics between the New York Knicks and the Charlotte Hornets and identified a certain player (Rice), playing a certain position (shooting guard), shooting at a certain rate (83%), on a certain type of shot (jump shots)
 - **Astronomy**

Jet Propulsion Laboratory & Palomar Observatory discovered 22 quasars with the help of data mining

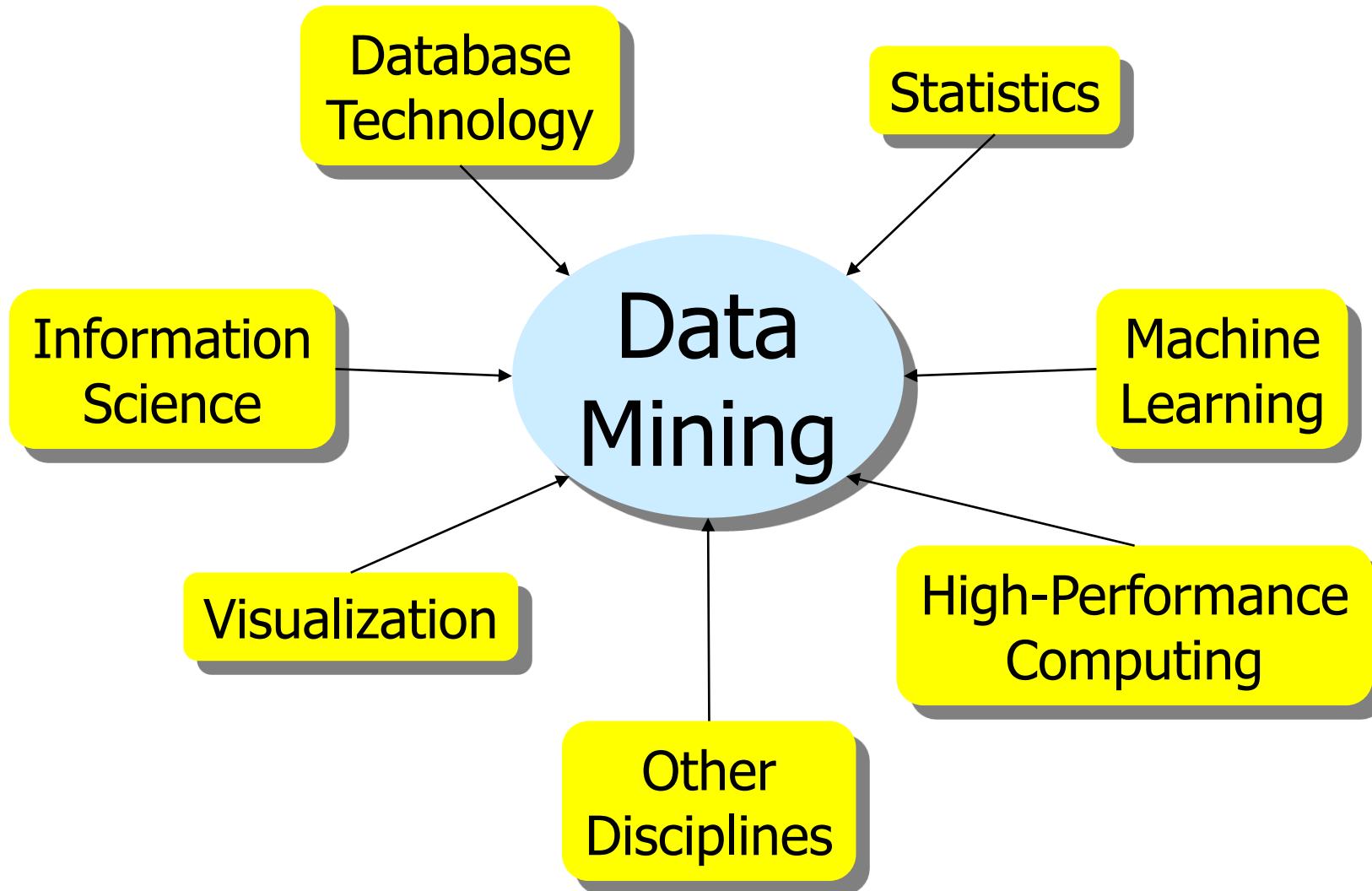
Commercial Applications

- Market analysis
 - Data sources: credit card and loyalty card transactions, discount coupons, lifestyle studies
 - Target marketing: Find customer segments with common characteristics like income, interests, and revenue.
 - Prediction: Which product characteristics contribute to attract new customers?
 - Pricing strategy: Identify customer segments and find an appropriate pricing for each segment.
 - Market basket analysis: Which products are frequently purchased together?
- Fraud detection
 - Who: banking & insurance, health care, telecommunications (phone cards)
 - Auto insurance: Discover groups of people who stage accidents to collect on insurance.
 - Medical insurance: Detect ring of doctors, ring of patients, and ring of references.

Data Mining and the ETL Process

- Data mining results can also be used in the data staging area to:
 - enrich data in the data warehouse
 - apply predictive data mining models to new data
 - e.g., assign new customers to customer segments previously identified by clustering techniques;
 - support cleansing
 - wrong or unlikely values can be detected based on data mining models
 - eventually, missing values can be deduced

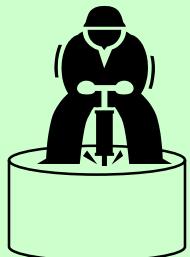
Confluence of Multiple Disciplines



Data Mining vs. Statistics

Data Mining:

- Experimental
- Adventerous attitude (inherited from machine learning)
- Algorithm is king
- (combination of) different data types (image, speech, text, etc.)
- Must contend with non-statistical issues, e.g. how to obtain (clean) data
- Two classes of tools:
 - Pattern detection (small departures from the norm): sampling bad
 - Model building (main features of the shape of the distribution): sampling ok



Statistics:

- Conservative
- Mathematical rigor
- Model is king
- Numerical data only



A central concern is how to make statements about a population when one has observed only a sample



Major Issues in Data Mining

- Mining different kinds of knowledge in databases
- Interactive mining of knowledge at multiple levels of abstraction
- Incorporation of background knowledge
- Data mining query languages and ad hoc data mining
- Presentation and visualization of data mining results
- Handling noisy or incomplete data
- Pattern evaluation - the interestingness problem
- Efficiency of data mining algorithms
- Parallel, distributed and incremental mining algorithms
- Handling of relational and complex data types
- Mining information from heterogeneous databases and global information systems

Methodology

Performance

Data Types



Overview

- Introduction
 - Terms & Definitions
 - Disciplines & Applications
- Data Mining Techniques
 - Overview
 - Association Rule Discovery
 - Clustering
 - Classification
 - Regression
 - ...
- Data Mining Systems
 - Tools
 - Trends

Data Mining Techniques

- Four data mining techniques are generally accepted
 - Association rule discovery
 - Clustering/segmentation
 - Classification
 - Regression
- Data mining techniques are either **descriptive** or **predictive**
- Many different algorithms exist that realize a specific technique
- A specific sort of technique depends heavily on the type of application, i.e., which problem has to be solved
 - Credit assessment for bank loans
 - Shopping basket analysis for new layout of a supermarket
 - Customer segmentation for customer mailings

Descriptive Techniques

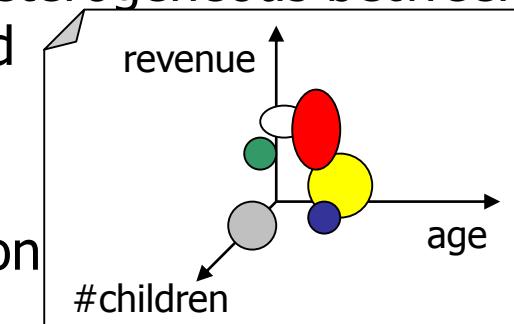
• Association rule discovery

- Given a set of purchase transactions (baskets) which contain a set of items. Find rules of the form: If a purchase transaction contains item X and item Y then the purchase transaction also contains item Z in N% of all purchase transactions.
- Example of application:
Shopping basket analysis

{beer, nappies} → {potato chips}
support = 0.04
confidence = 0.81

• Clustering/segmentation

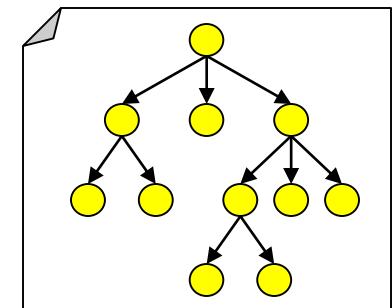
- Given a set of rows with a set of fields. Find sets of rows with common characteristics - the clusters. Rows are possibly homogeneous inside a cluster, and possibly heterogeneous between two clusters. Characterize each cluster by field values and rank the fields such that the most distinguishing fields come first.
- Example of application: customer segmentation



Predictive Techniques

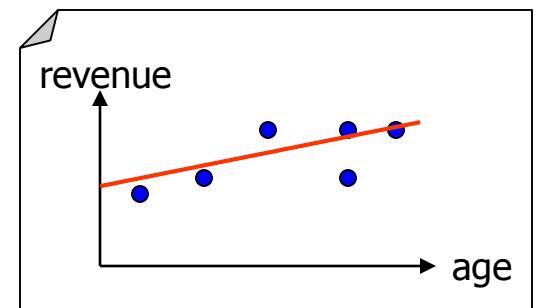
• Classification

- Given a set of rows with a set of fields and a special categorical target field, called class label. Compute a classification model such that the class label can be predicted by using the model and a set of field values without the class label. Optimize the model such that a class label can be predicted with a minimal number of field values.
- Example of application: credit assessment



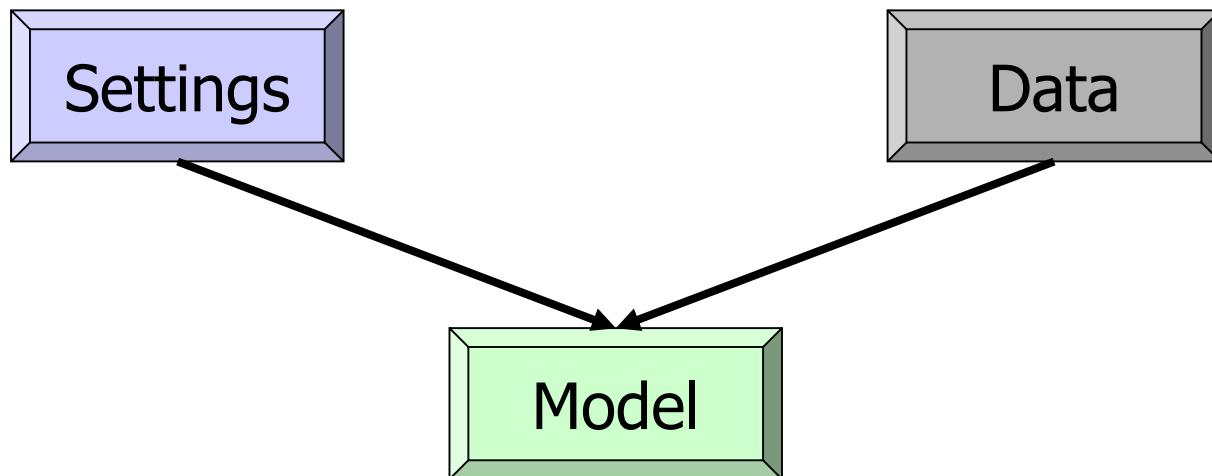
• Regression

- Regression is very similar to classification except for the type of the target field. Rather than predicting a class label, regression is predicting a numeric value. Hence, the predicted value might not be identical with any value contained in the data used to build the model.
- Example of application: customer ranking



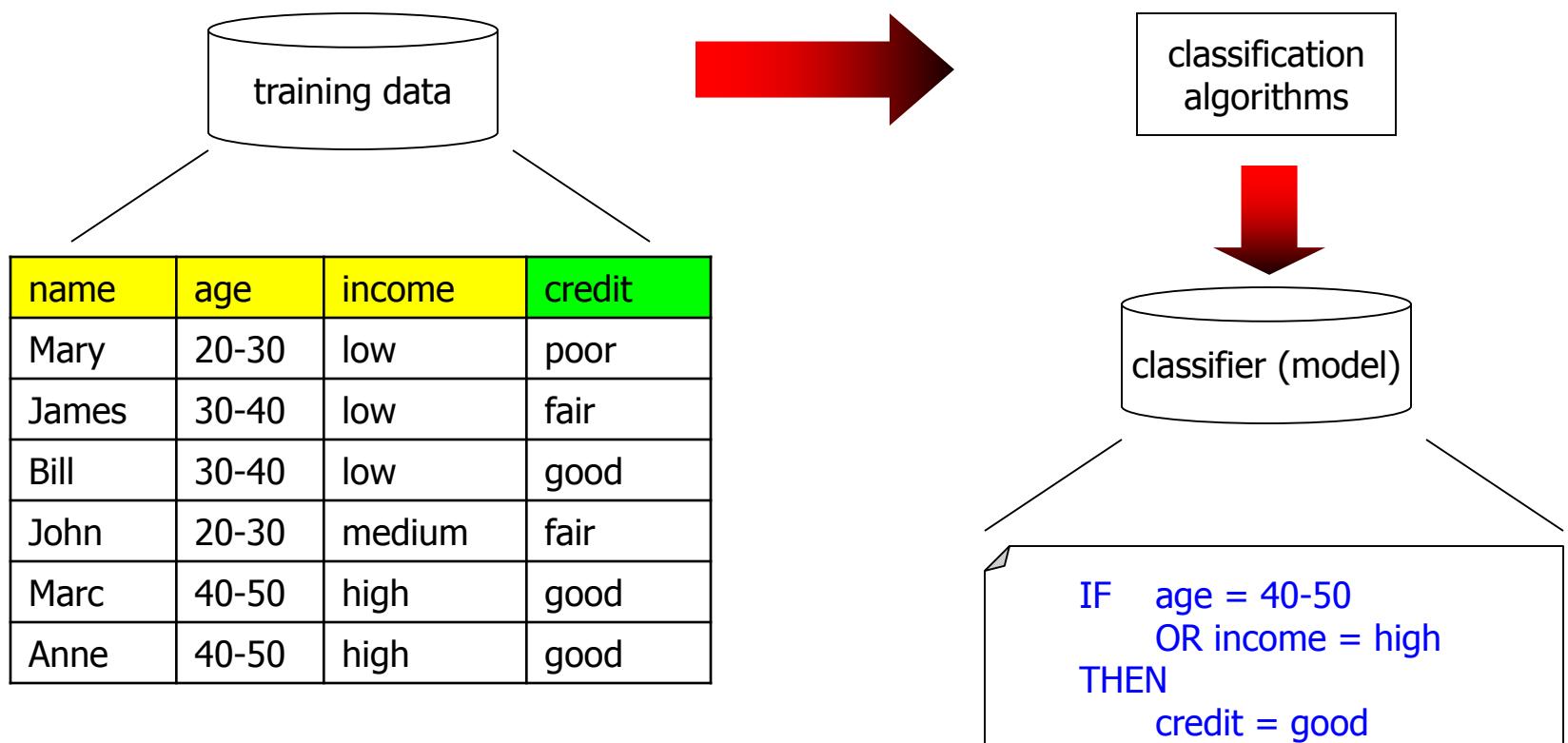
Training Phase

- Goal: Compute the data mining model.
- The training phase is common to all data mining techniques.
- Essentially, the data mining needs some settings and a set of rows to compute a data mining model.
- Setting covers, e.g.:
 - constraints like minimum support in association rule discovery
 - special handling of a field as in case of predictive techniques



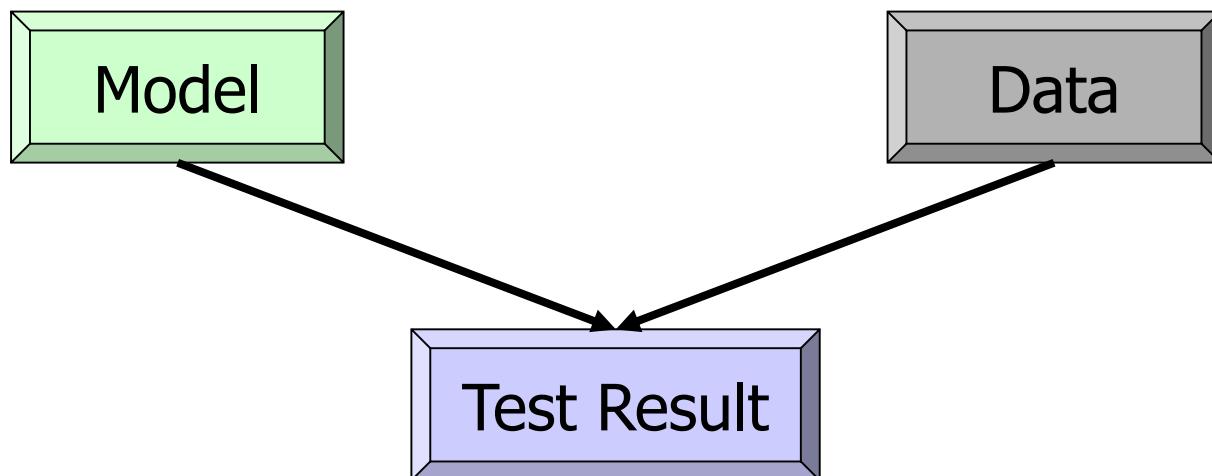
Classification: Training Phase

- **Given:** a set of training tuples carrying a class label
- **Aim:** a model (classifier) that assigns a class label to a given tuple by deriving the label from the values of the tuple's attributes

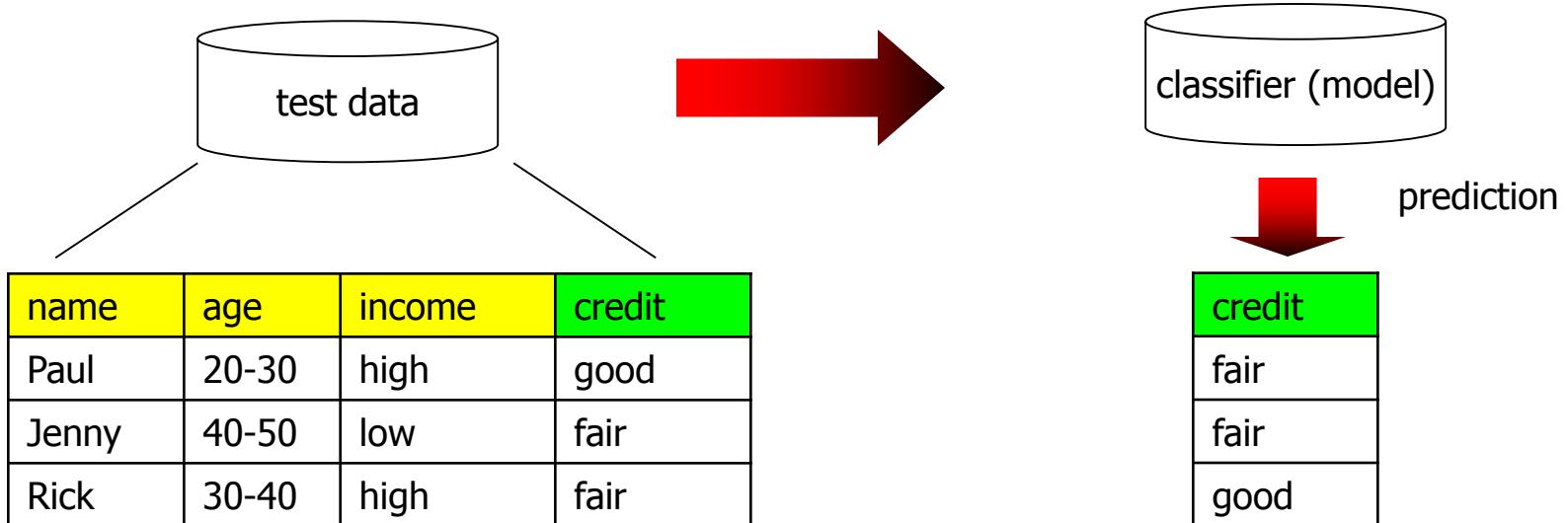


Test Phase

- Goal: Test the quality of the prediction using the data mining model.
- Set of rows with the special training field is read and for each row the application of the mining model is invoked.
- The predicted value is compared to the actual value in the special field.
- A statistical result about the false predictions is computed.



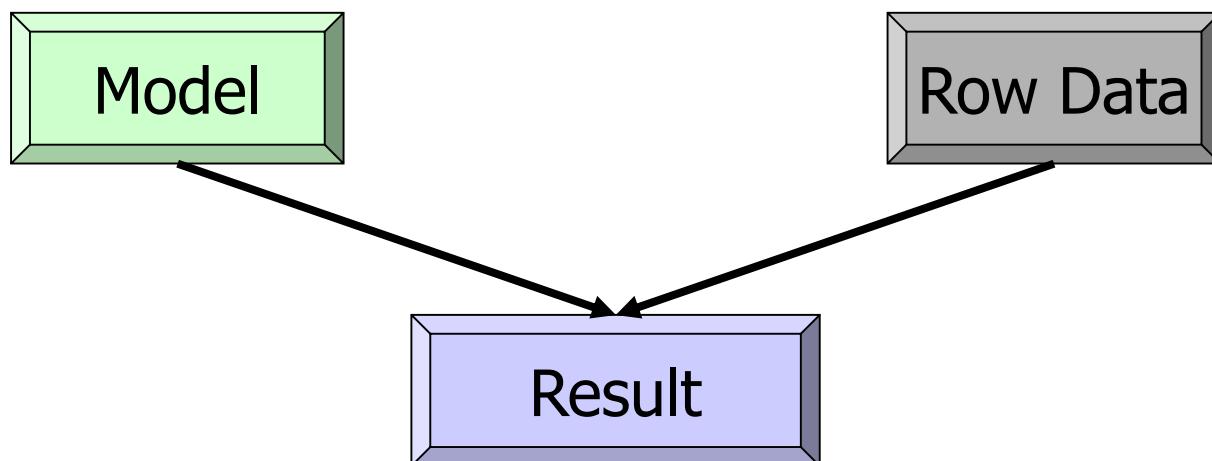
Classification: Test Phase



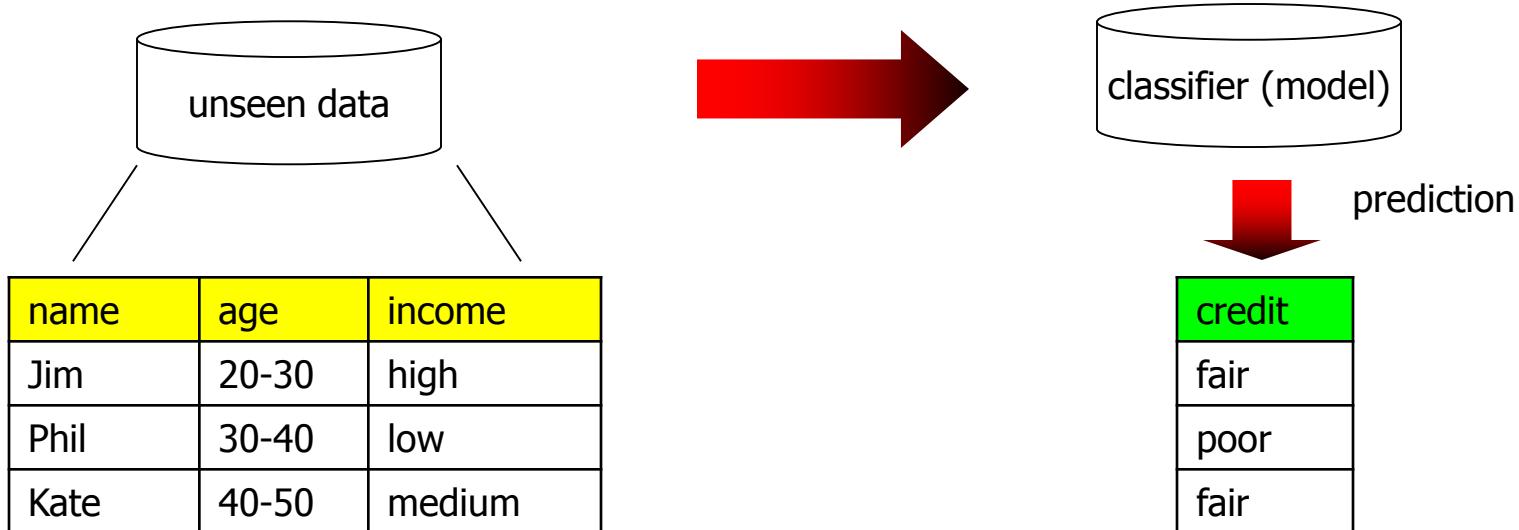
If there is a significant discrepancy between the expected (and by definition correct) result and the result predicted by the model it may be necessary to **adapt/rebuild the model**.

Application Phase

- Goal: Evaluate rows against a data mining model and compute one or more values.
- Applying a clustering model to a customer row would assign a cluster id and a confidence to the customer row.
- In case of classification a class label is computed for the row.
- To apply the model correctly to a data row, the fields of the row have to be assigned to the fields which were identified as the relevant fields during the training phase.



Classification: Application Phase



Data Mining Phases

Data mining technique	Training phase	Test phase	Application phase
Classification	✓	✓	✓
Regression	✓	✓	✓
Clustering	✓	✗	✓
Association rule discovery	✓	✗	✗

Source: ISO/IEC 13249-6:2002 Information technology -- Database languages -- SQL multimedia and application packages -- Part 6: Data mining



Overview

- Introduction
 - Terms & Definitions
 - Disciplines & Applications
- Data Mining Techniques
 - Overview
 - Association Rule Discovery
 - Clustering
 - Classification
 - Regression
 - ...
- Data Mining Systems
 - Tools
 - Trends

Association Rule Discovery (ARD)

- An **itemset** is a set of items
- Given:
 - a set of transactions (itemsets) $D = \{t_1, t_2, \dots, t_n\}$
 - a threshold minimum support s_{min} , where $0 \leq s_{min} \leq 1$
 - a threshold minimum confidence c_{min} , where $0 \leq c_{min} \leq 1$
- **Support** of an itemset X:
 $s(X) = |\{t \in D \mid X \subseteq t\}| / |D|$
- Support is also called „frequency“
- Itemset X with $s(X) \geq s_{min}$ is called **frequent**
- Rule $L \rightarrow R$:
 If L is part of a transaction t,
 then R is also part of t (L und R are itemsets)
- **Support** of a rule $L \rightarrow R$:
 $s(L,R) = s(L \cup R)$
- **Confidence** of a rule $L \rightarrow R$:
 $c(L,R) = s(L \cup R) / s(L)$

Database D

tid	item
101	beer
101	cheese
102	beer
102	milk
102	tomatoes
103	cheese

$$s_{min} = 0.2$$

$$c_{min} = 0.5$$

Association Rule Discovery (ARD)

- **Problem:**
Find all rules $L \rightarrow R$, where L and R are itemsets
and the following conditions are true:
 - $s(L \cup R) \geq s_{\min}$
 - $c(L, R) \geq c_{\min}$.
- Such rules are called **association rules**

Database D

tid	item
101	beer
101	cheese
102	beer
102	milk
102	tomatoes
103	cheese

$$S_{\min} = 0.2$$

$$C_{\min} = 0.5$$



An example association rule:

$$\{\text{beer}\} \rightarrow \{\text{milk, tomatoes}\}$$

$$\begin{aligned} \text{support} &= s(\{\text{beer, milk, tomatoes}\}) \\ &= 1/3 = 33\% \end{aligned}$$

$$\text{confidence} = 1/2 = 50\%$$

Computation of Association Rules

- 1. Frequent itemsets generation:

- compute frequent itemsets from transactions
- an itemset X is called frequent if $s(X) \geq s_{\min}$
- **Apriori property**: each subset of a frequent itemset is also frequent.
Thus, if $\{A, B\}$ is frequent then both $\{A\}$ and $\{B\}$ are frequent.
- s_{\min} low: many itemsets that occur rarely
- s_{\min} high: few itemsets that occur frequently

expensive

- 2. Association rules generation:

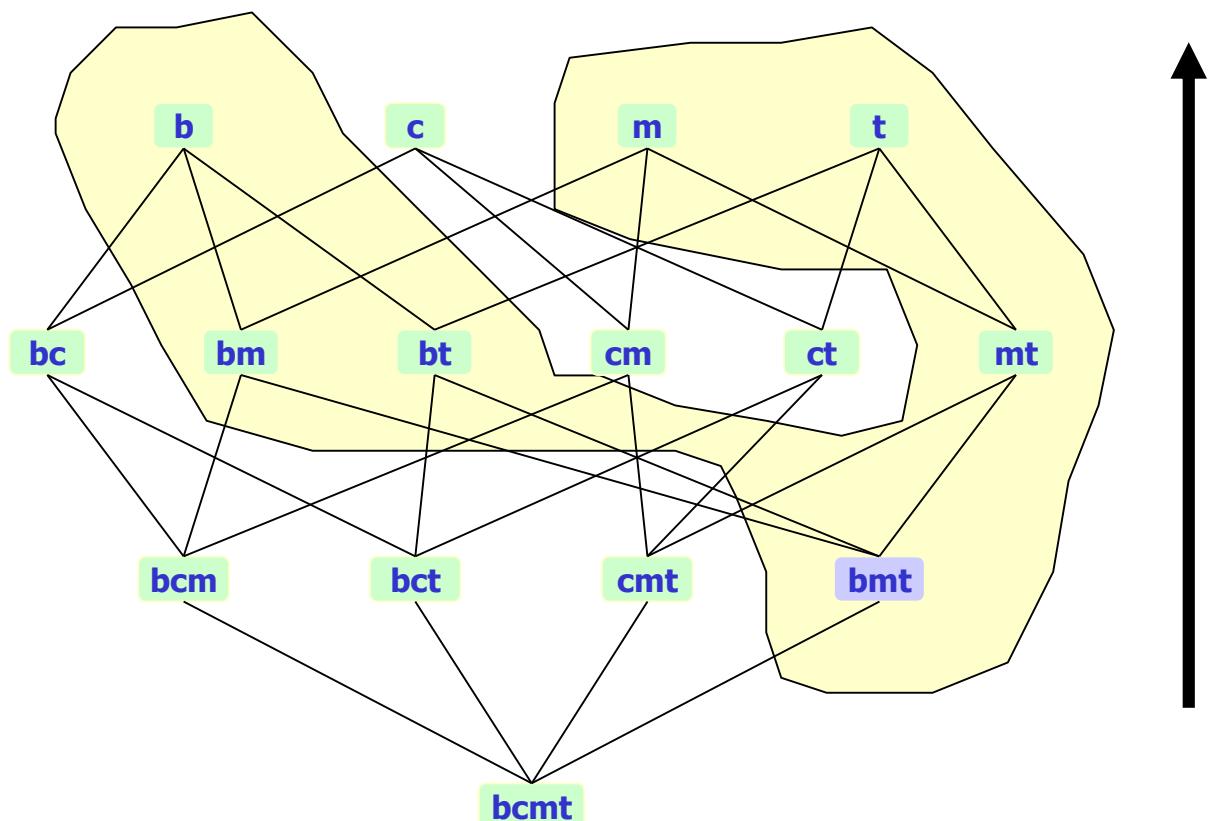
- compute association rules from frequent itemsets
- c_{\min} low: many rules where many are „insecure“
- c_{\min} high: few rules which are all (almost) logically correct
(true implications when $c_{\min} = 1$)

cheap

Apriori Property

Database D

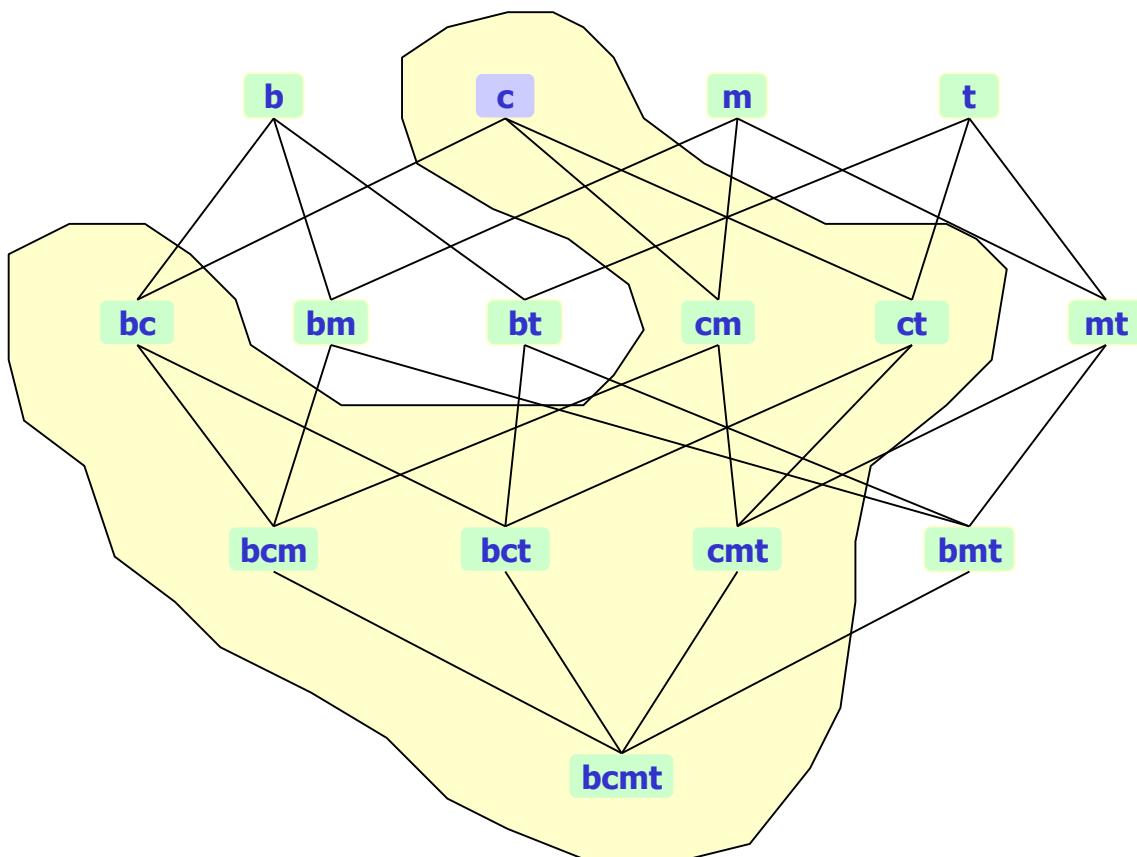
tid	item
101	beer
101	cheese
102	beer
102	milk
102	tomatoes
103	cheese



Apriori Property

Database D

tid	item
101	beer
101	cheese
102	beer
102	milk
102	tomatoes
103	cheese



Apriori Algorithm

F : result set of frequent itemsets (of different sizes)

$F[k]$: set of frequent itemsets of size k (k-itemsets)

$C[k]$: set of candidate itemsets of size k

```
SetOfItemsets generateFrequentItemsets(Integer minimumSupport) begin
    F[1] = {frequent items};
    for (k = 1; F[k] ≠ Ø; k++) do begin
        C[k+1] = generateCandidates(k, F[k]);
        for each transaction t ∈ database do
            for each candidate c ∈ C[k+1] do
                if c ⊆ t then
                    c.count++;
        for each candidate c ∈ C[k+1] do
            if c.count ≥ minimumSupport then
                F[k+1] = F[k+1] ∪ {c};
    end;
    F = Ø;
    while k ≥ 1 do begin
        F = F ∪ F[k];
        k--;
    end;
    return F;
end;
```

Apriori Algorithm: Candidate Itemsets Generation

C: result set of candidate itemsets of size k+1

c: candidate itemset of size k+1

```
SetOfItemsets generateCandidates (Integer k; SetOfItemsets F) begin
    C = Ø;
    for each k-itemset f ∈ F do
        for each k-itemset g ∈ F do begin
            i = 1;
            while (i < k) and (f[i] = g[i]) do begin
                c[i] = f[i];
                i++;
            end;
            if (i = k) and (f[k] < g[k]) then begin
                c[k] = f[k];
                c[k+1] = g[k];
                if not hasInfrequentSubset(c, F) then
                    // only those cand's are kept where all (k-1)-subsets are frequent
                    C = C ∪ {c};
            end;
        end;
    return C;
end;
```

```
Boolean hasInfrequentSubset (Itemset c; SetOfItemsets F) begin
    for each (k-1)-subset s of c do
        if s ∉ F then
            return true;
    return false;
end;
```

Apriori Algorithm: Example (1)

Database D

tid	itemset
1	{1, 2, 3}
2	{3, 4, 5}
3	{1, 4}
4	{2, 3, 4, 5}
5	{1, 3}
6	{1, 3, 4}

C[1]

itemset	support
{1}	4
{2}	2
{3}	5
{4}	4
{5}	2

 $s_{\min} = 1/3$

F[1]

itemset	support
{1}	4
{2}	2
{3}	5
{4}	4
{5}	2

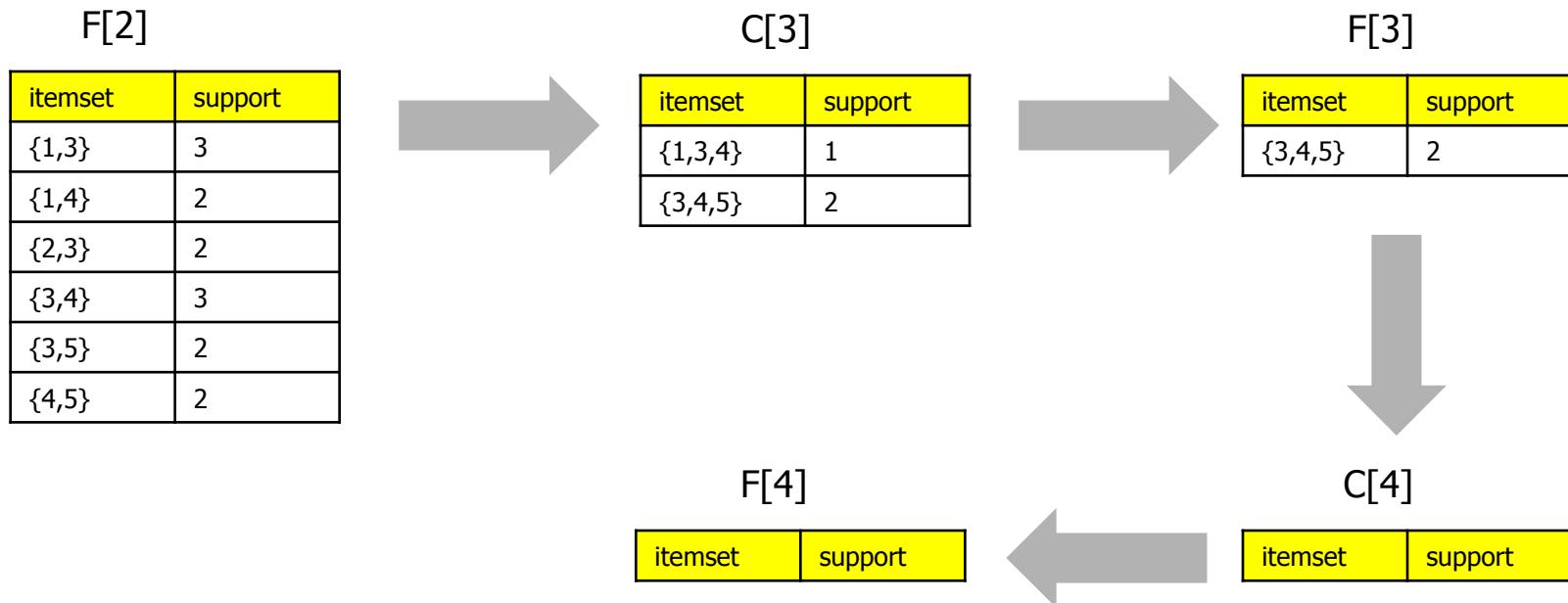
F[2]

itemset	support
{1,3}	3
{1,4}	2
{2,3}	2
{3,4}	3
{3,5}	2
{4,5}	2

C[2]

itemset	support
{1,2}	1
{1,3}	3
{1,4}	2
{1,5}	0
{2,3}	2
{2,4}	1
{2,5}	1
{3,4}	3
{3,5}	2
{4,5}	2

Apriori Algorithm: Example (2)



Database D

tid	itemset
1	{1, 2, 3}
2	{3, 4, 5}
3	{1, 4}
4	{2, 3, 4, 5}
5	{1, 3}
6	{1, 3, 4}

$C[4] = \emptyset \Rightarrow F[4] = \emptyset \Rightarrow$ stop algorithm!
 The frequent itemsets are {
 {1}, {2}, {3}, {4}, {5},
 {1,3}, {1,4}, {2,3}, {3,4}, {3,5}, {4,5},
 {3,4,5}}.

Apriori Algorithm: Candidate Itemsets Generation with SQL

$C_{k+1} (a_1, a_2, \dots, a_{k+1})$: result table containing candidate itemsets of size $k+1$

$F_k (a_1, a_2, \dots, a_k)$: table containing frequent itemsets of size k

```
INSERT INTO Ck+1
SELECT x1.a1, x1.a2, ..., x1.ak, x2.ak
FROM Fk AS x1, Fk AS x2
WHERE
-- match the itemset prefixes of size k-1
    x1.a1 = x2.a1 AND
    x1.a2 = x2.a2 AND
    ...
    x1.ak-1 = x2.ak-1 AND
-- avoid duplicates
    x1.ak < x2.ak;
```

Note that **we still need the prune step** (`hasInfrequentSubset(c, F)`) after this candidate generation. The next slide shows how this prune step can be added to this statement.

Apriori Algorithm: Candidate Itemsets Generation with SQL including the Prune Step

```
INSERT INTO Ck+1
SELECT x1.a1, x1.a2, ..., x1.ak, x2.ak
FROM   Fk AS x1, Fk AS x2, Fk AS x3, ..., Fk AS xk+1
WHERE
-- match the itemset prefixes of size k-1
    x1.a1 = x2.a1 AND
    x1.a2 = x2.a2 AND
    ...
    x1.ak-1 = x2.ak-1 AND
-- avoid duplicates
    x1.ak < x2.ak AND
-- PRUNE STEP: check if all k-subsets are frequent
-- skip item 1
    x1.a2 = x3.a1 AND
    x1.a3 = x3.a2 AND
    x1.a4 = x3.a3 AND
    ...
    x1.ak-1 = x3.ak-2 AND
    x1.ak = x3.ak-1 AND
    x2.ak = x3.ak AND
-- skip item 2
    ...
-- skip item k-1
    x1.a1 = xk+1.a1 AND
    x1.a2 = xk+1.a2 AND
    x1.a3 = xk+1.a3 AND
    ...
    x1.ak-2 = xk+1.ak-2 AND
    x1.ak = xk+1.ak-1 AND
    x2.ak = xk+1.ak;
```

Algorithm to Compute Association Rules

- Precondition: all frequent itemsets have been computed
- each itemset has two attributes:
 - count: „support“ as an integer
 - size: cardinality, i.e. number of items in itemset

```
for each frequent k-itemset f ∈ F where k ≥ 2 do
    generateRules(f, f);
    // Computes all rules (g --> f\g) for all g ⊂ f.
    generateRules(Itemset f, Itemset left) begin
        A = {a ⊂ left | a is of size:(left.size-1) };
        for each a ∈ A do begin
            confidence = f.count / a.count;
            if confidence ≥ minConfidence then begin
                rules.add(a, f\a);           // set of result association rules
                if (left.size-1 > 1) then
                    generateRules(f, a);
            end;
        end;
    end;
```

Algorithm to Compute Association Rules

- The algorithm we have seen exploits the following property:
 - Given itemsets A, B, and C.
If $B \rightarrow A \setminus B$ does not hold, neither does $C \rightarrow A \setminus C$ for any $C \subset B$.
 - If $\{a,b,c\} \rightarrow \{d\}$ has not enough confidence, the algorithm does not call `generateRules({a,b,c,d}, {a,b,c})`.
 - Note, that the confidence of a rule $L \rightarrow R$ is defined as
 $c(L,R) = s(L \cup R) / s(L)$.

AR Generation: Example

itemset	count
{3}	5
{4}	4
{5}	2
{3,4}	3
{3,5}	2
{4,5}	2
{3,4,5}	2

We restrict ourselves to itemset {3,4,5}.

Minimum confidence = $1/2 = 50\%$.

generateRules({3,4,5}, {3,4,5}):

A = {{3,4}, {3,5}, {4,5}}.

a = {3,4}: c(a) = 2/3 $\geq 1/2 \rightarrow \{3,4\} \rightarrow \{5\}$ is AR

\rightarrow generateRules({3,4,5}, {3,4}):

A = {{3}, {4}}.

a = {3}: c(a) = 2/5 < 1/2.

a = {4}: c(a) = 2/4 $\geq 1/2 \rightarrow \{4\} \rightarrow \{3,5\}$ is AR

a = {3,5}: c(a) = 2/2 $\geq 1/2 \rightarrow \{3,5\} \rightarrow \{4\}$ is AR

\rightarrow generateRules({3,4,5}, {3,5}):

A = {{3}, {5}}.

a = {5}: c(a) = 2/2 $\geq 1/2 \rightarrow \{5\} \rightarrow \{3,4\}$ is AR

... (as before)

a = {4,5}: c(a) = 2/2 $\geq 1/2 \rightarrow \{4,5\} \rightarrow \{3\}$ is AR

\rightarrow generateRules({3,4,5}, {4,5}):

A = {{4}, {5}}.

... (as before)



Thus, the following association rules can be derived from frequent itemset {3,4,5}:

{4} \rightarrow {3,5}

{5} \rightarrow {3,4}

{3,4} \rightarrow {5}

{3,5} \rightarrow {4}

{4,5} \rightarrow {3}

Association Rule Generation

- A faster algorithm than the previous one exploits a property that can be derived from the previous one:
 - For a rule $B \rightarrow A \setminus B$ to hold, all rules of the form $C \rightarrow A \setminus C$ must also hold, where $B \subset C$.
 - Note, that the confidence of a rule $L \rightarrow R$ is defined as $c(L,R) = s(L \cup R) / s(L)$.
 - The algorithm makes use of the `generateCandidates` that is used in the Apriori algorithm.
 - We don't show this algorithm here.

Types of Association Rules

simple rules

- Boolean AR:
 - rule concerns associations between the presence or absence of items
 - $\{\text{beer}\} \rightarrow \{\text{milk, tomatoes}\}$
- single-dimensional AR:
 - items or attribute in a rule reference only one dimension
 - $\{\text{purchase}=\{\text{beer}\}\} \rightarrow \{\text{purchase}=\{\text{milk, tomatoes}\}\}$

complex rules

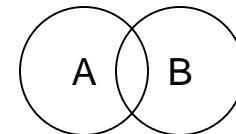
- quantitative AR:
 - rule concerns associations between quantitative items or attributes
 - $\{17 \leq \text{age} \leq 22, \text{hair}=\text{"green"}\} \rightarrow \{10k \leq \text{income} \leq 20k\}$
- multidimensional AR:
 - rule references several dimensions
 - $\{\text{age}=50, \text{income}=100k, \text{hair}=\text{"gray"}\} \rightarrow \{\text{married}=\text{"yes"}\}$
- constrained AR

Constraint-based Association Rules

- Data constraints
 - Set of task-relevant data is specified.
- Constraints on dimensions or dimension levels
 - Dimensions or levels of the concept hierarchies to be used for ARD are specified.
- Interestingness constraints
 - Thresholds on statistical measures on rule interestingness are specified.
- Rule constraints
 - Specify form of rules to be discovered, e.g. by metarules and rule templates.

Beyond Support & Confidence: Interest

- One can define further measures to find „interesting“ AR
- Two events A and B are independent if the predicate $p(A \wedge B) = p(A) \times p(B)$ holds, otherwise they are correlated
- Interest of itemsets {A, B}:
$$s(\{A, B\}) / (s(\{A\}) \times s(\{B\}))$$
- It is a measure of correlation between events
 - interest = 1: A and B are independent events
 - interest < 1: A and B are negatively correlated
 - interest > 1: A and B are positively correlated
- Sometimes, interest is also called “surprise factor” or “lift factor”



	coffee	not coffee	Σ
tea	20	5	25
not tea	70	5	75
Σ	90	10	100

example: AR {tea} → {coffee}
support = 20/100 = 0.2
confidence = (20/100) / (25/100) = 0.8
interest = $0.2 / (0.25 \times 0.9) = 0.89$,
i.e. interest(drink tea \wedge drink coffee) is negatively correlated, i.e. it is rather unlikely that you start drinking coffee if you are already a tea worshipper

Improving the Efficiency of Apriori

- Major goals for improving the frequent itemset discovery phase:
 - reducing the number of passes over the database of transactions
 - reducing the number of candidate itemsets to be generated
 - reducing the size of the database of transactions
 - exploiting parallel processing
- Approaches:
 - Hash-based techniques, e.g. direct hashing and pruning (DHP)
 - Transaction reduction
 - Sampling
 - Dynamic itemset counting
 - Partitioning
 - Parallel FID

Improving the Efficiency of Apriori

- **Transaction reduction:**
 - Transactions that do not contain any frequent k-itemsets are removed from further consideration in subsequent scans of the database because they cannot contain any frequent (k+1)-itemsets.
- **Sampling:**
 - Determine frequent itemsets L_S on a random sample S of the given data D . By a second pass on D the actual frequencies of all itemsets in L_S are determined.
- **Partitioning:**
 - Phase I: Subdivide the transactions into non-overlapping partitions. Derive local frequent itemsets for each partition (1st scan).
 - Phase II: Candidate itemsets: All itemsets that are frequent in at least one partition. Determine global frequent itemsets (2nd scan).

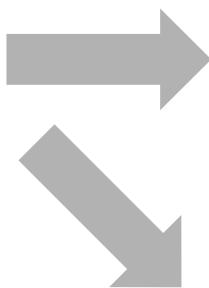
Direct Hashing and Pruning (DHP)

- Basic algorithm:
 - While frequent k-itemsets are generated from C_k ,
 $(k+1)$ -itemsets are generated from each transaction.
 - The $k+1$ -itemsets are mapped into buckets of a hash table and bucket counts are increased.
 - Bucket counts below the support threshold allow to exclude $(k+1)$ -candidate itemsets.
- Advantages:
 - Number of candidate 2-itemsets is reduced by orders of magnitude.

DHP: Example

Database D

tid	itemset
100	{A,C,D}
200	{B,C,E}
300	{A,B,C,E}
400	{B,E}

 $C[1]$

itemset	support
{A}	2
{B}	3
{C}	3
{D}	1
{E}	3

$$s_{\min} = 1/2$$

 $F[1]$

itemset	support
{A}	2
{B}	3
{C}	3
{E}	3

0	1	2	3	4	5	6	hash address
3	1	2	0	3	1	3	count
{C,E}	{A,E}	{B,C}		{B,E}	{A,B}	{A,C}	itemsets hashed to this bucket
{A,D}						{C,D}	

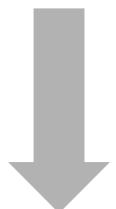
$$h(\{x,y\})=((\text{order of } x)*10+(\text{order of } y)) \bmod 7$$

 $C[2]$

itemset
{A,C}
{B,C}
{B,E}
{C,E}

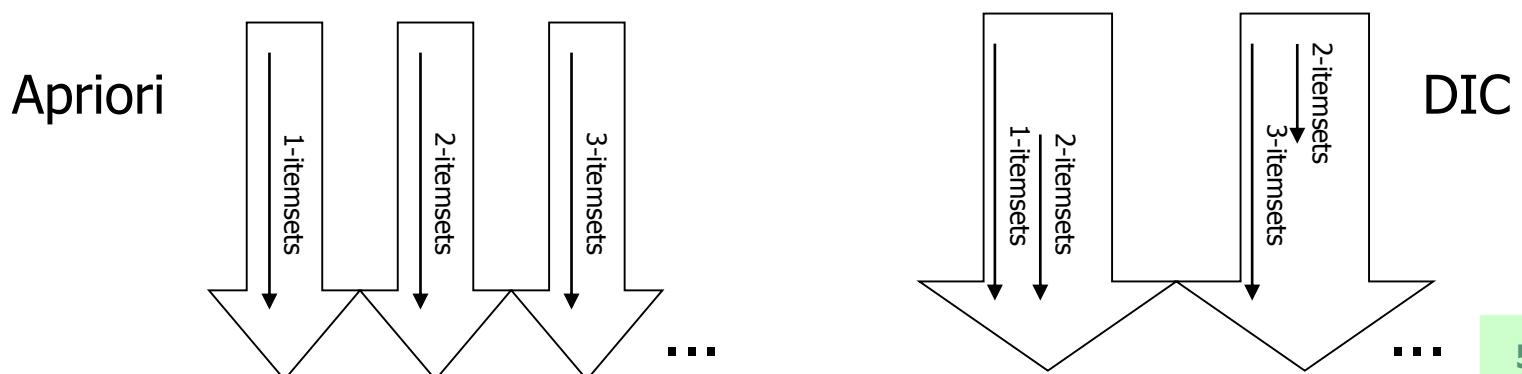
 $C[2]'$

itemset	support in bucket
{A,B}	1
{A,C}	3
{A,E}	1
{B,C}	2
{B,E}	3
{C,E}	3



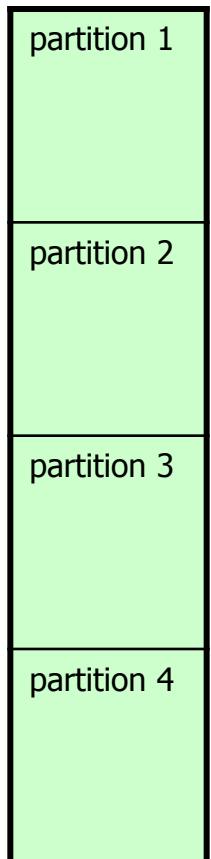
Dynamic Itemset Counting (DIC)

- Basic algorithm:
 - Database is partitioned into several blocks of M transactions.
 - At the end of each block, the support of itemsets encountered so far is estimated.
 - A new candidate itemset is generated if all its subsets are estimated to be frequent.
 - Algorithm needs a data structure that allows to store itemsets of various sizes.
- Advantages:
 - Number of passes over the database of transactions is reduced.

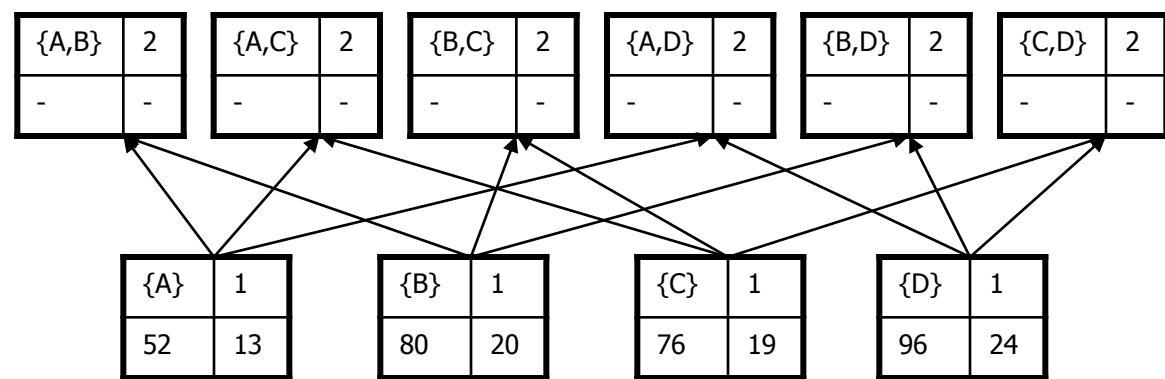


DIC: Example

Database D
of size 100

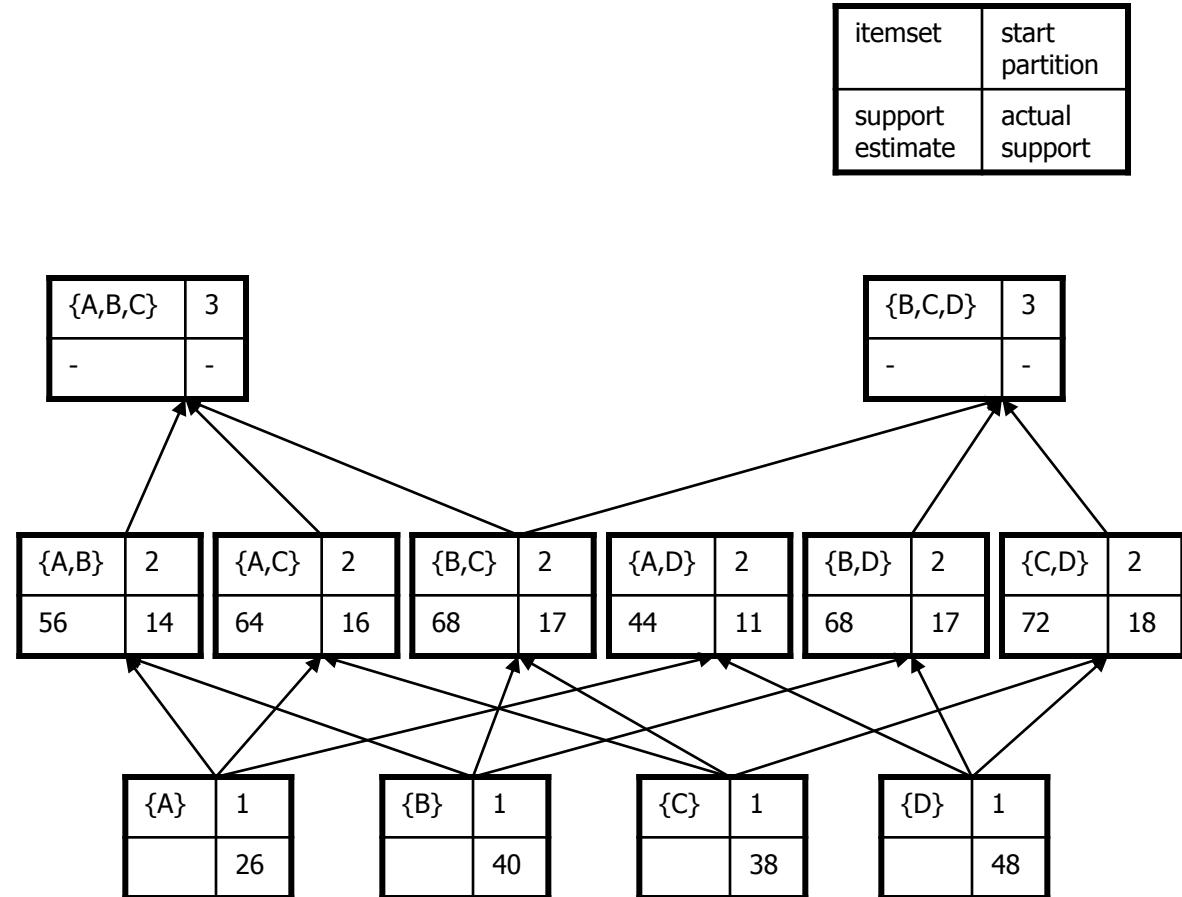
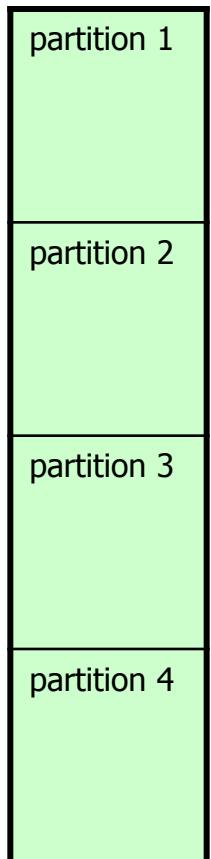


itemset	start partition
support estimate	actual support



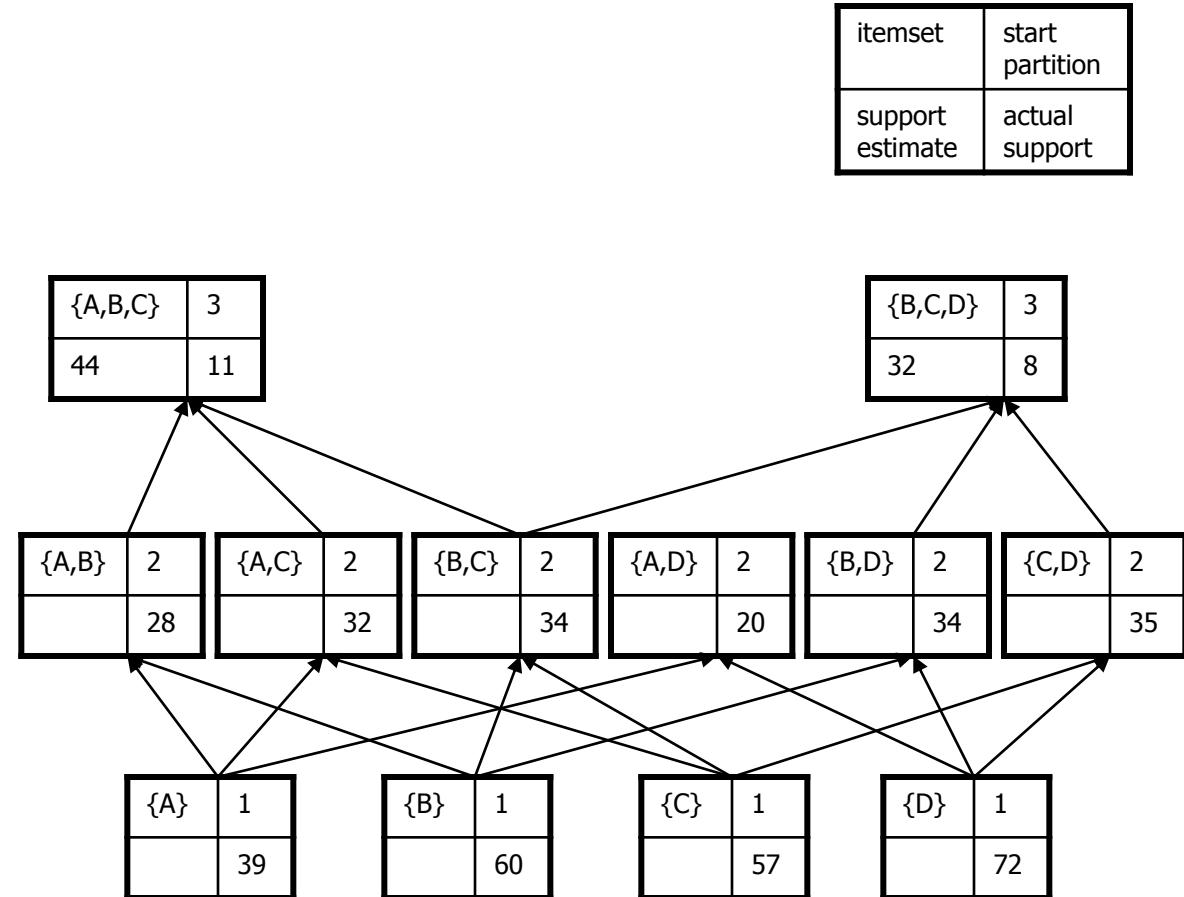
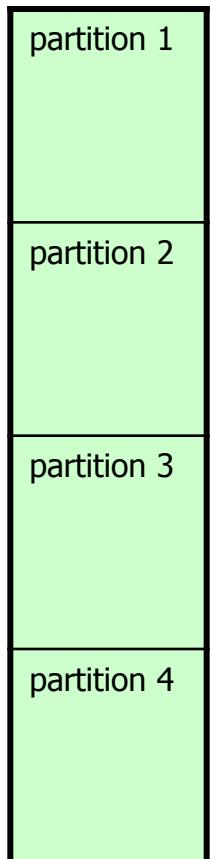
DIC: Example

Database D
of size 100



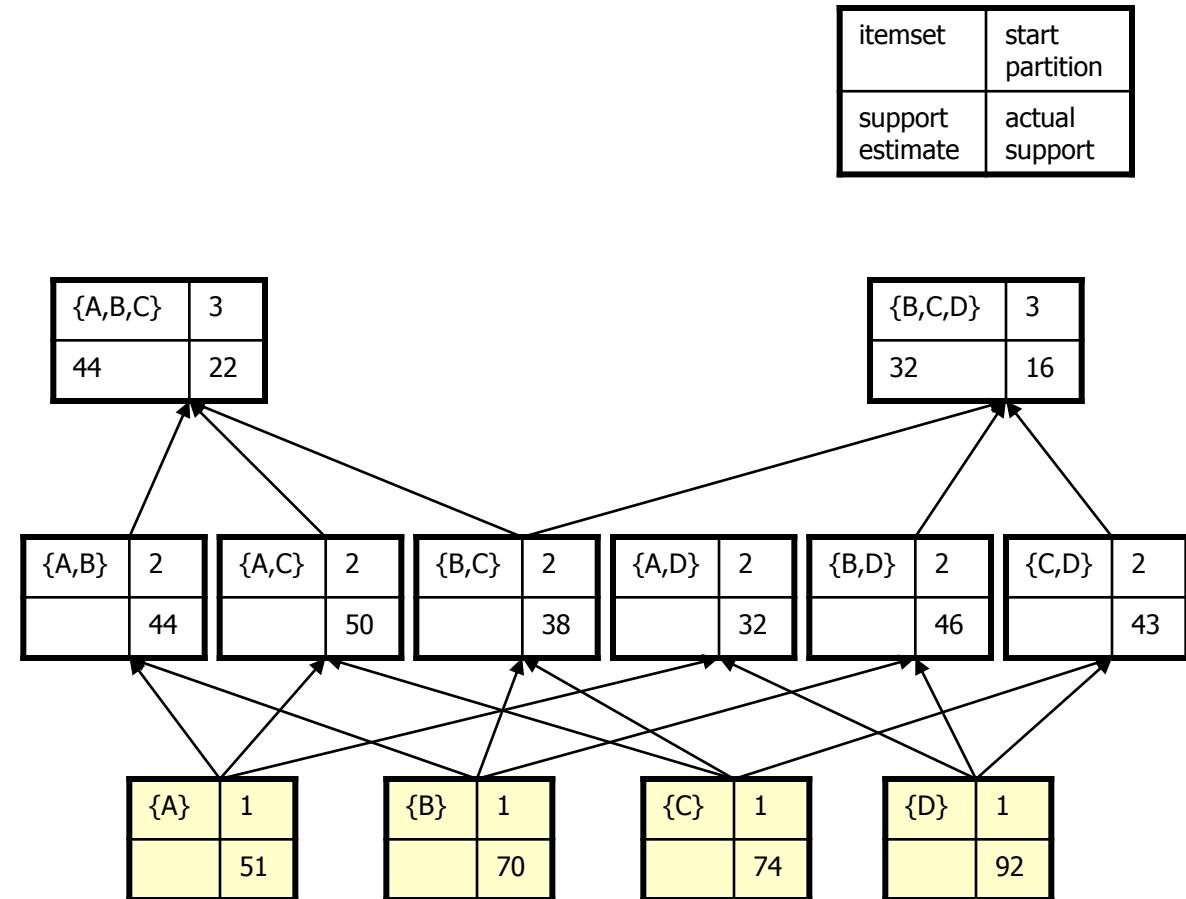
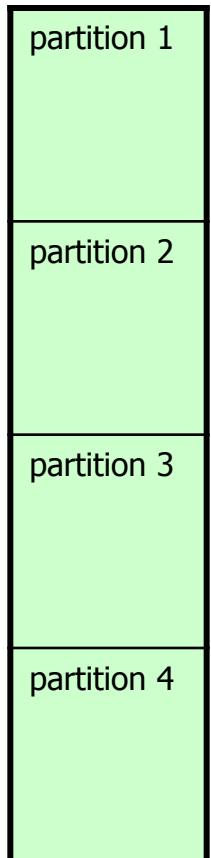
DIC: Example

Database D
of size 100



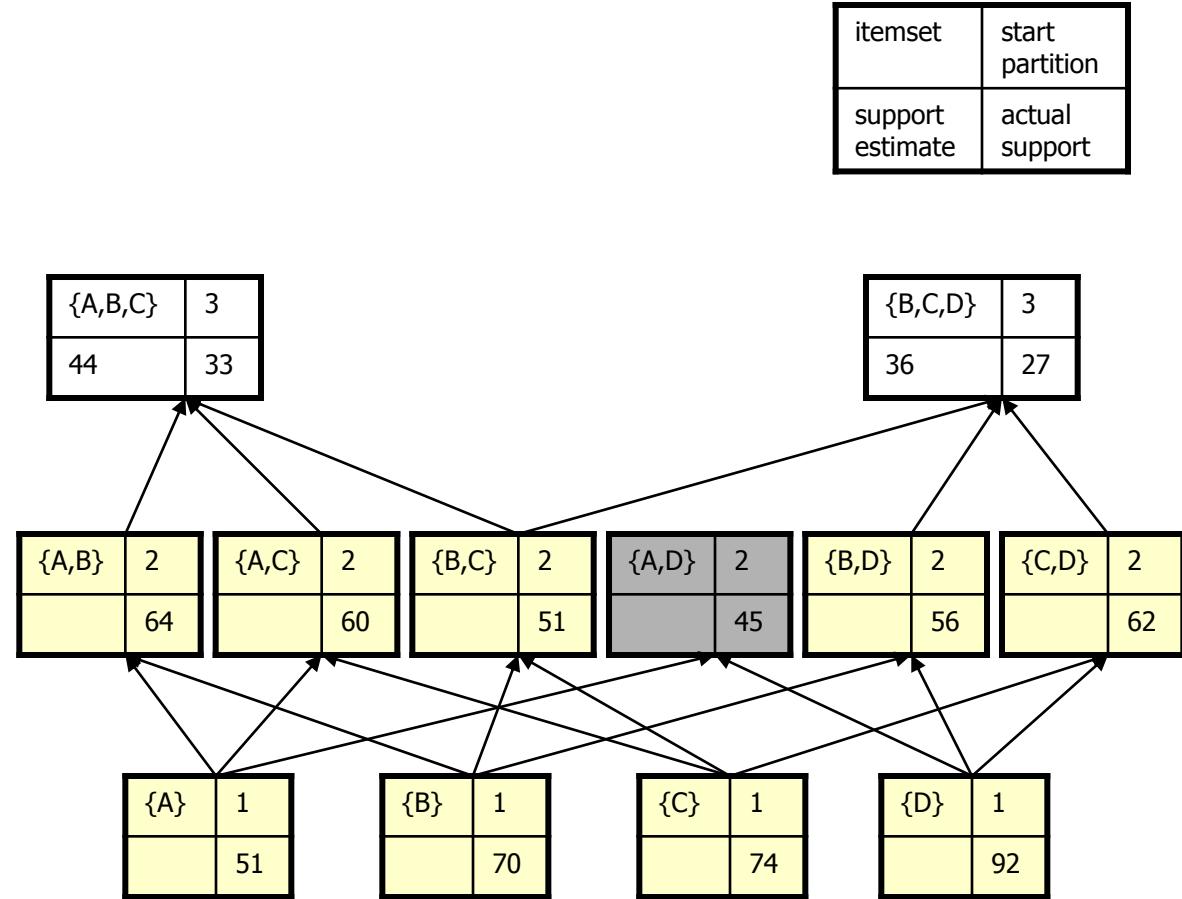
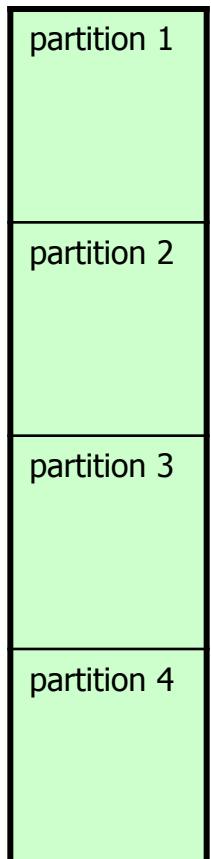
DIC: Example

Database D
of size 100



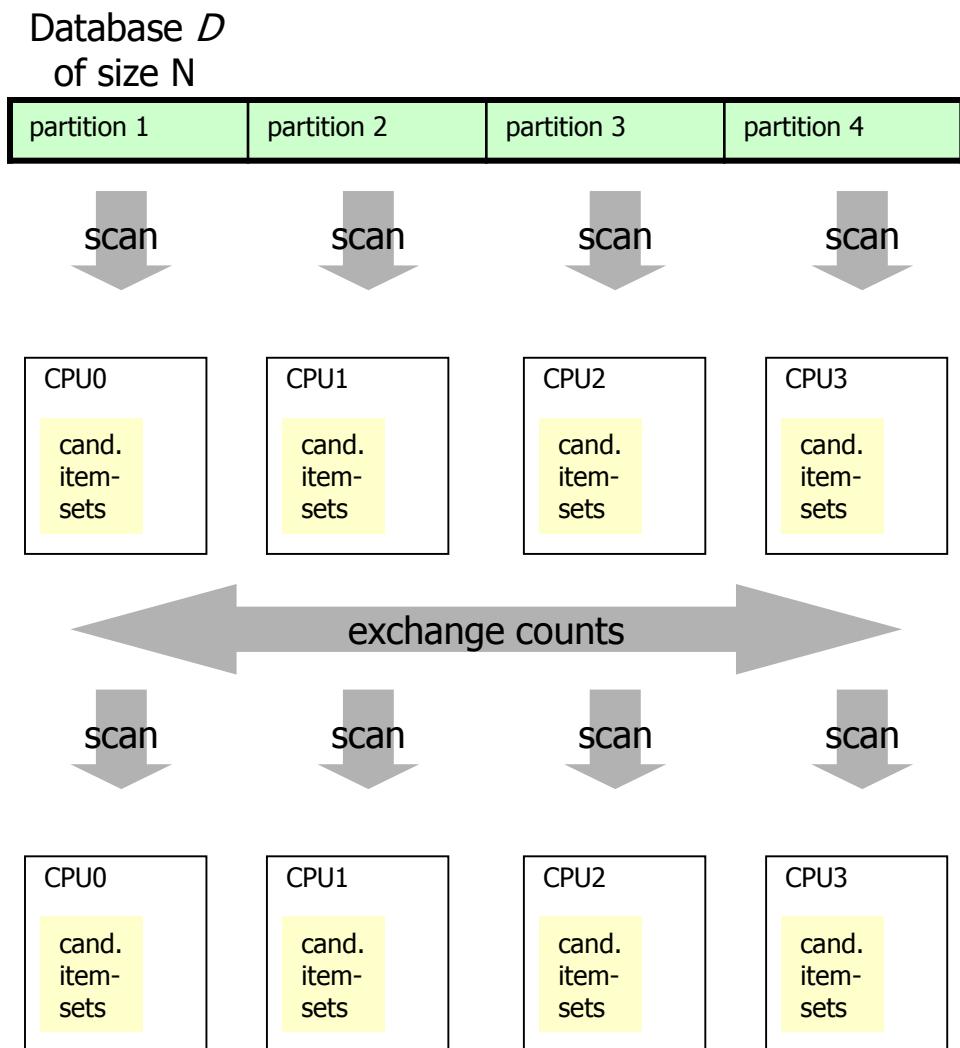
DIC: Example

Database D
of size 100



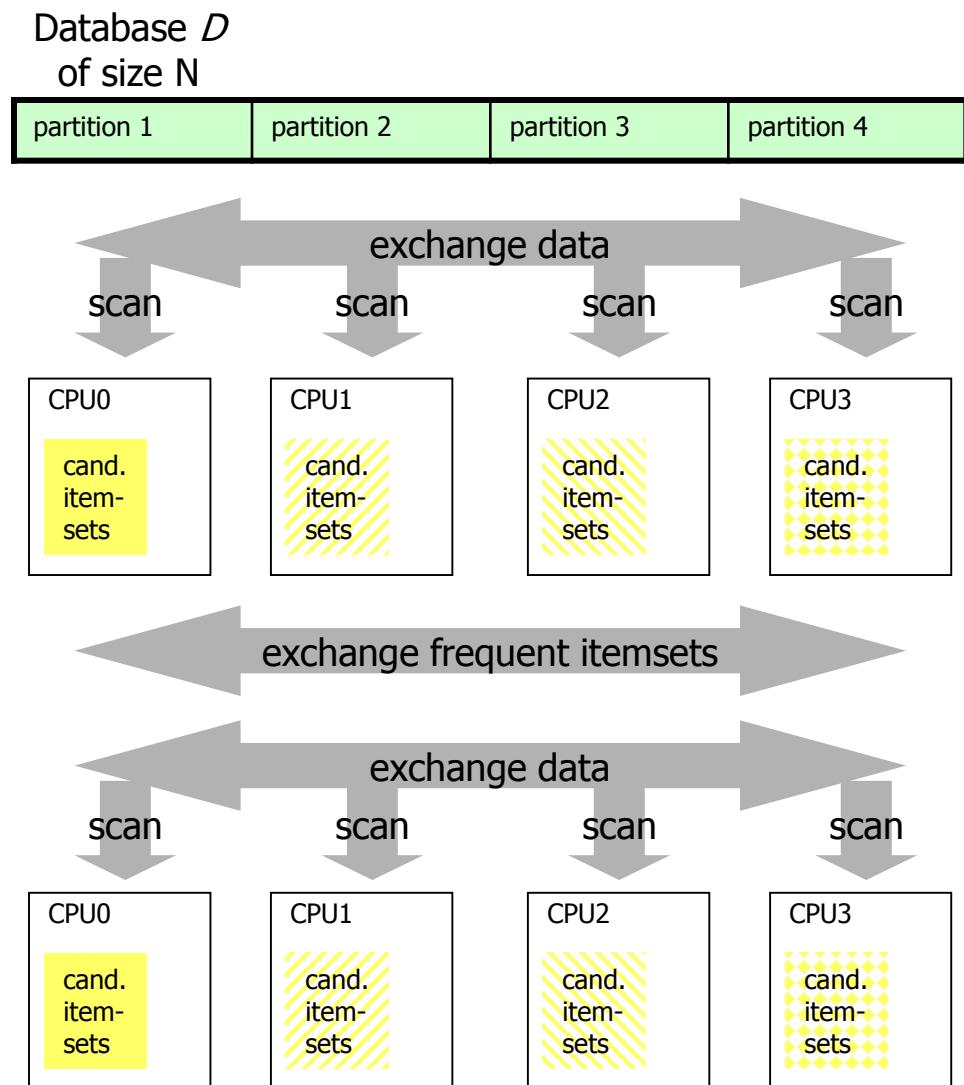
Parallel FID: Count Distribution

- All CPUs work on the same set of candidate itemsets.
- At the end of each pass:
 - local counts are distributed.
 - all nodes generate new candidate itemsets (identical on all nodes)
- Pros:
 - no transactional data exchanged
- Cons:
 - aggregate memory of the system is not exploited



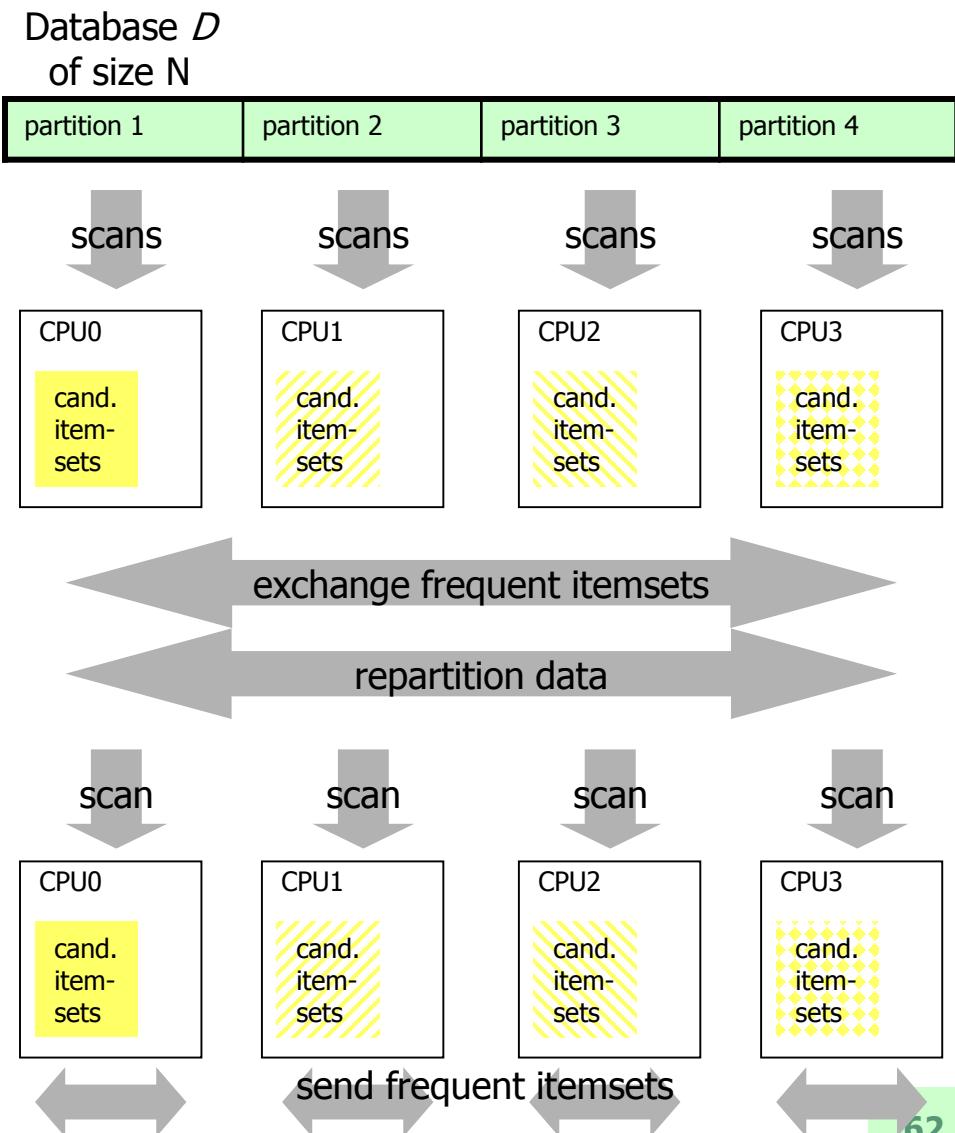
Parallel FID: Data Distribution

- All CPUs work on an individual set of candidate itemsets.
- As part of each pass:
 - data pages are distributed (each processor needs local data as well as data from other nodes)
 - frequent itemsets are determined and distributed (each processor needs to know all L_k to compute C_{k+1} for the next pass)
- Pros:
 - candidate itemsets are distributed → exploit memory
- Cons:
 - data tuples have to be exchanged

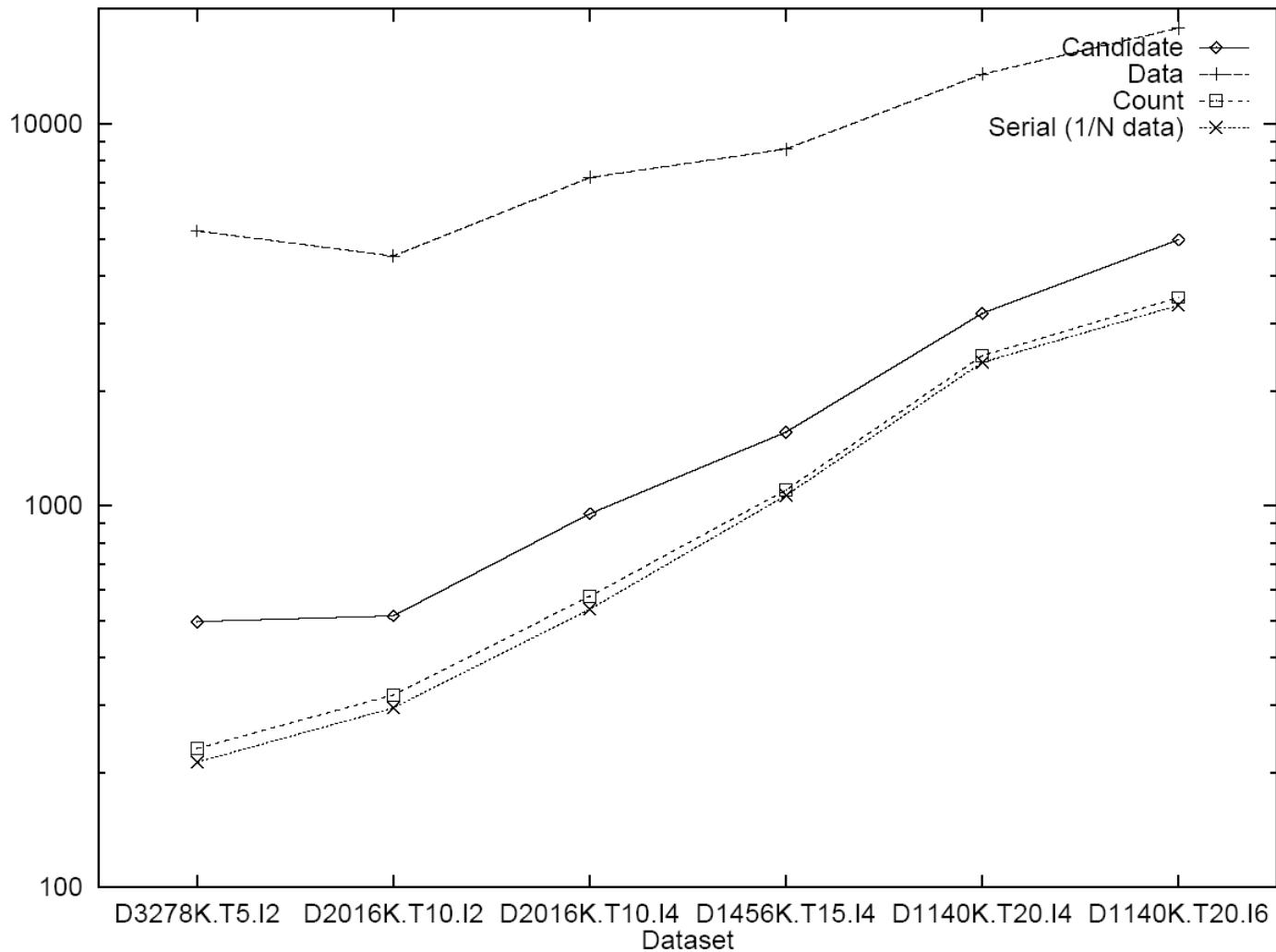


Parallel FID: Candidate Distribution

- Data distribution or count distribution is used until a certain pass I.
- Pass I:
 - partition frequent itemsets L_{k-1} among the processors
 - generate local candidates C_k
 - determine global counts for local candidate itemsets
 - repartition data according to the partitioning of frequent itemsets
 - broadcast frequent itemsets F_k
- Further passes:
 - based on local data
 - only frequent itemsets exchanged



Performance Evaluation



History of AR Research

- Apriori [Agrawal, Imielinski, Swami, SIGMOD 1993]
- Optimizations of Apriori
 - Fast algorithm [Agrawal, Srikant, VLDB 1994]
 - DHP: Direct Hash and Pruning [Park, Chen, Yu, SIGMOD 1995]
 - Partitioning Algorithm [Savasere, Omiecinski, Navathe, VLDB 1995]
 - Sampling [Toivonen, VLDB 1996]
 - DIC: Dynamic Itemset Counting [Brin, Motwani, Ullman, Tsur, SIGMOD 1997]
- Generalisations of the ARD problem
 - Generalized AR [Srikant, Agrawal, VLDB 1995] [Han, Fu, VLDB 1995]
 - Quantitative AR [Srikant, Agrawal, SIGMOD 1996]
 - N-Dimensional AR [Lu et. al., DMKD 1998]
 - Temporal AR [Ozden, Ramaswamy, Silberschatz, ICDE 1998]
- Parallel Mining [Agrawal, Shafer, TKDE 1996]
- Distributed Mining [Cheung, Ng, Fu, Fu, PDIIS 1996]
- Incremental Mining [Cheung, Ng, Wong, ICDE 1996]



Overview

- Introduction
 - Terms & Definitions
 - Disciplines & Applications
- Data Mining Techniques
 - Overview
 - Association Rule Discovery
 - Clustering
 - Classification
 - Regression
 - ...
- Data Mining Systems
 - Tools
 - Trends

Clustering

- **Given:**
A set of n d -dimensional objects
(tupels, records)
- **Problem:**
A natural partitioning of the object universe into a set of k **clusters** and a set of remaining objects (noise)
- **Properties:**
Objects of ...
 - ... the **same** cluster are **similar** (maximize intra-cluster similarity)
 - ... **different** clusters are **different** (minimize inter-cluster similarity)
- Clusters are not pre-defined but automatically generated
- Clustering = “unsupervised classification”
- Clustering Methods:
 - Partitioning methods
 - Hierarchical methods
 - Density-based methods
 - Grid-based methods
 - Model-based methods

Partitioning Methods

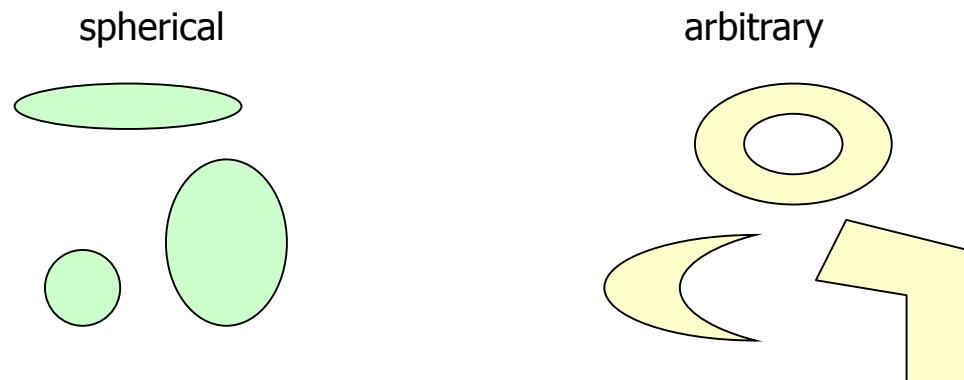
- A database of n objects is divided into k clusters such that:
 - each cluster contains at least one object
 - each object is assigned to exactly one cluster
- Basic algorithm:
 - create an initial partitioning
 - iterative relocation method attempts to improve the partitioning by moving objects from one cluster to another.
- Popular heuristics:
 - goal: avoid enumeration of all possible combinations
 - k-means: cluster is represented by the mean value of the objects in the cluster
 - k-medoids: cluster is represented by the object close to the center of the cluster

Hierarchical Methods

- Methods that create a hierarchical decomposition of the set of objects.
 - top-down:
 - start with all objects in one cluster
 - in each iteration a cluster is split into smaller clusters
 - until single-object clusters are reached or a termination condition holds
 - bottom-up:
 - start with each object forming a separate cluster
 - in each iteration two clusters (close to each other) are merged
 - until single cluster is reached or a termination condition holds
- Algorithms:
 - BIRCH
 - CURE

Density-based Methods

- General idea:
 - Continue growing the given cluster as long as the density (number of objects or data points) in the "neighborhood" exceeds some threshold. E.g. for each data point within a given cluster, the neighborhood of a given radius has to contain at least a minimum number of points.
- Advantage:
 - allows to filter out noise (outliers)
 - allows to derive clusters of arbitrary shape (in contrast to spherical shaped clusters)
- Algorithms:
 - DBSCAN
 - DENCLUE



Model-based Methods

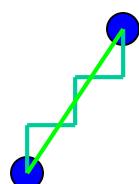
- General idea:
 - Use some model to describe each cluster.
 - Attempt to optimize the fit between the given data and the mathematical model.
 - Automatically determine the number of clusters based on standard statistics
- Approaches:
 - statistical
 - neural network
- Examples:
 - IBM Intelligent Miner demographic clustering
 - COBWEB

Grid-based Methods

- Clustering based on a grid data structure:
 - divide multi-dimensional space objects into a finite number of cells
 - clustering operations are performed on this grid structure
 - each grid cell may summarize the information of a group of objects that map into the cell
- Algorithms:
 - STING
 - WaveCluster
 - CLIQUE

Clustering

- Given two objects with n **continuous** attributes x (x_1, \dots, x_n) and y (y_1, \dots, y_n)
- The Minkowski-distance between x and y is defined as follows:
$$d(x, y) = (|x_1 - y_1|^q + \dots + |x_n - y_n|^q)^{1/q}$$
 - Euclidean distance: $q = 2$
 - Manhattan distance: $q = 1$
- K-Means
 - Determine k prototypes p_1, \dots, p_k within the given set of objects
 - Assign objects to the nearest prototype
 - Iterative algorithm:
 - Shift prototypes towards the mean of its point set
 - Reassign the objects to their (probably shifted) nearest prototype



K-Means Algorithm

- Given:
 - k: number of clusters
 - a “database” containing n objects
- Properties:
 - Time complexity is $O(nkt)$, where t is the number of iterations. Normally, $k \ll n$ and $t \ll n$.
 - Unsuitable to find clusters with non-convex shapes.

```
k_means() begin
    arbitrarily choose k objects as the initial cluster centers;
    repeat
        assign each object to the cluster to which the object is
        the most similar based on the mean value of the objects
        of the cluster;
        calculate the mean value of the objects for each cluster;
    until no change;
end;
```

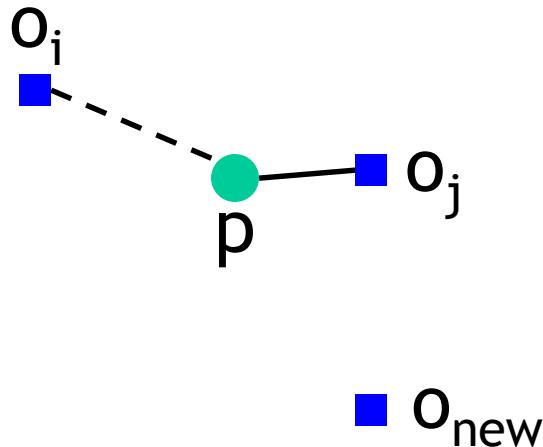
K-Medoids Algorithm

- **Given:**
 - k: number of clusters
 - a “database” containing n objects
- Cost function:
 - based on **difference** in square-error if current medoid is replaced by non-medoid object.
 - Total cost is sum of costs incurred by all non-medoid objects.
 - Minimize distance stop-criterion: $\sum_{i=1..k} \sum_{j=1..n} d(p_i, x_j^i)$
- Properties:
 - K-Medoids is more robust to outliers than k-Means
 - K-Medoids processing is more costly than k-Means

K-Medoids Algorithm

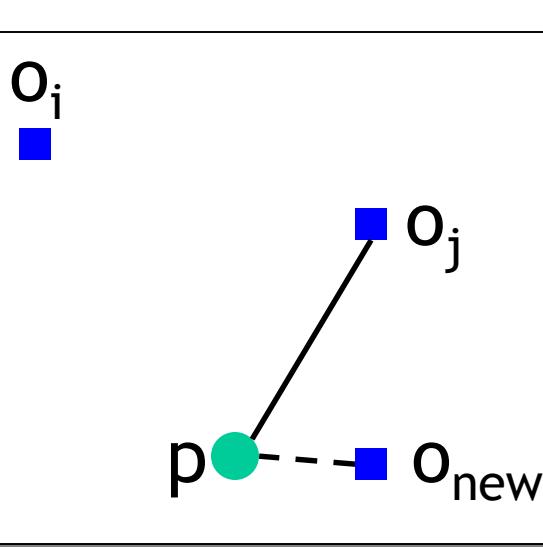
```
k_medoids() begin
arbitrarily choose k objects as the initial medoids;
repeat
    assign each remaining object to the cluster with the nearest medoid;
    for cluster j = 1 to k do begin
        o_new = randomly selected non-medoid object in cluster j;
        S      = total cost of swapping the current medoid o_j with o_new;
        if S < 0 then
            swap o_j with o_new to form a new medoid for the cluster j;
        end;
    until no change;
end;
```

K-Medoids: Reassignment of Objects to Clusters (1/2)



Case 1:

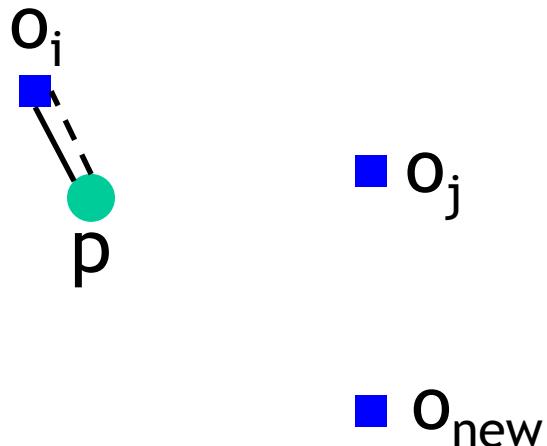
p currently belongs to medoid o_j .
If o_j is replaced by o_{new} as a medoid
and p is closest to one of o_i , $i \neq j$,
then p is reassigned to o_i .



Case 2:

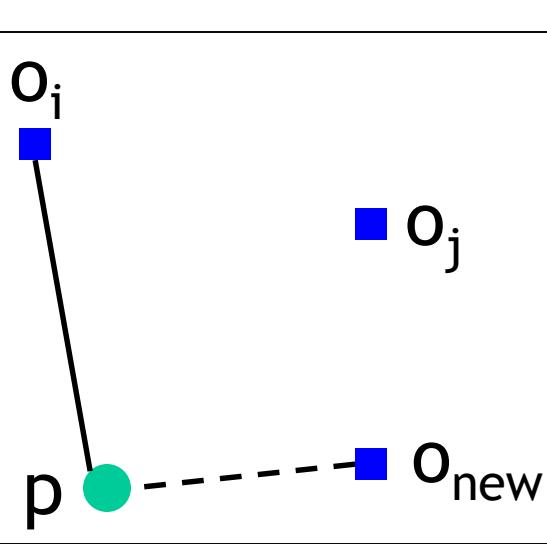
p currently belongs to medoid o_j .
If o_j is replaced by o_{new} as a medoid
and p is closest to o_{new} ,
then p is reassigned to o_{new} .

K-Medoids: Reassignment of Objects to Clusters (2/2)



Case 3:

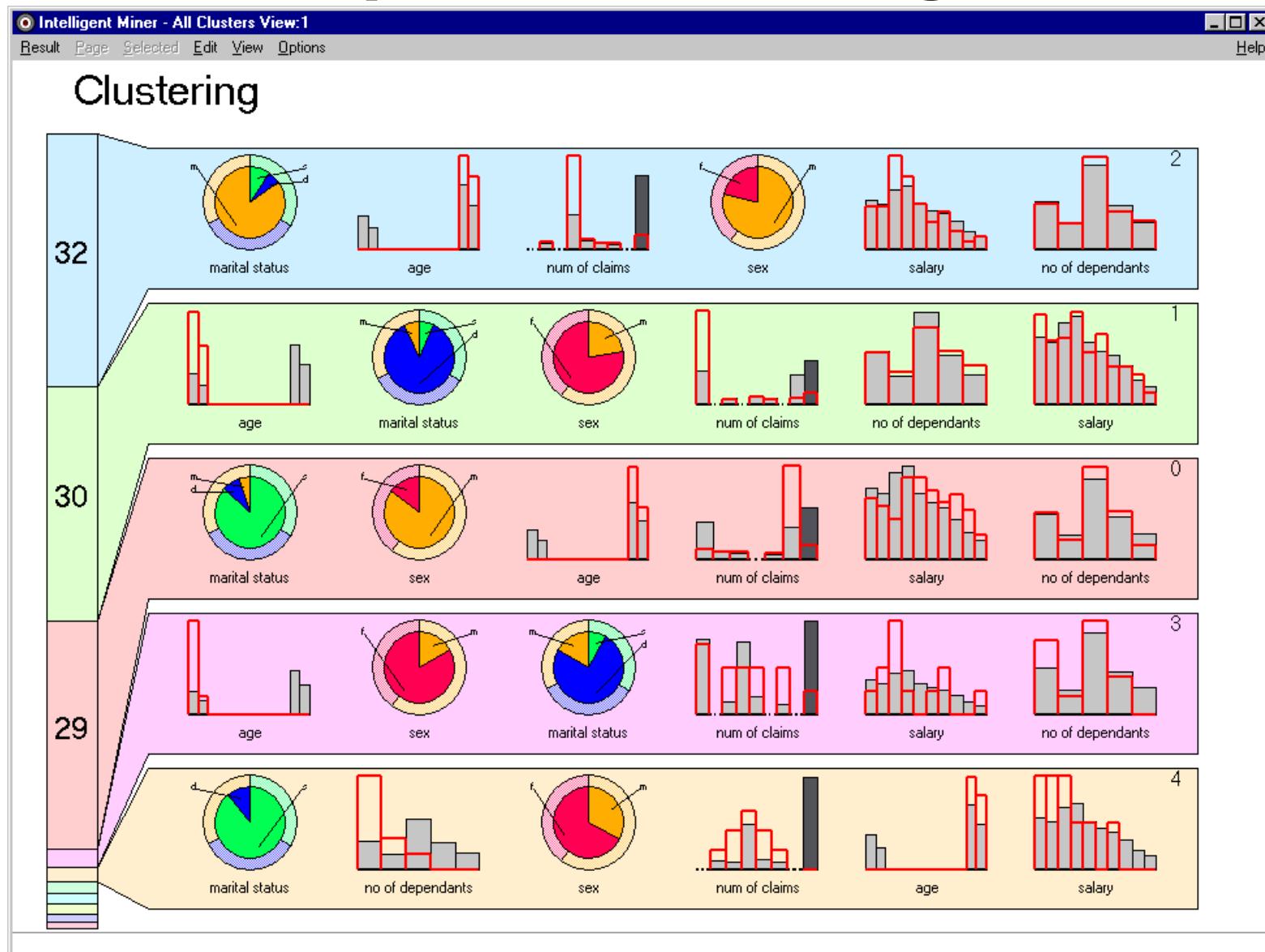
p currently belongs to medoid o_i , $i \neq j$.
If o_j is replaced by o_{new} as a medoid
and p is still closest to o_i ,
then the assignment does not change.



Case 4:

p currently belongs to medoid o_i , $i \neq j$.
If o_j is replaced by o_{new} as a medoid
and p is closest to o_{new} ,
then p is reassigned to o_{new} .

Example of a Clustering Result



Interpretation of the Clustering Result

- Horizontal strip is a cluster
- Clusters are ordered by size from top to bottom
- Variables are ordered by importance to the cluster from left to right (based on a χ^2 -test between variable and cluster ID; further measures are entropy and Condorcet-criterion)
- Bar graph: frequency distribution of numeric (integer), binary and **continuous** variables
 - red bars: distribution of the variable within the cluster
 - gray solid bars: distribution of the variable within the whole data set (universe)
- Pie charts: **categorical** variables
 - inner pie: distribution of the categories within the current cluster
 - outer: distribution of the categories within the whole data set
- The more different the cluster distribution is from the average (universe), the more „interesting“ or distinct the cluster is

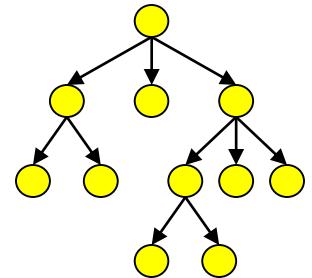


Overview

- Introduction
 - Terms & Definitions
 - Disciplines & Applications
- Data Mining Techniques
 - Overview
 - Association Rule Discovery
 - Clustering
 - Classification
 - Regression
 - ...
- Data Mining Systems
 - Tools
 - Trends

Classification: Decision Trees

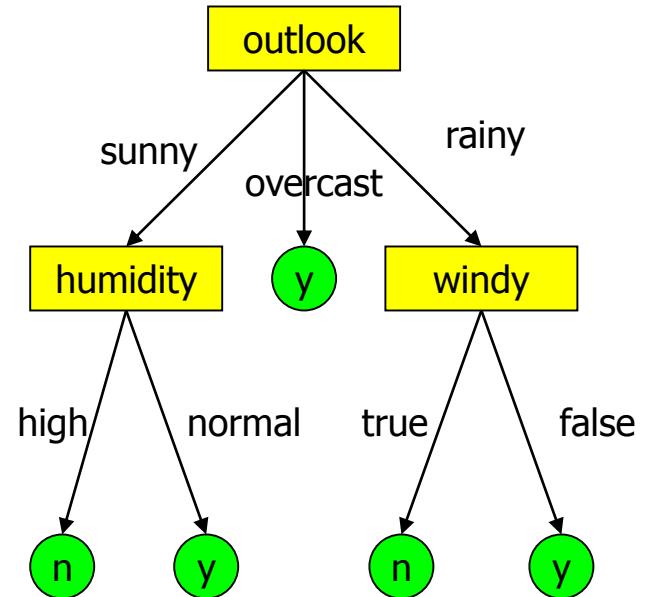
- A tree, where
 - an inner node is a test (condition) on a single attribute,
 - an edge is the result of a test, and
 - a leaf node represents a class.
- Each path from the root to a leaf is equivalent to a classification rule
- Characteristics of the generic algorithm
 - top-down recursive
 - divide & conquer
 - greedy (danger: a local maximum is not always the global maximum)



Decision Trees Example: Play Tennis?

D

outlook	temperature	humidity	windy	play
sunny	hot	high	false	n
sunny	hot	high	true	n
overcast	hot	high	false	y
rainy	mild	high	false	y
rainy	cool	normal	false	y
rainy	cool	normal	true	n
overcast	cool	normal	true	y
sunny	mild	high	false	n
sunny	cool	normal	false	y
rainy	mild	normal	false	y
sunny	mild	normal	true	y
overcast	mild	high	true	y
overcast	hot	normal	false	y
rainy	mild	high	true	n



IF outlook = sunny AND humidity = normal
THEN play tennis

Decision Tree Induction: Generic Algorithm

```
Node generateDecisionTree(samples, attributeList) begin
    create a new node n;
    if samples are all of class c then begin
        label leaf node n with c;
        return n;
    else if attributeList == Ø then begin
        label leaf node n with majorityClass(n);
        return n;
    end;
    select bestSplit from attributeList; // Depends on specific algorithm!

    for each value v of attribute bestSplit do begin
        sv = set of samples with (bestSplit == v);
        if sv == Ø then begin
            create a new node nv and label it with majorityClass(samples)
        else
            nv = generateDecisionTree(sv, attributeList \ bestSplit);
        end;
        create a branch from n to nv labeled with the test (bestSplit == v);
    end;
    return n;
end;
```

Generic Algorithm: Example

D₁

outlook	temperature	humidity	windy	play
overcast	hot	high	false	y
overcast	cool	normal	true	y
overcast	mild	high	true	y
overcast	hot	normal	false	y

D₂

outlook	temperature	humidity	windy	play
sunny	hot	high	false	n
sunny	hot	high	true	n
sunny	mild	high	false	n
sunny	cool	normal	false	y
sunny	mild	normal	true	y

D₃

outlook	temperature	humidity	windy	play
rainy	mild	high	false	y
rainy	cool	normal	false	y
rainy	cool	normal	true	n
rainy	mild	normal	false	y
rainy	mild	high	true	n

generateDecisionTree(D,{o,t,h,w})

new node: 

best split: outlook

sv=D₁

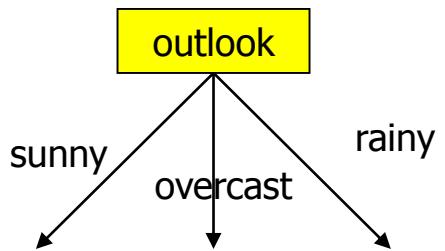
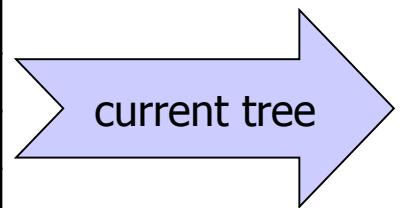
→ generateDecisionTree(D₁,{t,h,w})

sv=D₂

→ generateDecisionTree(D₂,{t,h,w})

sv=D₃

→ generateDecisionTree(D₃,{t,h,w})



Generic Algorithm: Example

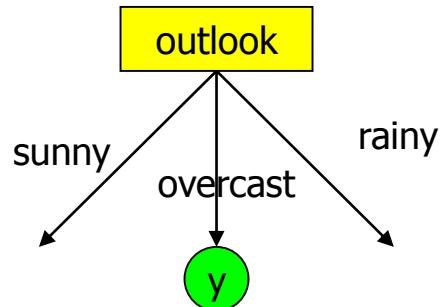
D₁

outlook	temperature	humidity	windy	play
overcast	hot	high	false	y
overcast	cool	normal	true	y
overcast	mild	high	true	y
overcast	hot	normal	false	y

generateDecisionTree(D₁, {t,h,w})

new node: 

samples are all of class *play=y*



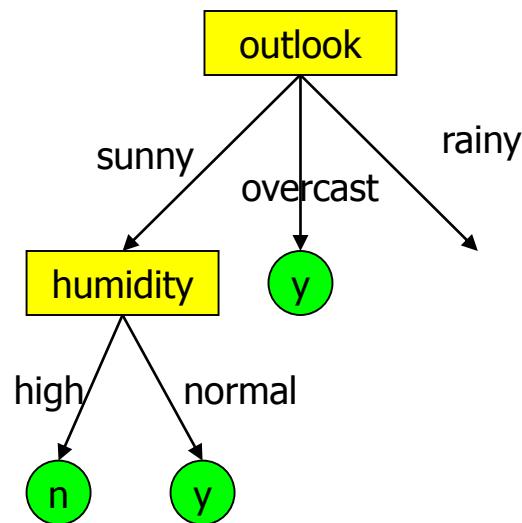
Generic Algorithm: Example

D_{21}

outlook	temperature	humidity	windy	play
sunny	hot	high	false	n
sunny	hot	high	true	n
sunny	mild	high	false	n

D_{22}

outlook	temperature	humidity	windy	play
sunny	cool	normal	false	y
sunny	mild	normal	true	y



generateDecisionTree($D_2, \{t, h, w\}$)

new node: 

best split: humidity

sv= D_{21}

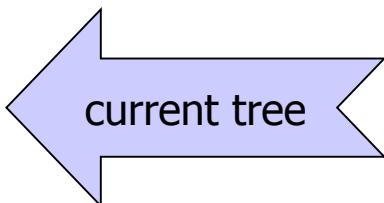
→ generateDecisionTree($D_{21}, \{t, w\}$)

→ samples are all of class *play*=n

sv= D_{22}

→ generateDecisionTree($D_{22}, \{t, w\}$)

→ samples are all of class *play*=y

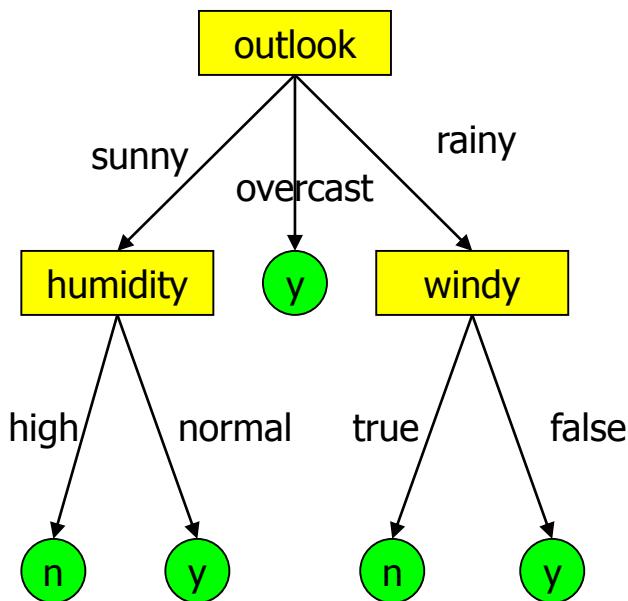


D_{31}

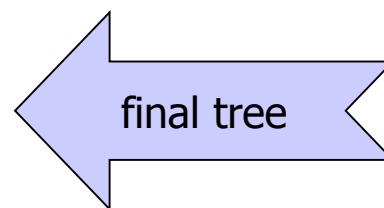
outlook	temperature	humidity	windy	play
rainy	mild	high	false	y
rainy	cool	normal	false	y
rainy	mild	normal	false	y

 D_{32}

outlook	temperature	humidity	windy	play
rainy	cool	normal	true	n
rainy	mild	high	true	n

generateDecisionTree($D_3, \{t, h, w\}$)new node: 

best split: windy

sv= D_{31} → generateDecisionTree($D_{31}, \{t, h\}$)→ samples are all of class *play*=ysv= D_{32} → generateDecisionTree($D_{32}, \{t, h\}$)→ samples are all of class *play*=n

Characteristics of Algorithms

- Sorts of splits
 - arity/degree of split (binary, ternary, etc.)
 - categorical vs. continuous attributes
- Stop rule: How to decide that a node is a leaf?
 - all samples belong to the same class (samples are all of class c)
 - impurity measure value is below a given threshold
 - no attribute remaining to be split (`attributeList == Ø`)
 - no samples in the partition
- Labeling rule
 - a leaf is labeled with the class which the majority of the samples belong to

Overfitting Problem

- Ideal classification:
Find the most „simple“ decision tree that suits to the training data, i.e. find the most simple model that is fulfilled by the training data.
- Overfitting
 - Spurious patterns may emerge by chance
- Overfitting is caused by:
 - noise & outliers
 - a too small set of training data
 - local maxima in the data
- two heuristics are frequently used to avoid overfitting:
 - stop earlier: interrupt splitting further nodes
 - post-prune: allow overfitting and simplify the tree afterwards

Algorithms & their Split Criteria

- There are many algorithms, that mainly differ in the divide (split) criterion:
- Machine learning: ID3 (Iterative Dichotomizer) [Quinlan, 1986], C4.5 [Quinlan, 1993]
 - entropy (concept in information theory) measures „impurity“ of a split
 - choose attribute maximizing entropy reduction
- Statistics: CART (Classification And Regression Trees) [Breiman et. al., 1984]
 - gini index is another measure for „impurity“
 - choose attribute minimizing the gini index
- Pattern recognition: ChAID (Chi-squared Automated Interaction Detection) [Magidson, 1994]
 - measures correlation between each attribute and its corresponding class label
 - choose attribute with maximum correlation

Example Split Criterion: Gini Index of CART-Algorithm

Given:

- two classes: Pos und Neg
- a data set S with p elements in class Pos and n elements in Neg

$$f_p = p / (p + n)$$

$$f_n = n / (p + n)$$

$$\text{gini}(S) = 1 - f_p^2 - f_n^2$$

Let S_1 and S_2 be data sets, where $S_1 \cup S_2 = S$ and $S_1 \cap S_2 = \emptyset$

$$\text{gini}_{\text{split}}(S_1, S_2) = \frac{\text{gini}(S_1) \times (p_1 + n_1) / (p + n)}{} + \frac{\text{gini}(S_2) \times (p_2 + n_2) / (p + n)}{}$$

Aim: Find S_1 and S_2 such that $\text{gini}_{\text{split}}(S_1, S_2)$ is minimized.

Gini Index Example

outlook	temperature	humidity	windy	play
sunny	hot	high	false	n
sunny	hot	high	true	n
overcast	hot	high	false	y
rainy	mild	high	false	y
rainy	cool	normal	false	y
rainy	cool	normal	true	n
overcast	cool	normal	true	y
sunny	mild	high	false	n
sunny	cool	normal	false	y
rainy	mild	normal	false	y
sunny	mild	normal	true	y
overcast	mild	high	true	y
overcast	hot	normal	false	y
rainy	mild	high	true	n

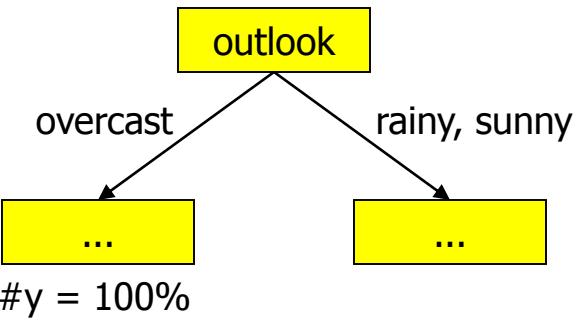
Assumption: We want to split the root node on the attribute **outlook**.

$$\begin{aligned}
 S_1 &= \{\text{overcast}\}, \text{gini} = 1 - (4/4)^2 - (0/4)^2 = 0.000 \\
 S_2 &= \{\text{sunny}\}, \text{gini} = 1 - (2/5)^2 - (3/5)^2 = 0.480 \\
 S_3 &= \{\text{rainy}\}, \text{gini} = 1 - (3/5)^2 - (2/5)^2 = 0.480 \\
 S_4 &= \{\text{overcast, sunny}\}, \text{gini} = 1 - (6/9)^2 - (3/9)^2 = 0.444 \\
 S_5 &= \{\text{overcast, rainy}\}, \text{gini} = 1 - (7/9)^2 - (2/9)^2 = 0.346 \\
 S_6 &= \{\text{sunny, rainy}\}, \text{gini} = 1 - (5/10)^2 - (5/10)^2 = 0.500
 \end{aligned}$$

$$\text{gini}_{\text{split}}(S_1, S_6) = 0.000 \times 4/14 + 0.500 \times 10/14 = 0.357$$

$$\begin{aligned}
 \text{gini}_{\text{split}}(S_2, S_5) &= 0.480 \times 5/14 + 0.346 \times 9/14 = 0.394 \\
 \text{gini}_{\text{split}}(S_3, S_4) &= 0.480 \times 5/14 + 0.444 \times 9/14 = 0.457
 \end{aligned}$$

Choose minimum → $S_1 = \{\text{overcast}\}$, $S_6 = \{\text{sunny, rainy}\}$



Example Split Criterion: Information Gain

Given:

- set S of s data samples
- class label has m distinct values, resulting in classes C_i ($i=1,\dots,m$)
- p_i is the probability that a sample belongs to class C_i
- attribute A having distinct values $\{a_1, a_2, \dots, a_v\}$
- S is partitioned in subsets $\{S_1, S_2, \dots, S_v\}$ by attribute A where subset S_j contains samples with value a_j of A
- s_{ij} is the number of samples of class C_i in subset S_j
- p_{ij} is the probability that a sample in S_j belongs to class C_i

$$p_{ij} = \frac{s_{ij}}{|S_j|}$$

Example Split Criterion: Information Gain

- Information needed to classify a given sample: $I(s_1, s_2, \dots, s_m) = -\sum_{i=1}^m p_i \log_2(p_i)$
- For each subset S_j : $I(s_{1j}, \dots, s_{mj}) = -\sum_{i=1}^m p_{ij} \log_2(p_{ij})$
- Entropy for partitioning by attribute A: $E(A) = \sum_{j=1}^v \frac{s_{1j} + \dots + s_{mj}}{s} I(s_{1j}, \dots, s_{mj})$
- Information **gain** for partitioning by attribute A: $Gain(A) = I(s_1, s_2, \dots, s_m) - E(A)$
- **Aim:** Identify split attribute with maximum information gain.
- Remarks:
 - Gain is the expected reduction in entropy caused by knowing the value of attribute A.
 - Gini index and information gain disagree on split in only 2%. Source: [RS04]

Information Gain Example

outlook	temperature	humidity	windy	play
sunny	hot	high	false	n
sunny	hot	high	true	n
overcast	hot	high	false	y
rainy	mild	high	false	y
rainy	cool	normal	false	y
rainy	cool	normal	true	n
overcast	cool	normal	true	y
sunny	mild	high	false	n
sunny	cool	normal	false	y
rainy	mild	normal	false	y
sunny	mild	normal	true	y
overcast	mild	high	true	y
overcast	hot	normal	false	y
rainy	mild	high	true	n

$$S_1 = \{play=y\}; S_2 = \{play=n\}; s_1 = 9; s_2 = 5$$

$$I(s_1, s_2) = -9/14 \log_2 9/14 - 5/14 \log_2 5/14 = 0.940$$

Gain of split on attribute outlook:

$$\text{outlook=sunny: } s_{11} = 2, s_{21} = 3:$$

$$I(s_{11}, s_{21}) = -2/5 \log_2 2/5 - 3/5 \log_2 3/5 = 0.971$$

$$\text{outlook=overcast: } s_{12} = 4, s_{22} = 0:$$

$$I(s_{12}, s_{22}) = -1 \log_2 1 - 0 \log_2 0 = 0$$

$$\text{outlook=rainy: } s_{13} = 3, s_{23} = 2:$$

$$I(s_{13}, s_{23}) = -3/5 \log_2 3/5 - 2/5 \log_2 2/5 = 0.971$$

$$E(\text{outlook}) = 5/14 * 0.971 + 4/14 * 0 + 5/14 * 0.971 = 0.693$$

$$\text{Gain(outlook)} = 0.940 - 0.693 = 0.247$$

$$\text{Gain(temperature)} = 0.029$$

$$\text{Gain(humidity)} = 0.152$$

$$\text{Gain(windy)} = 0.048$$

=> split on attribute outlook

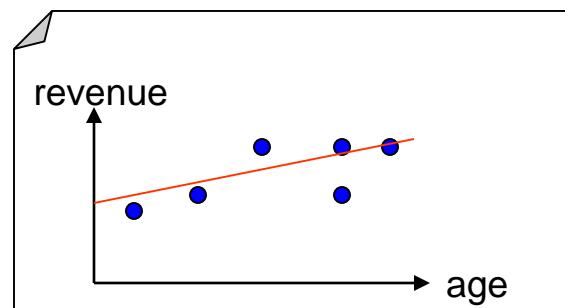
Classification: Exotics

- Apart from decision tree induction there are several further methods for classification:
 - Statistics
 - Bayesian Classification based on Bayes theorem: $P(C_i | X) = \frac{P(X | C_i)P(C_i)}{P(X)}$
 - Artificial neural networks
 - Genetic algorithms
 - Classification based on association rules:
 - constraint-based ARD may deliver rules of the form:

$A_{\text{quan1}} \text{ AND } A_{\text{quan2}} \rightarrow A_{\text{cat}}$

Regression

- Construction of models of continuous attributes as functions of other attributes
- Example: a prediction model of sales of a product given its price
- Many problems solvable by linear regression, where attribute (response variable) y is modeled as a linear function of other attributes x_1, \dots, x_n (predictor variables):
$$y = a_0 + a_1x_1 + \dots + a_nx_n$$
- Coefficients a_0, \dots, a_n are computed from the training data set using the least square method





Overview

- Introduction
 - Terms & Definitions
 - Disciplines & Applications
- Data Mining Techniques
 - Overview
 - Association Rule Discovery
 - Clustering
 - Classification
 - Regression
 - ...
- Data Mining Systems
 - Tools
 - Trends

Classification of Data Mining Systems

- Kind of database and kind of data
 - relational, object-relational, object-oriented, ...
 - spatial, time-series, multimedia, ...
 - Kind of techniques
 - degree of user interaction: autonomous, interactive, ...
 - methods from database area, machine learning, statistics, visualization
 - Kind of knowledge
 - data mining functionalities like characterization, association, classification, clustering, ...
 - level of abstraction
 - regularities vs. irregularities
 - Kind of applications
 - finance, telecommunications, stock markets, ...
- 

Architecture of Data Mining Systems

Coupling of data mining system (DMS) and database system

- no coupling
 - data mining purely based on files
 - no database system and no data warehouse involved
 - DMS has to provide flexibility and efficiency for data storage and processing
 - data pre-processing has to be provided by the DMS
- semitight coupling
 - database systems provide essential data mining primitives
 - intermediate mining results are stored in the database
- loose coupling
 - database system / data warehouse provides data for mining tasks and stores results
 - mining tasks are main memory based in this setting
 - lack of performance and scalability
- tight coupling
 - database provides a data mining subsystem
 - data mining queries are optimized in the database system

Some Commercial Data Mining Products

Company	Product
IBM	InfoSphere Warehouse (formerly Intelligent Miner)
Oracle	Oracle Data Mining (ODM)
Angoss	KnowledgeStudio
SAS	Enterprise Miner
SPSS	Clementine

Overview of companies: www.kdnuggets.com/companies/products.html

IBM InfoSphere Warehouse

(formerly IBM Intelligent Miner)

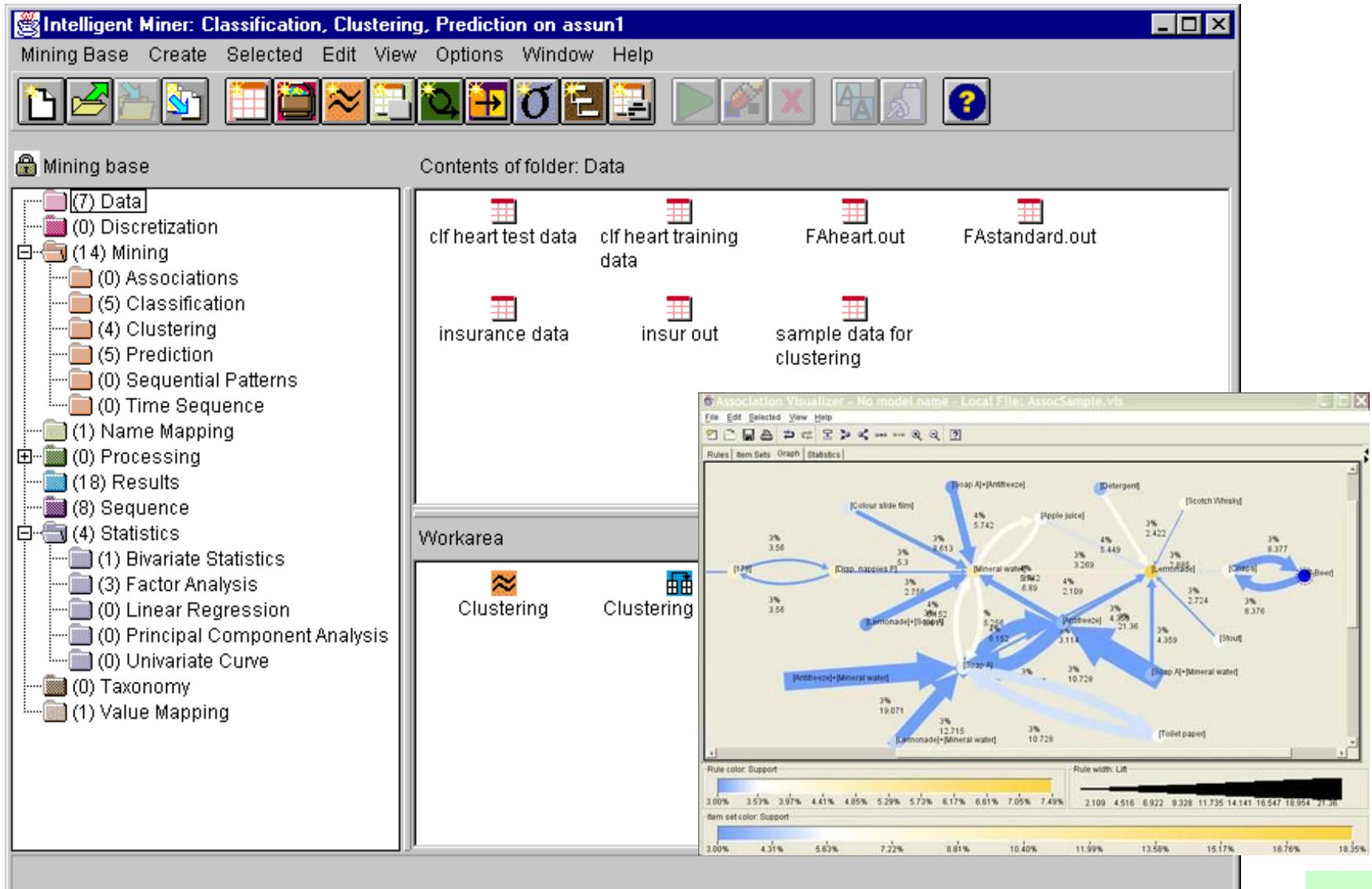
Intelligent Miner: Classification, Clustering, Prediction on assun1

Mining Base Create Selected Edit View Options Window Help

Mining base Contents of folder: Data

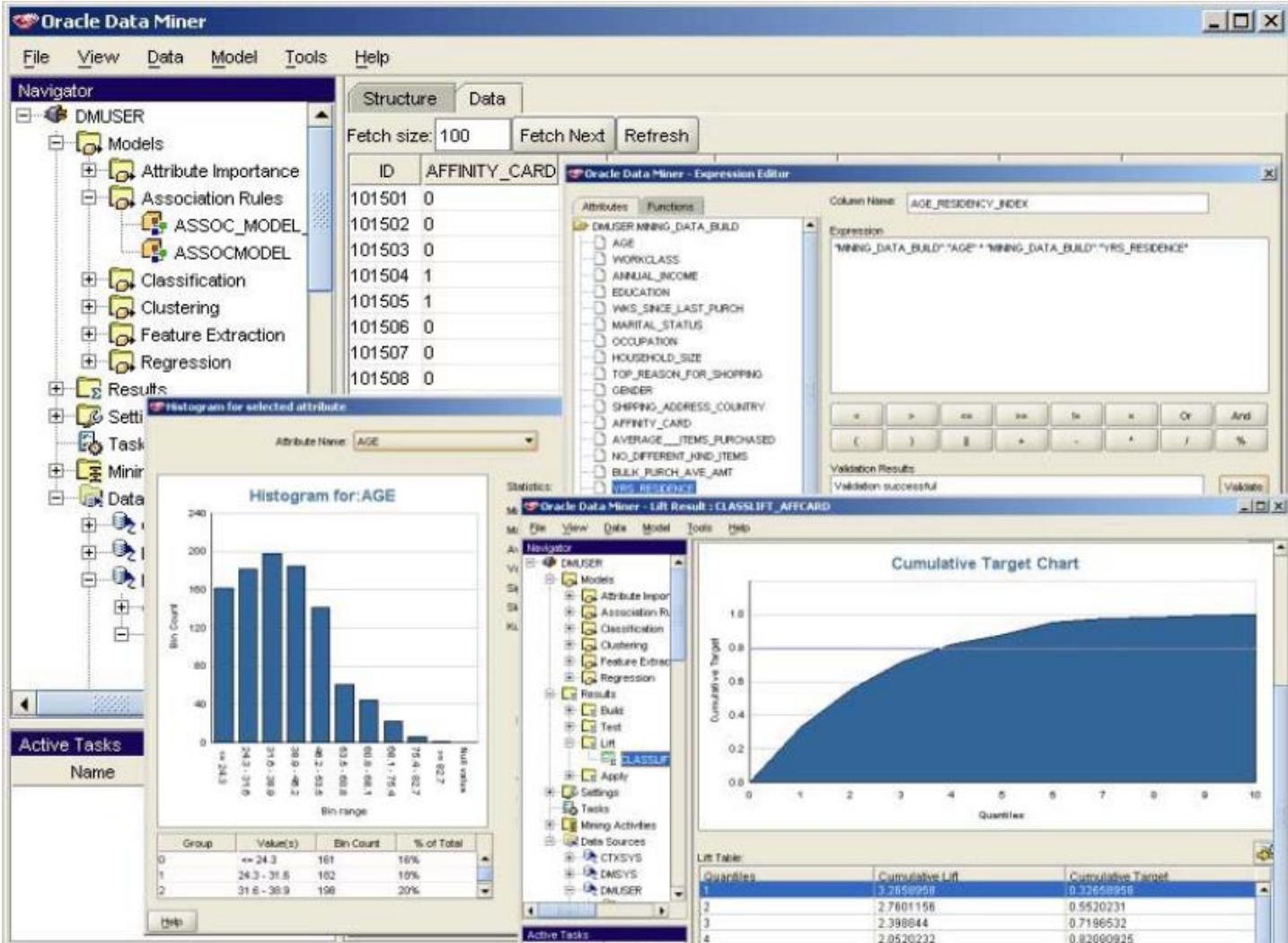
Workarea

Association Visualizer - No model name - Local File: AssocSample.viz.



The screenshot displays the IBM Intelligent Miner application. The main window has a blue header bar with the title "Intelligent Miner: Classification, Clustering, Prediction on assun1" and a menu bar with options like Mining Base, Create, Selected, Edit, View, Options, Window, and Help. Below the menu is a toolbar with various icons. The left panel, titled "Mining base", contains a tree view of mining components: Data, Discretization, Mining (Associations, Classification, Clustering, Prediction, Sequential Patterns, Time Sequence), Name Mapping, Processing, Results, Sequence, and Statistics (Bivariate Statistics, Factor Analysis, Linear Regression, Principal Component Analysis, Univariate Curve). The "Mining" node is expanded. The right panel, titled "Contents of folder: Data", shows files: clf heart test data, clf heart training data, Fheart.out, FAstandard.out, insurance data, insur.out, and sample data for clustering. A separate window titled "Association Visualizer" is open, showing a graph of item sets and their support values. The graph includes nodes like [Colour slide film], [Soap A]+[Antifreeze], [Detergent], [Scotch Whisky], [Centrifugal], [Beef], [Cassis], [Stou], [Toilet paper], [Lemonade]+[Soap A], [Mineral water], [Apple juice], [Antifreeze], [Soap A], [Toothpaste], [Lemonade]+[Mineral water], and [Toothpaste]. Support values range from 3.56% to 19.071%.

Oracle Data Mining (ODM)



The screenshot displays the Oracle Data Miner application window. The left sidebar shows a tree view of mining tasks under 'DMUSER':

- Models
 - Attribute Importance
 - Association Rules
 - ASSOC_MODEL
 - ASSOCMODEL
 - Classification
 - Clustering
 - Feature Extraction
 - Regression
- Results
- Settings
- Tasks
- Mining Activities
- Data
 - Histogram for selected attribute
 - Attribute Name: AGE

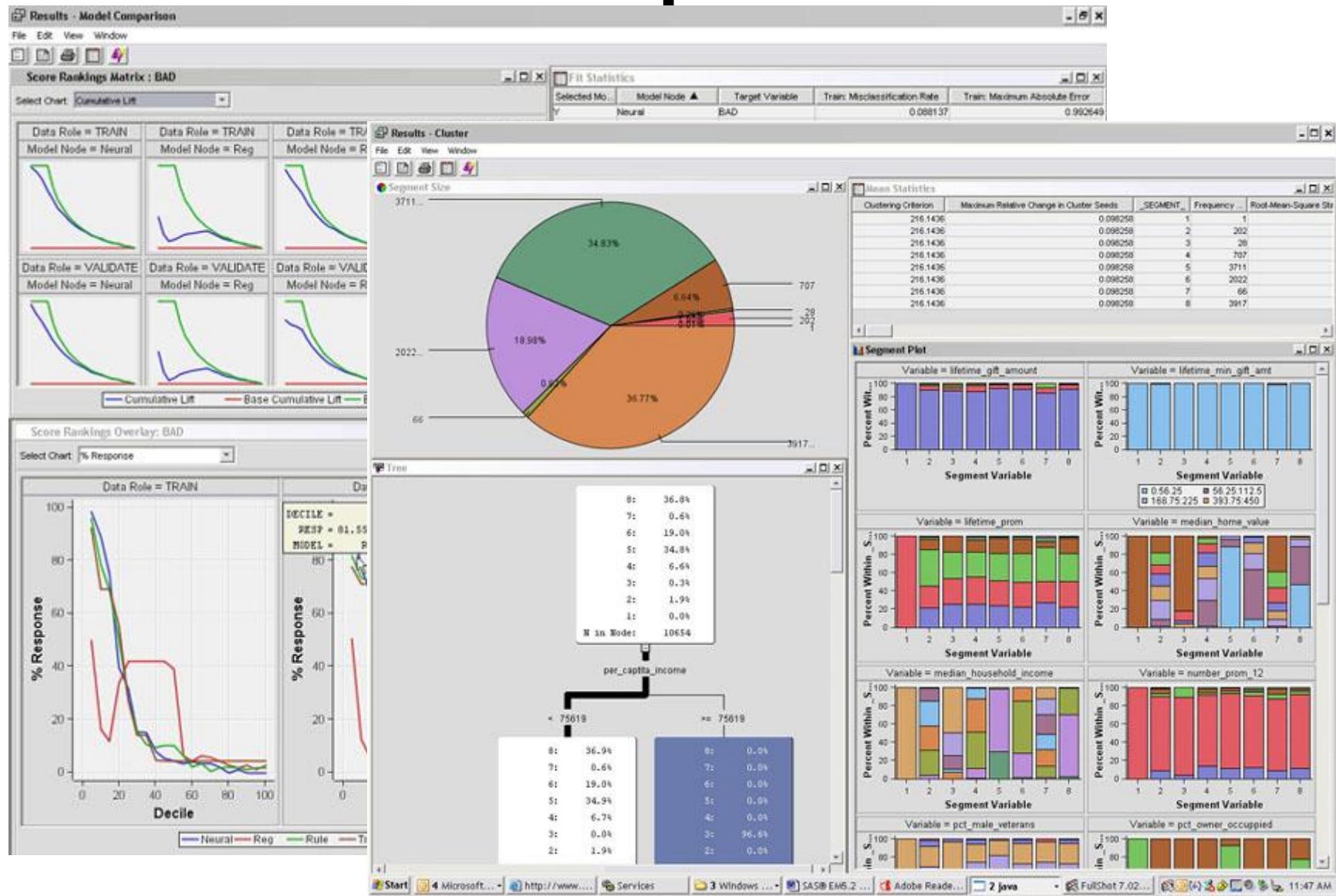
The main area contains several panes:

- Structure** and **Data** tabs. The Data tab shows a table with columns ID and AFFINITY_CARD, containing rows 101501 through 101508.
- Expression Editor** pane: Column Name: AGE_RESIDENCY_INDEX. Expression: 'MINING_DATA_BUILD"."AGE" * "MINING_DATA_BUILD"."YRS_RESIDENCE"'. Validation Results: Validation successful.
- Histogram for: AGE**: A bar chart showing Bin Count vs. Bin range. The distribution is roughly bell-shaped, peaking around 18-24 years.
- Statistics** table:

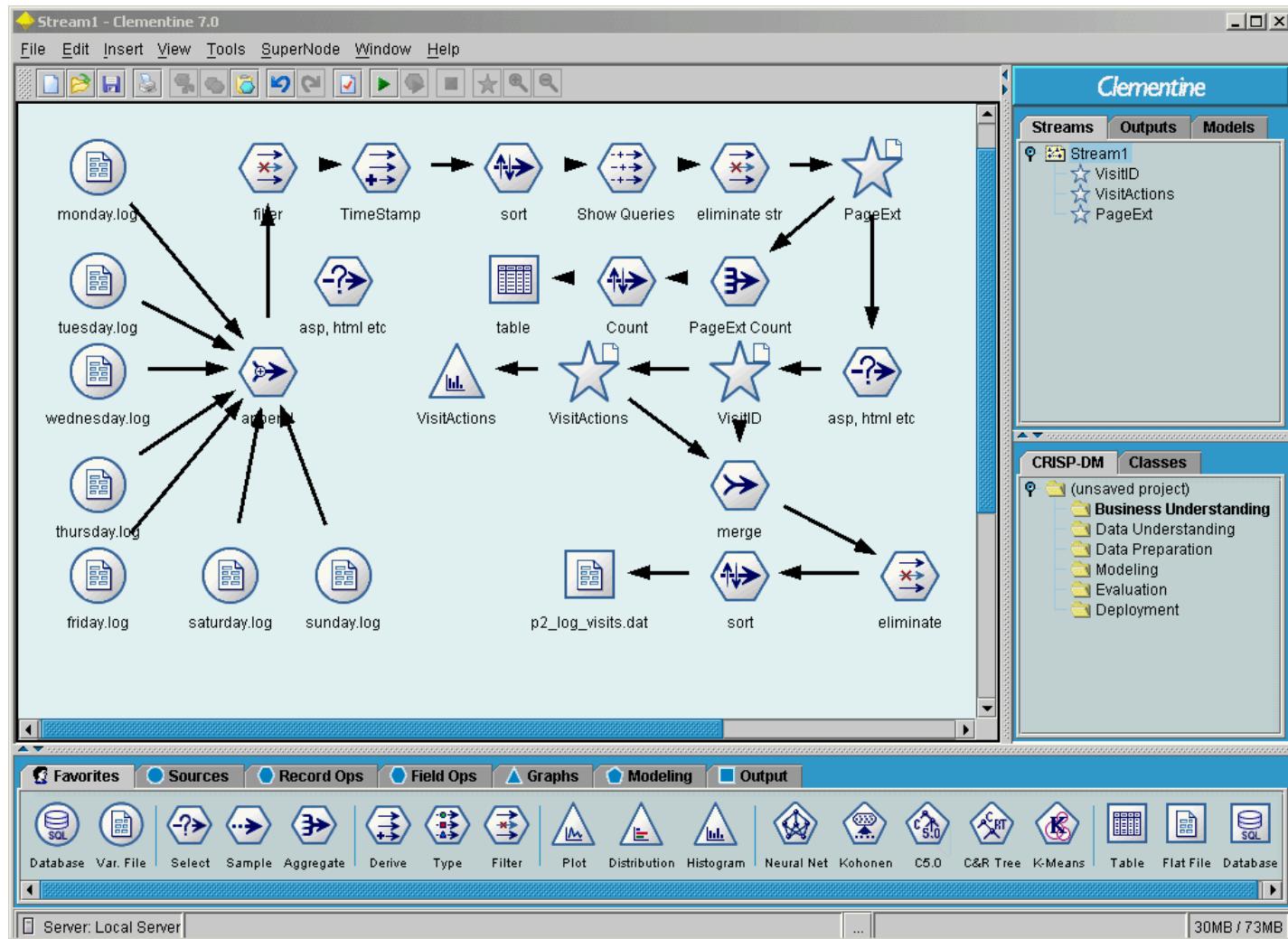
Group	Value(s)	Bin Count	% of Total
0	<= 24.3	161	16%
1	24.3 - 31.6	162	16%
2	31.6 - 38.9	198	20%
- Cumulative Target Chart**: A line graph showing Cumulative Target vs. Quantiles. The curve rises from 0.0 at quantile 0 to approximately 0.95 at quantile 10.
- Lift Table**: A table showing Quantiles, Cumulative Lift, and Cumulative Target.

Quantiles	Cumulative Lift	Cumulative Target
1	3.2650956	0.32650956
2	2.7601156	0.5520231
3	2.398644	0.7196532
4	2.0520232	0.82060925

SAS Enterprise Miner



SPSS Clementine



Criteria for DM System Deployment

- Assessment of the company
 - size, expertise, coaching quality, quality of consulting
- System architecture
 - distribution on multiple tiers, multi-processing support, database support
- Data exploration
 - visualization of value distribution, recognition of outliers, null values, etc.
- Data preparation
 - computation of new (derived) values, categorization of variables, sampling
- Data mining techniques support
 - parameterization, presentation of results
- Usability
 - ergonomics, quality of online help system
- Automation
 - sequential execution, vertical integration
 - (e.g. use of models in other applications)

Outlook: Trends in Data Mining

- **Vertical** solutions: in-depth integration of business logic into DM systems, e.g. CRM
- **Scalable** data mining methods, e.g. constraint-based mining to increase efficiency
- **Integration** with DBMS, data warehouses, and Web for portability and scalability
- **Privacy** protection and information **security** in DM
- **Standards**
 - OLE DB for DM (Microsoft)
 - PMML (Data Mining Group)
 - CRISP-SM (CRISP project)
 - SQL/MM Part 6 (ISO)
 - benchmarking & interoperability (The Analytic Solutions Forum)
 - privacy (CPExchange)
- **Web mining**
 - Web log mining
 - Web content mining
 - DM services over the Web

Summary

- Data Mining: one step in the Knowledge Discovery Process
- Data mining phases: training, test, application
- Several data mining techniques
- Many different algorithms exist that realize a specific technique
- Descriptive techniques:
 - Association rule discovery (Apriori)
 - Clustering/segmentation (k-means, k-mediods)
- Predictive techniques:
 - Classification (Decision Tree, C4.5)
 - Regression

Papers

- [AS96] R. Agrawal, J. C. Shafer: Parallel Mining of Association rules: Design, Implementation and Experience. IBM Research Report RJ 10004, 1996.
- [BF+84] L. Breiman, J. H. Friedman, R. A. Olshen, C. J. Stone: Classification and Regression Trees. Wadsworth 1984.
- [CC+00] P. Chapman, J. Clinton, R. Kerber, T. Khabaza, T. Reinartz, C. Shearer, R. Wirth: CRISP-DM 1.0, Step-by-step Data Mining Guide. SPSS Inc., August, 2000.
- [FP+96] U. Fayyad, G. Piatetsky-Shapiro, P. Smyth: The KDD Process for Extracting Useful Knowledge from Volumes of Data. CACM Vol. 39, No. 11, 1996.
- [Han99] D. J. Hand: Statistics and Data Mining: Intersecting Disciplines. SIGKDD Explorations, Vol. 1, No.1, June, 1999.
- [Mag94] J. Magidson: The CHAID approach to segmentation modeling. In: R. P. Bagozzi (ed.) Advanced Models of Marketing Research. Blackwell Business, 1994.



Papers

- [Qui86] J. R. Quinlan: Induction of Decision Trees. *Machine Learning*, Vol. 1, No. 1, 1986.
- [Qui93] Quinlan, J. R.: C4.5: Programs for MachineLearning. Morgan Kaufman, 1993.
- [RS04] Raileanu, L. E.; Stoffel, K.: Theoretical Comparison between the Gini Index and Information Gain Criteria. *Annals of Mathematics and Artificial Intelligence*, May 2004, vol. 41, no. 1, pp. 77-93(17)



www.kdnuggets.com
www.acm.org/sigkdd

Data-Warehouse-, Data-Mining- und OLAP-Technologien

Chapter 7: Database Support

Bernhard Mitschang
Universität Stuttgart

Winter Term 2015/2016

Overview

- Support for OLAP and Data Mining in SQL
 - OLAP:
ROLLUP, CUBE, WINDOW, OLAP Functions
 - Data Mining (SQL/MM)
- Database Support for OLAP
 - Partitioning
 - Materialized Views
 - Indexes
 - Optimization of OLAP Queries (Star Joins)
- Performance Evaluation: TPC Benchmarks

SQL:1999

- ISO/IEC 9075-x:1999 1999
 - Part 1: Framework (SQL/Framework)
 - Part 2: Foundation (SQL/Foundation)
 - Part 3: Call-Level Interface (SQL/CLI)
 - Part 4: Persistent Stored Modules (SQL/PSM)
 - Part 5: Host Language Bindings (SQL/Bindings)
- ISO/IEC 9075-x:2000 2000
 - Part 9: Management of External Data (SQL/MED)
 - Part 10: Object Language Bindings (SQL/OLB)
 - Part 13: SQL Routines and Types Using the Java TM Programming Language (SQL/JRT)
 - Amendment 1: On-Line Analytical Processing (SQL/OLAP)

(x = part number)

SQL:2003

• ISO/IEC 9075-x:2003	2003	pages
▪ Part 1: Framework (SQL/Framework)		72
▪ Part 2: Foundation (SQL/Foundation)		1245
▪ Part 3: Call-Level Interface (SQL/CLI)		395
▪ Part 4: Persistent Stored Modules (SQL/PSM)		172
▪ Part 5: Host Language Bindings (SQL/Bindings)		
▪ Part 9: Management of External Data (SQL/MED)		481
▪ Part 10: Object Language Bindings (SQL/OLB)		380
▪ new Part 11: Information and Definition Schemas (SQL/Schemata)		286
▪ Part 13: SQL Routines and Types Using the Java TM Programming Language (SQL/JRT)		192
▪ new Part 14: XML-Related Specifications (SQL/XML)		255
▪ Amendment 1: On-Line Analytical Processing (SQL/OLAP)		

SQL/MM

Multimedia and Application Packages

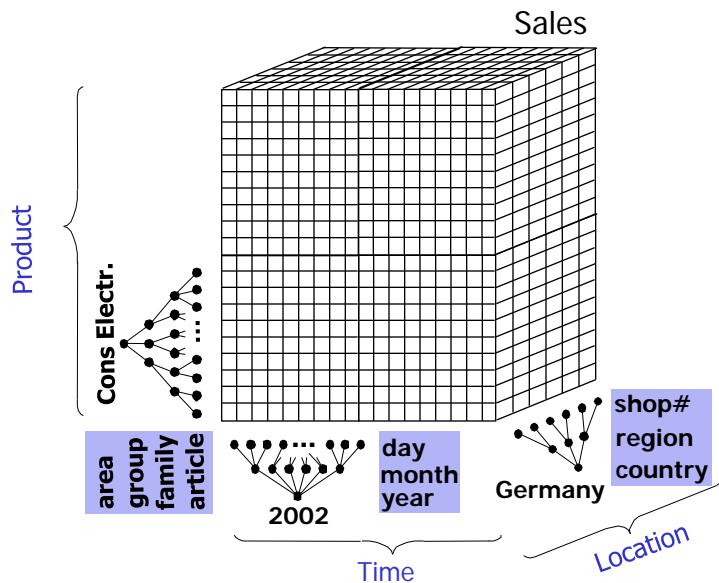
- ISO/IEC 13249-x:2003

- ISO/IEC 13249 defines a number of packages of generic data types common to various kinds of data used in multimedia and application areas, to enable that data to be stored and manipulated in an SQL database.
- Part 1: Framework 2002
- Part 2: Full-Text 2003
- Part 3: Spatial 2006
- Part 5: Still Image 2003
- Part 6: Data Mining 2006
- Part 7: History draft

SQL and Complex Data Analysis

- Data analysis features of SQL (even before SQL/OLAP):
 - aggregate functions
 - user-defined functions
 - GROUP BY and HAVING
 - CUBE and ROLLUP
- **But:** Early versions of SQL provide limited support for complex data analysis.
- Processing scenario:
 - retrieve data from the database into the client application environment
 - perform analysis on the client
 - overhead by retrieval of huge amounts of data

ROLLUP



Sales			
country	month	group	sales
F	01/04	100	500
F	01/04	200	1000
F	01/04	300	250
F	02/04	100	750
F	02/04	200	1250
F	02/04	300	2000
F	02/04	400	500
GB	01/04	100	400
GB	01/04	200	800
GB	01/04	300	300
GB	02/04	100	2000
GB	02/04	200	2500
GB	02/04	300	100
GB	02/04	400	100
GB	02/04	500	100
GB	02/04	600	100
I	01/04	200	250
I	01/04	300	750
I	01/04	400	200
I	01/04	500	1500
I	01/04	600	800

- Support for Roll-up (dimension reduction)
- Compute:
 - Sales per country, month and group,
 - Sales per country and month, and
 - Sales per country

GROUP BY country, month, group

GROUP BY country, month

GROUP BY country

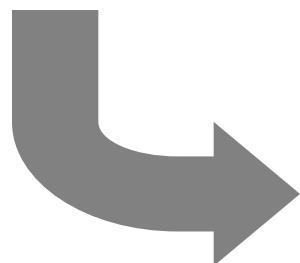
ROLLUP

query using grouping sets

```
SELECT country, month, group,
       SUM(sales) AS sales
  FROM Sales
 GROUP BY GROUPING SETS (
    (country, month group),
    (country, month),
    (country),
    () )
```

query using ROLLUP

```
SELECT country, month, group,
       SUM(sales) AS sales
  FROM Sales
 GROUP BY ROLLUP
    (country, month, group)
```



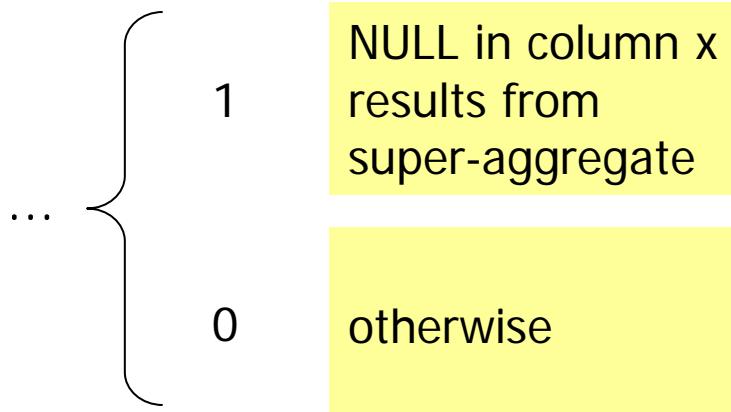
country	month	group	sales
F	01/04	100	500
F	01/04	200	1000
F	01/04	300	250
F	01/04	-	1750
F	02/04	100	750
F	02/04	200	1250
F	02/04	300	2000
F	02/04	400	500
F	02/04	-	4500
F	-	-	6250
GB	01/04	100	400
GB	01/04	200	800
GB	01/04	300	300
GB	01/04	-	1500
GB	02/04	100	2000
GB	02/04	200	2500
GB	02/04	300	100
GB	02/04	400	100
GB	02/04	500	100
GB	02/04	600	100
GB	02/04	-	4900
GB	-	-	6400
I	01/04	200	250
I	01/04	300	750
I	01/04	400	200
I	01/04	500	1500
I	01/04	600	800
I	01/04	-	3500
I	-	-	3500
-	-	-	16150

"-": NULL value

GROUPING Function

```
SELECT country, month, group,
       SUM(sales) AS sales,
       GROUPING (month) AS GM,
       GROUPING (group) AS GG
  FROM Sales
 GROUP BY ROLLUP
    (country, month, group)
```

GROUPING(x) = ...

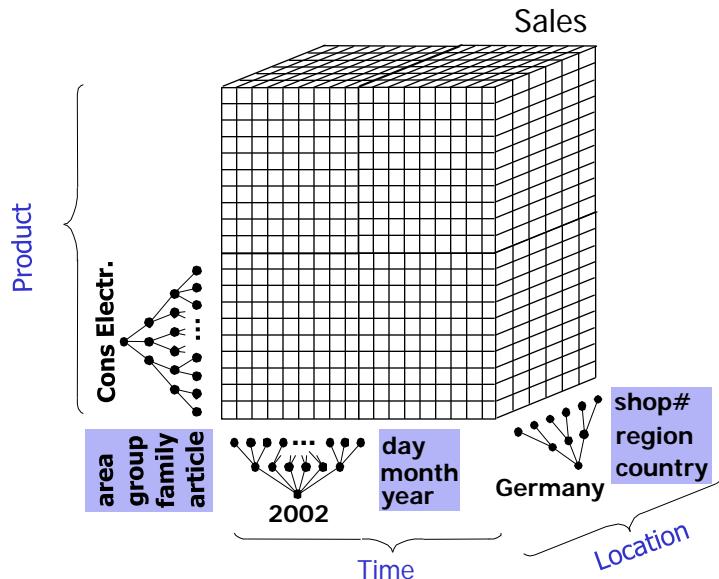


country	month	group	sales	GM	GG
F	01/04	100	500	0	0
F	01/04	200	1000	0	0
F	01/04	300	250	0	0
F	01/04	-	1750	0	1
F	02/04	100	750	0	0
F	02/04	200	1250	0	0
F	02/04	300	2000	0	0
F	02/04	400	500	0	0
F	02/04	-	4500	0	1
F	-	-	6250	1	1
GB	01/04	100	400	0	0
GB	01/04	200	800	0	0
GB	01/04	300	300	0	0
GB	01/04	-	1500		
GB	02/04	100	2000	0	0
GB	02/04	200	2500	0	0
GB	02/04	300	100	0	0
GB	02/04	400	100	0	0
GB	02/04	500	100	0	0
GB	02/04	600	100	0	0
GB	02/04	-	4900	0	1
GB	-	-	6400	1	1
I	01/04	200	250	0	0
I	01/04	300	750	0	0
I	01/04	400	200	0	0
I	01/04	500	1500	0	0
I	01/04	600	800	0	0
I	01/04	-	3500	0	1
I	-	-	3500	1	1
-	-	-	16150	1	1

Does this
NULL value
result from
ROLLUP?

"-": NULL value

CUBE



query using grouping sets

```
SELECT country, month, group, SUM(sales)
FROM Sales
GROUP BY GROUPING SETS (
    (country, month, group),
    (country, month),
    (country, group),
    (month, group),
    (country), (month), (group), () )
```

- Support for Roll-up (dimension reduction)
- Compute:
 - Sales for all aggregation groups of attributes country, month and group.

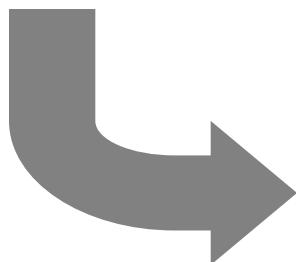


GROUP BY country, month, group
GROUP BY country, group
GROUP BY country, month
GROUP BY month, group
GROUP BY country
GROUP BY month
GROUP BY group

CUBE

equivalent query using CUBE

```
SELECT country, month, group,
       SUM(sales)
  FROM Sales
 GROUP BY CUBE
        (country, month, group)
```



country	month	group	sales
F	01/04	100	500
F	01/04	200	1000
F	01/04	300	250
F	01/04	-	1750
F	02/04	100	750
F	02/04	200	1250
F	02/04	300	2000
F	02/04	400	500
F	02/04	-	4500
F	-	-	6250
GB	01/04	100	400
GB	01/04	200	800
GB	01/04	300	300
GB	01/04	-	1500
GB	02/04	100	2000
GB	02/04	200	2500
GB	02/04	300	100
GB	02/04	400	100
GB	02/04	500	100
GB	02/04	600	100
GB	02/04	-	4900
GB	-	-	6400
I	01/04	200	250
I	01/04	300	750
I	01/04	400	200
I	01/04	500	1500
I	01/04	600	800
I	01/04	-	3500
I	-	-	3500
-	-	-	16150

"-": NULL value

country	month	group	sales
F	-	100	1250
F	-	200	2250
F	-	300	2250
F	-	400	500
GB	-	100	2400
GB	-	200	3300
GB	-	300	400
GB	-	400	100
GB	-	500	100
GB	-	600	100
I	-	200	250
I	-	300	750
I	-	400	200
I	-	500	1500
I	-	600	800
-	01/04	100	900
-	01/04	200	2050
-	01/04	300	1300
-	01/04	400	200
-	01/04	500	1500
-	01/04	600	800
-	02/04	100	2750
-	02/04	200	3750
-	02/04	300	2100
-	02/04	400	600
-	02/04	500	100
-	02/04	600	100
-	01/04	-	6750
-	02/04	-	9400
-	-	100	3650
-	-	200	5800
-	-	300	3400
-	-	400	800
-	-	500	1600
-	-	600	900

Syntax: Cube and Rollup

```
<group by clause> ::=  
    GROUP BY <grouping specification>
```

```
<grouping specification> ::=  
    <grouping column reference>  
    | <rollup list>  
    | <cube list>  
    | <grouping sets specification>  
    | <grand total>  
    | <concatenated grouping>
```

```
<rollup list> ::=  
    ROLLUP ( <grouping column reference list> )
```

```
<cube list> ::=  
    CUBE ( <grouping column reference list> )
```

```
<grouping sets specification> ::=  
    GROUPING SETS ( <grouping set list> )
```

```
<grouping set list> ::=  
    <grouping set> [ { <comma> <grouping set> } ... ]
```

```
<concatenated grouping> ::=  
    <grouping set> <comma> <grouping set list>
```

```
<grouping set> ::=  
    <ordinary grouping set>  
    | <rollup list>  
    | <cube list>  
    | <grand total>
```

```
<ordinary grouping set> ::=  
    <grouping column reference>  
    | ( <grouping column reference list> )
```

```
<grand total> ::=  
    ()
```

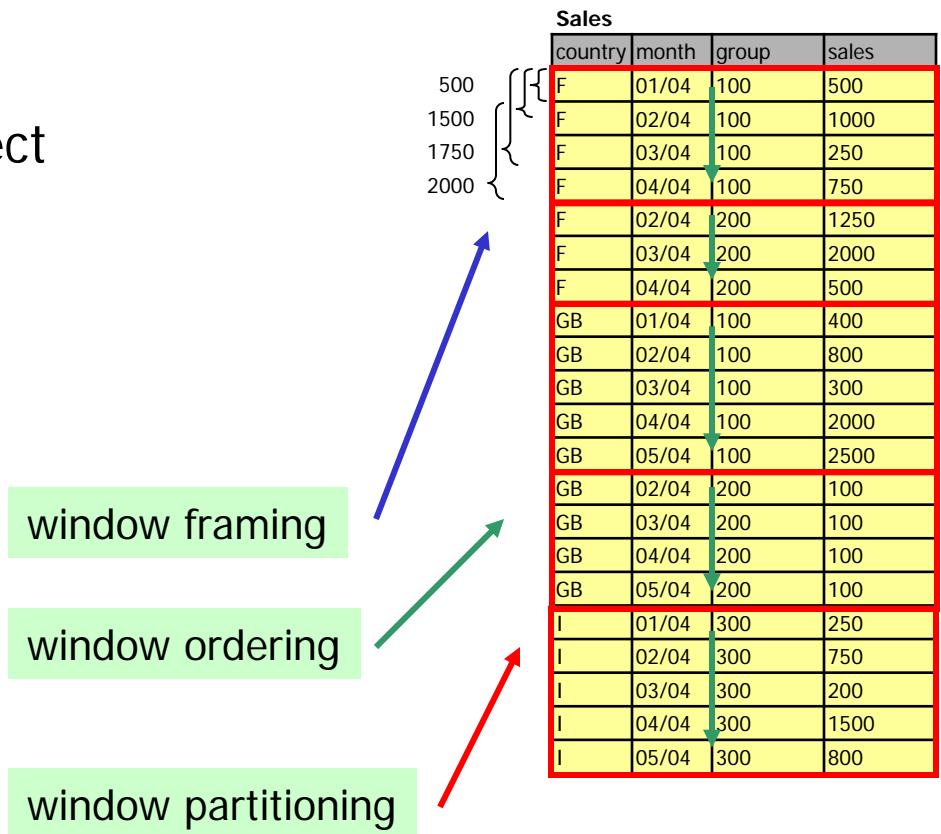
```
<grouping column reference list> ::=  
    <grouping column reference>  
    [ { <comma> <grouping column reference> } ... ]
```

```
<grouping column reference> ::=  
    <column reference> [ <collate clause> ]
```

WINDOW

- User-defined selection of rows within a query.
- Perform calculations with respect to the current row and the window defined.
- Example:

Calculate the sum of sales for each month and the two preceding months; provide the sum per country and group.



The diagram shows a table named 'Sales' with four columns: 'country', 'month', 'group', and 'sales'. The data is grouped by 'country' (F, GB, I) and ordered by 'month'. A vertical red line at month 04/04 defines the 'window framing'. A green arrow points from the 'window framing' text to this line. A blue arrow points from the 'window ordering' text to the 'month' column header. A red arrow points from the 'window partitioning' text to the 'country' column header. Brackets on the left side of the table indicate window frames for each country: F spans from month 01/04 to 04/04, GB spans from 01/04 to 05/04, and I spans from 01/04 to 04/04. The 'group' column values are 100 for F and 200 for GB, while the I group values are 300.

Sales			
country	month	group	sales
F	01/04	100	500
F	02/04	100	1000
F	03/04	100	250
F	04/04	100	750
F	02/04	200	1250
F	03/04	200	2000
F	04/04	200	500
GB	01/04	100	400
GB	02/04	100	800
GB	03/04	100	300
GB	04/04	100	2000
GB	05/04	100	2500
GB	02/04	200	100
GB	03/04	200	100
GB	04/04	200	100
GB	05/04	200	100
I	01/04	300	250
I	02/04	300	750
I	03/04	300	200
I	04/04	300	1500
I	05/04	300	800

WINDOW

```
SELECT country, month, group, sales,
       SUM(sales) OVER w AS moving_sum
```

```
FROM Sales AS s
```

```
WINDOW w AS (
```

```
    PARTITION BY s.country, s.group
```

```
    ORDER BY s.month
```

```
    ROWS 2 PRECEDING )
```

window partitioning

window ordering

window framing

```
SELECT country, month, group, sales,
       SUM(sales) OVER (
```

```
    PARTITION BY s.country,
                 s.group
```

```
    ORDER BY s.month
```

```
    ROWS 2 PRECEDING )
```

```
    AS moving_sum
```

```
FROM Sales AS s
```

in-line window specification



country	month	group	sales	moving_sum
F	01/04	100	500	500
F	02/04	100	1000	1500
F	03/04	100	250	1750
F	04/04	100	750	2000
F	02/04	200	1250	1250
F	03/04	200	2000	3250
F	04/04	200	500	3750
GB	01/04	100	400	400
GB	02/04	100	800	1200
GB	03/04	100	300	1500
GB	04/04	100	2000	3100
GB	05/04	100	2500	4800
GB	02/04	200	100	100
GB	03/04	200	100	200
GB	04/04	200	100	300
GB	05/04	200	100	300
I	01/04	300	250	250
I	02/04	300	750	1000
I	03/04	300	200	1200
I	04/04	300	1500	2450
I	05/04	300	800	2500

WINDOW Ordering

ORDER BY <sort specification> [{ <comma> <sort specification> } ...]

<sort specification> ::= =

<sort key> [<ordering specification>]

[<null ordering>]

ASC | DESC

NULLS FIRST | NULLS LAST

<ordering specification> ::=

<null ordering> ::=

- Specifies the ordering for each of the partitions.
- **NULLS FIRST**: All NULL values appear first in the ordering - regardless of whether you specify ASC or DESC
- similar for **NULLS LAST**
- <null ordering> is provided for WINDOW ordering only.
- Without <null ordering> clause: Null values are sorted as though they were less than all non-null values, or as they were greater than all non-null values. The choice is implementation-defined.
- Window ordering has no influence on the order of rows that are returned.

WINDOW Framing

```
<window frame units> <window frame extent> [ <window frame exclusion> ]
<window frame units> ::= ROWS | RANGE
<window frame extent> ::= <window frame start> | <window frame between>
<window frame start> ::= UNBOUNDED PRECEDING
    | <unsigned value> PRECEDING
    | CURRENT ROW
<window frame between> ::= BETWEEN      <window frame bound>
                           AND        <window frame bound>
<window frame bound> ::= <window frame start>
    | UNBOUNDED FOLLOWING
    | <unsigned value> FOLLOWING
<window frame exclusion> ::= EXCLUDE CURRENT ROW | EXCLUDE GROUP
    | EXCLUDE TIES | EXCLUDE NO OTHERS
```

- **ROWS**: physical aggregation groups
 - for input data that is "dense"
 - for multiple ordering columns
- **RANGE**: logical aggregation groups
 - for input data with gaps and ties (multiple rows for a given value)
 - for single ordering columns only

WINDOW Framing

```
... ORDER BY month      RANGE BETWEEN INTERVAL '1' MONTH PRECEDING  
                           AND INTERVAL '1' MONTH FOLLOWING  
                           /* logical aggregation group including the last, current, and next month */  
... ORDER BY month      ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING  
                           /* physical aggregation group on a dense column */  
  
... ROWS BETWEEN 5 PRECEDING AND CURRENT ROW  
... ROWS 5 PRECEDING  
                           /* preceding 5 rows and current row */  
... ROWS UNBOUNDED PRECEDING  
                           /* all preceding rows and current row */  
... ROWS BETWEEN CURRENT ROW AND UNBOUNDED FOLLOWING  
                           /* current row and all following rows */  
... ROWS BETWEEN 6 PRECEDING AND 3 PRECEDING  
                           /* the 3 rows starting 6 rows before the current row, current row is not  
                           included */
```

OLAP Functions: Numeric Value Functions

- **LN (x):**
Returns the natural logarithm of x. Exception if x is zero or negative.
- **EXP (x):**
Returns the value computed by raising the value of *e* (the base of natural logarithms) to the power specified by x.
- **POWER (x, y):**
Returns the value computed by raising x to the power of y.

x	y	result
0	0	1
0	>0	0
0	<0	exception
<0	no integer	exception

- **SQRT (x):**
Returns the square root of x.
= POWER (x, 0.5)
- **FLOOR (x):**
Returns the integer value nearest to positive infinity that is not greater than x.
- **CEILING (x):**
Returns the integer value nearest to negative infinity that is no less than x.
- **WIDTH_BUCKET (x, l, r, n):**
Returns a number indicating the partition into which x should be placed, if the range from l to r is divided into n equal-sized partitions.

OLAP Functions: Ranking

- **Sparse ranking:**

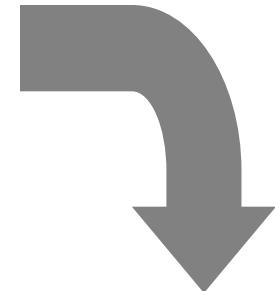
RANK returns a number that indicates the rank of the "current" row among the rows in the row's partition.

RANK provides a sparse ranking, i.e., gaps are possible.

- **Dense ranking:**

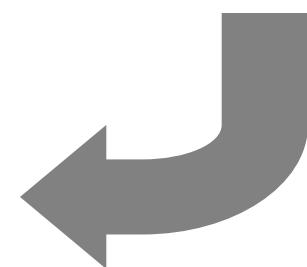
DENSE_RANK returns a ranking without gaps.

A	B
1	10
1	20
1	20
1	20
1	30
1	40



```
SELECT A, B,  
       RANK OVER  
        (PARTITION BY A  
         ORDER BY B ASC) AS r,  
       DENSE_RANK OVER  
        (PARTITION BY A  
         ORDER BY B DESC) AS dr,  
     FROM T
```

A	B	r	dr
1	10	1	4
1	20	2	3
1	20	2	3
1	30	5	2
1	40	6	1

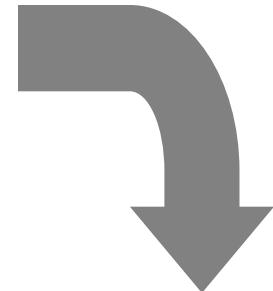


OLAP Functions: Ranking

- **Relative ranking:**

provided by PERCENT_RANK:
 $(\text{RANK of row in its partition} - 1) / (\text{number of rows in the partition} - 1)$

A	B
1	10
1	20
1	20
1	20
1	30
1	40



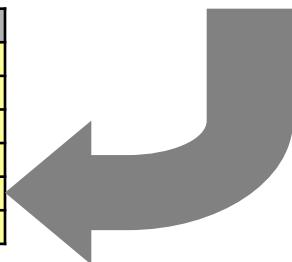
provided by CUME_DIST:
number of rows that precede or
peers with the row being examined
divided by the total number of rows
in the partition.

- **Row Number Function:**

ROW_NUMBER assigns a number to
each row, based on its position
within its window partition

```
SELECT A, B,  
PERCENT_RANK() OVER  
(PARTITION BY A  
ORDER BY B ASC) AS pr,  
CUME_DIST() OVER  
(PARTITION BY A  
ORDER BY B ASC) AS cr,  
ROW_NUMBER() OVER  
(PARTITION BY A  
ORDER BY B ASC) AS rn  
FROM T
```

A	B	pr	cr	rn
1	10	0	0.16	1
1	20	0.2	0.66	2
1	20	0.2	0.66	3
1	20	0.2	0.66	4
1	30	0.8	0.83	5
1	40	1	1	6



OLAP Functions: Ranking

- **Sparse ranking:**
RANK() OVER ws \Rightarrow (COUNT(*) OVER (ws RANGE UNBOUNDED PRECEDING) - COUNT(*) OVER (ws RANGE CURRENT ROW) + 1)
- **Dense ranking:**
DENSE_RANK()
OVER ws \Rightarrow (COUNT(DISTINCT ROW (ve1, ..., ven)) OVER (ws RANGE UNBOUNDED PRECEDING)
ve1, ..., ven: value expressions in the sort spec. of window ws
- **Relative ranking:**
PERCENT_RANK()
OVER ws \Rightarrow CASE
WHEN COUNT(*) OVER (ws RANGE BETWEEN UNBOUNDED PRECEDING
AND UNBOUNDED FOLLOWING) = 1
THEN CAST (0 AS ANT)
ELSE
(CAST (RANK () OVER (ws) AS ANT) -1) /
(COUNT(*) OVER (ws RANGE BETWEEN UNBOUNDED PRECEDING
AND UNBOUNDED FOLLOWING) -1)

- CUME_DIST()
OVER ws \Rightarrow (CAST (COUNT(*) OVER ws RANGE UNBOUNDED PRECEDING) AS ANT) /
COUNT(*) OVER (ws RANGE BETWEEN UNBOUNDED PRECEDING
AND UNBOUNDED FOLLOWING))
ANT: any numeric type
- **Row number:**
ROW_NUMBER
OVER ws \Rightarrow COUNT(*) OVER ws RANGE UNBOUNDED PRECEDING

Aggregate Functions and the Filter Clause

- **FILTER Clause:**

Allows to specify that the rows of each window partition or grouped table are filtered by a search condition before the specified aggregate function is computed.

country	count	c2	c3
F	7	2	0
GB	9	2	1



Sales			
country	month	group	sales
F	01/04	100	500
F	01/04	200	1000
F	01/04	300	250
F	02/04	100	750
F	02/04	200	1250
F	02/04	300	2000
F	02/04	400	500
GB	01/04	100	400
GB	01/04	200	800
GB	01/04	300	300
GB	02/04	100	2000
GB	02/04	200	2500
GB	02/04	300	100
GB	02/04	400	100
GB	02/04	500	100
GB	02/04	600	100
I	01/04	200	250
I	01/04	300	750
I	01/04	400	200
I	01/04	500	1500
I	01/04	600	800

```
SELECT country, COUNT(*) AS count,
       COUNT(*) FILTER (WHERE sales>1000) AS c2,
       COUNT(*) FILTER (WHERE group=600) AS c3
  FROM Sales
 GROUP BY country
 HAVING country <> 'I'
```

Other OLAP Functions

- **STDDEV_POP (value expression):**
Computes the population standard deviation of the provided *expression* evaluated for each row of the group or partition¹, defined as the square root of the population variance.
- **STDDEV_SAMP (value expression):**
Computes the sample standard deviation of the provided *expression* evaluated for each row of the group or partition¹, defined as the square root of the sample variance.
- **VAR_POP (value expression):**
Computes the population variance of *expression* evaluated for each row of the group or partition¹, defined as the sum of squares of the difference of *expression* from the mean of *expression*, divided by the number of rows in the group or partition.
- **VAR_SAMP (value expression):**
Computes the sample variance of *expression* evaluated for each row of the group or partition¹, defined as the sum of squares of the difference of *expression* from the mean of *expression*, divided by one less than the number of rows in the group or partition.

¹if DISTINCT was specified, each row that remains after duplicates have been eliminated.

Other OLAP Functions

- **COVAR_POP (expr1, expr2):**
Computes the population covariance.
- **COVAR_SAMP (expr1, expr2):**
Computes the sample covariance.
- **CORR (expr1, expr2):**
Computes the correlation coefficient.
- **REGR_SLOPE (expr1, expr2):**
Computes the slope of the least-squares-fit linear equation.
- **REGR_INTERCEPT (expr1, expr2):**
Computes the y-intercept of the least-squares-fit linear equation.
- **REGR_COUNT (expr1, expr2):**
Computes the number of rows remaining in the group or partition.
- **REGR_R2 (expr1, expr2):**
Computes the square of the correlation coefficient.
- **REGR_AVGX (expr1, expr2):**
Computes the average of *expr2* for all rows in the group or partition.
- **REGR_AVGY (expr1, expr2):**
Computes the average of *expr1* for all rows in the group or partition.
- **REGR_SXX (expr1, expr2):**
Computes the sum of squares of *expr2* for all rows in the group or partition.
- **REGR_SYY (expr1, expr2):**
Computes the sum of squares of *expr1* for all rows in the group or partition.
- **REGR_SXY (expr1, expr2):**
Computes the sum of products of *expr2* times *expr1* for all rows in the group or partition.

Overview

- Support for OLAP and Data Mining in SQL
 - OLAP:
ROLLUP, CUBE, WINDOW, OLAP Functions
 - Data Mining (SQL/MM)
- Database Support for OLAP
 - Partitioning
 - Materialized Views
 - Indexes
 - Optimization of OLAP Queries (Star Joins)
- Performance Evaluation: TPC Benchmarks

SQL/MM Part 6: Data Mining

- **SQL/MM: Multimedia and Application Packages**
Part 6 falls into the "applications packages" because it does not address "multimedia" data but deals with models and algorithms on data through the use of SQL:1999 user-defined types.
- Supported data mining techniques:
 - association rules
 - clustering
 - regression
 - classification
- SQL/MM Data Mining defines types and routines that allow to
 - specify data sets for mining tasks
 - specify mining tasks
 - derive, test, and apply data mining models
 - import and export mining models

Object-Oriented Extensions in SQL:1999

- **User-defined structured types**
 - Type hierarchy supported
 - Type-specific behaviour may be specified as methods
 - Structured types may be used in
 - Column definitions
 - Table definitions
 - View definitions
- **Typed tables and typed views**
 - Tables and views each of whose rows is an instance of a structured type
 - Table hierarchy or view hierarchy supported
 - Reference types allow navigational access

User-Defined Structured Types

- Example: Type hierarchy

```
CREATE TYPE part_t AS (
    partno      CHAR(4),
    version     VARCHAR(10),
    projectno   CHAR(4),
    description  VARCHAR(50))
```

```
REF USING INT
```

objects are identified by integer values

```
CREATE TYPE colored_part_t UNDER part_t AS (
```

```
    color_id    INT)
```

```
NOT FINAL
```

inherits all components of type part

Typed Tables

- **Example:** Table hierarchy based on a type hierarchy

```
CREATE TABLE parts OF part_t (
    partno      WITH OPTIONS NOT NULL,
    version      WITH OPTIONS NOT NULL,
    projectno   WITH OPTIONS NOT NULL,
REF IS oid USER GENERATED)
```

Define the self referencing column oid
 which contains in each row a value
 that uniquely identifies the row. In this
 case the identifier is provided by the user.

parts				
oid	partno	version	projectno	description
1	P050	1.0	PJ23	bodywork
...

column
constraints

```
CREATE TABLE colored_parts OF colored_part_t UNDER parts
INHERIT SELECT PRIVILEGES (
    color_id      WITH OPTIONS NOT NULL)
```

colored_parts					
oid	partno	version	projectno	description	color_id
10	P102	1.2	PJ23	a column	117
...

User-Defined Methods

- **Example:** Method definition

```
CREATE METHOD next_color (n INT)
RETURNS INT
FOR colored_part_t
RETURN SELF.color_id + n
```

- **Example:** Method usage

```
SELECT partno, color_id, DEREF(oid).next_color(1) AS next
FROM colored_parts
```

colored_parts					
oid	partno	version	projectno	description	color_id
10	P102	1.2	PJ23	a column	117
...	



partno	color_id	next
P102	117	118
...		

Data Mining Phases

- The **training phase** generates a data mining model.
- The **test phase** consumes a set of tuples carrying values in a special field. Each tuple is evaluated by the data mining model. Then, the predicted value of the special field is compared to the actual value.
- During the **application phase** each given tuple is evaluated by the model and one or more values are computed (cluster-ID for clustering, a predicted value for classification or regression).

Data mining technique	Training phase	Test phase	Application phase
Classification	✓	✓	✓
Regression	✓	✓	✓
Clustering	✓	✗	✓
Association rule discovery	✓	✗	✗

Data Mining Types

Types		Category	Purpose
DM_RuleModel DM_ClusteringModel	DM_RegressionModel DM_ClasModel	Data mining models	Represents models that are the result of a data mining step
DM_RuleSettings DM_ClusSettings	DM_RegSettings DM_ClasSettings	Data mining settings	Describes the settings that are used to generate a model
DM_RuleTask DM_ClusTask	DM_RegTask DM_ClasTask	Data mining settings	Represents the information about a model generation task
DM_ClusResult DM_RegResult	DM_ClasResult	Data mining application results	Describes the results of an application run of a data mining model
DM_RegTestResult DM_ClasTestResult		Data mining test results	Describes the results of a test run of a data mining model
DM_LogicalDataSpec DM_MiningData DM_ApplicationData		Data mining data	Describes the data sets for training, testing, and application

DM_ClasModel Type and Routines

```
CREATE TYPE DM_ClasModel AS (
    DM_content CHARACTER LARGE
    OBJECT(DM_MaxContentLength))
INSTANTIABLE
NOT FINAL
STATIC METHOD DM_impClasModel
    (input CHARACTER LARGE OBJECT
        (DM_MaxContentLength))
RETURNS DM_ClasModel
LANGUAGE SQL
DETERMINISTIC
CONTAINS SQL
RETURNS NULL ON NULL INPUT,
METHOD DM_expClasModel()
RETURNS CHARACTER LARGE
OBJECT(DM_MaxContentLength)
LANGUAGE SQL
DETERMINISTIC
CONTAINS SQL,
METHOD DM_clasGetTask()
RETURNS DM_ClasTask
LANGUAGE SQL
DETERMINISTIC
CONTAINS SQL,
```

```
METHOD DM_clasCostRate()
RETURNS DOUBLE
LANGUAGE SQL
DETERMINISTIC
CONTAINS SQL,
METHOD DM_clasModelSchema()
RETURNS DM_MiningSchema
LANGUAGE SQL
DETERMINISTIC
CONTAINS SQL,
METHOD DM_applyClasModel
    (input DM_ApplicationData)
RETURNS DM_ClasResult
LANGUAGE SQL
DETERMINISTIC
CONTAINS SQL
RETURNS NULL ON NULL INPUT,
METHOD DM_testClasModel (input DM_MiningData)
RETURNS DM_ClasTestResult
LANGUAGE SQL
DETERMINISTIC
CONTAINS SQL
RETURNS NULL ON NULL INPUT
```

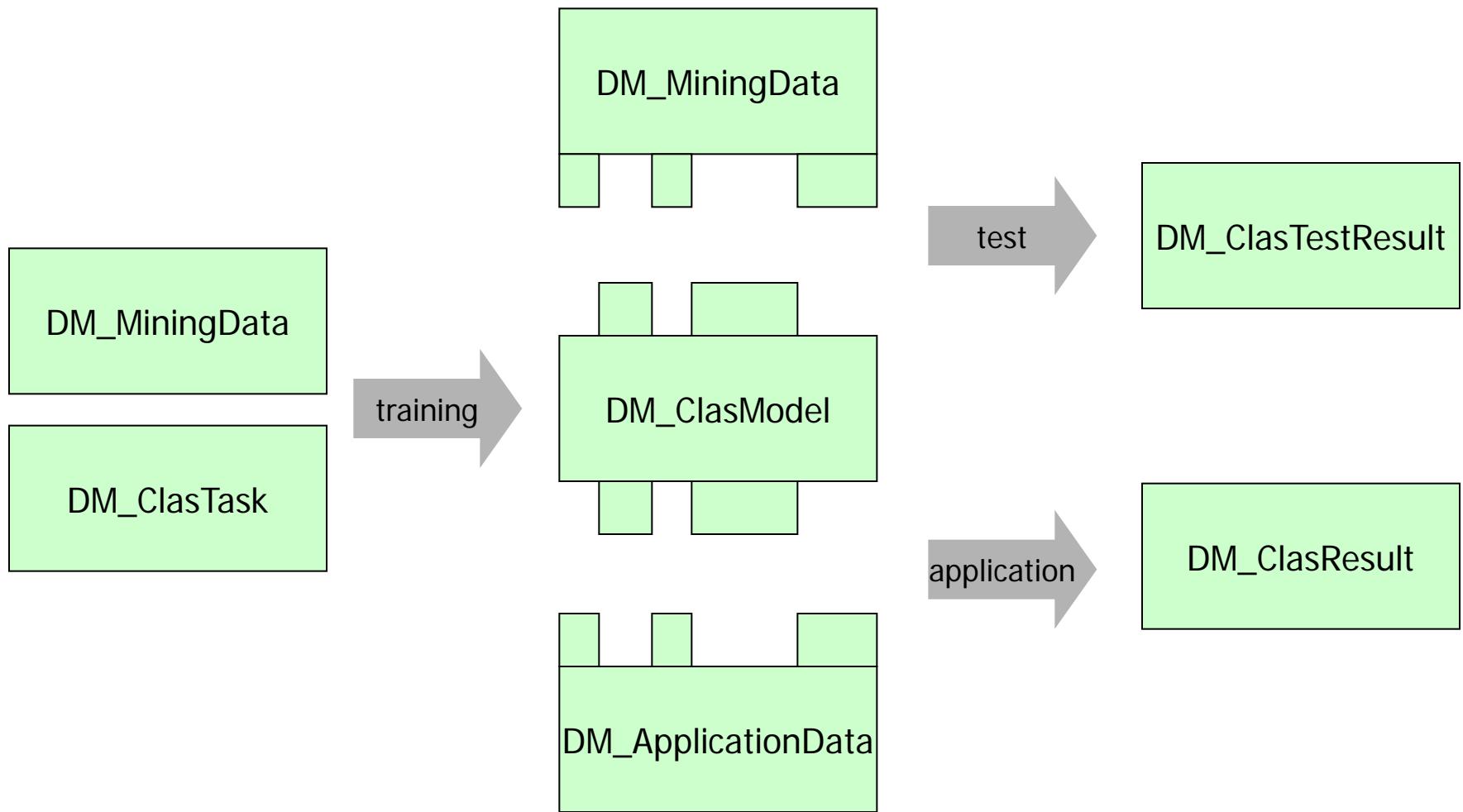
Sample Classification

Application scenario

- Given: A customer table CT of an insurance company with
 - columns C1, C2, ..., C9:
contain characteristics of customers
 - column r:
contains the risk class of each customer
(e.g. 'low', 'medium', 'high')
- Objectives:
 - Compute a classification model that allows to predict the risk for a new customer.
 - It should be possible to re-compute the classification model from time to time to see whether the model changes over time.
 - It must be possible to test a classification model with a given test table TT also containing columns C1, C2, ..., C9 and r.

CT				
C1	C2	...	C9	r

Classification: Phases and Types



Create a Mining Task

```
WITH MyData AS (DM_MiningData::DM_defMiningData('CT'))  
INSERT INTO MT (ID, TASK)  
VALUES (1, DM_ClasTask::DM_defClasTask( MyData, NULL,  
    ( new DM_ClasSettings()).DM_clasUseSchema(MyData.DM_genMiningSchema())  
    ).DM_clasSetTarget('r')  
    )  
)
```

- The DM_ClasTask value has to be stored to be able to run the data mining training several times. Table MT is used for this purpose.
- Main steps:
 - Create a DM_MiningData value using the static method DM_defMiningData.
 - Create a DM_MiningSchema value using the method DM_genMiningSchema of the DM_MiningData value.
 - Create a DM_ClasSettings value using the default constructor DM_ClasSettings and assign the DM_MiningSchema value as the schema to use.
 - Declare the column named 'r' as the predicted field using the DM_clasSetTarget method.
 - Create a DM_ClasTask value using the DM_defClasTask method.
 - Store the newly created DM_ClasTask value in table MT.

Training and Test Phase

```
INSERT INTO MM (ID, MODEL)  
VALUES (  
    1, MyTask.DM_buildClasModel()  
)
```

```
SELECT MODEL.DM_testClasModel(  
    DM_MiningData:::DM_defMiningData('TT'))  
FROM MM  
WHERE ID = 1
```

- Now that the DM_ClasTask value is generated and stored in the MT table, the classification training can be initiated and the classification model is computed.
- Since the model shall be used in later application and test runs, it is stored in a table MM having two columns ID of type integer and MODEL of type DM_ClasModel.

- A simple test of the model can be done using the data in table TT.
- The result of the test run is of type DM_ClasTestResult.
- DM_ClasTestResult provides a method DM_getClasError() that allows to derive the number of false classifications computed during a test of a classification model.

Application Phase

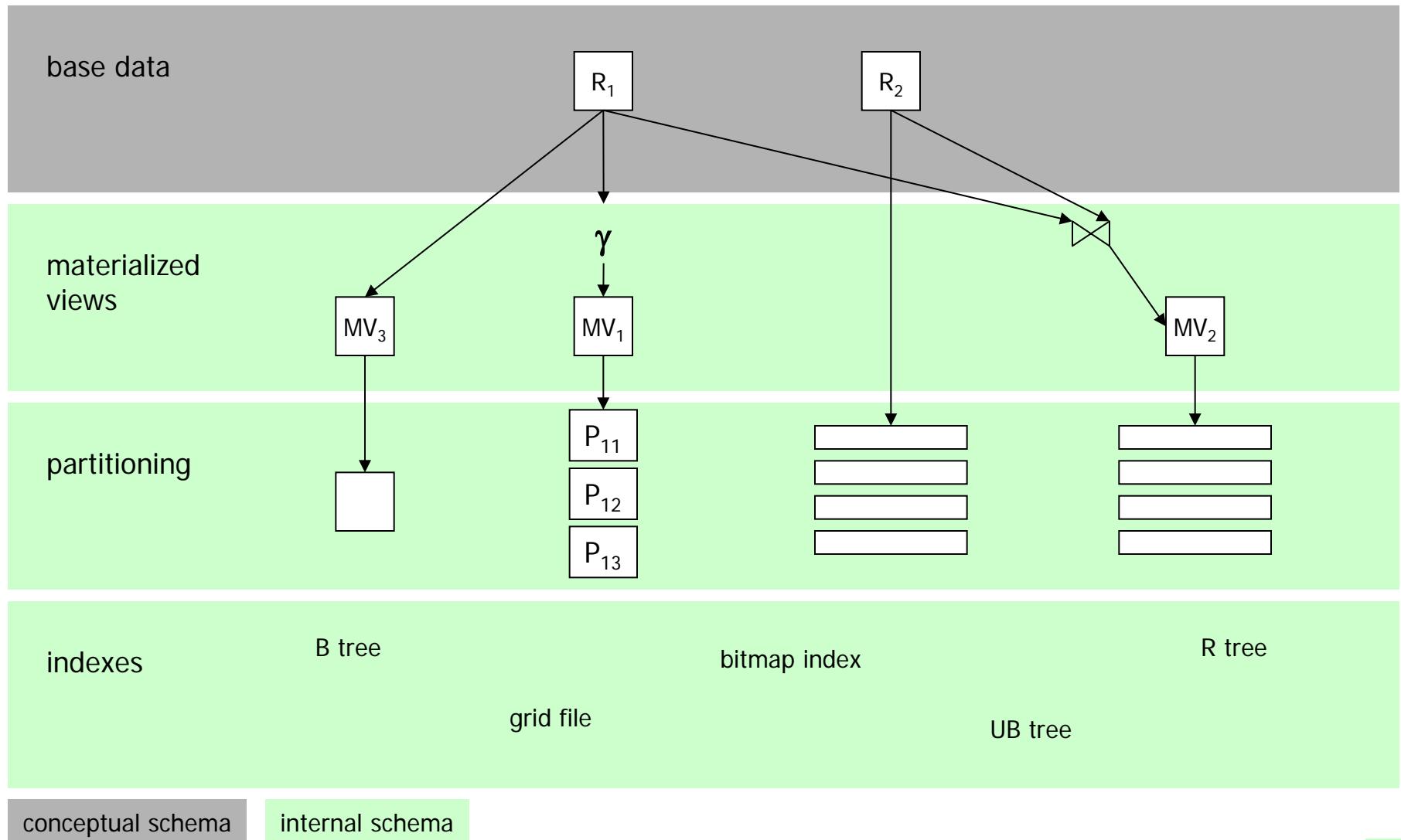
```
WITH MyModel AS (
    SELECT MODEL FROM MM
    WHERE ID = 1
)
SELECT C1, C2, C3, C4, C5, C6, C7, C8, C9, MyModel.DM_applyClasModel(
    DM_ApplicationData::DM_impApplData(
        '<C1>' + C1 + '</C1>' + '<C2>' + C2 + '</C2>' +
        '<C3>' + C3 + '</C3>' + '<C4>' + C4 + '</C4>' +
        '<C5>' + C5 + '</C5>' + '<C6>' + C6 + '</C6>' +
        '<C7>' + C7 + '</C7>' + '<C8>' + C8 + '</C8>' +
        '<C9>' + C9 + '</C9>'
    )
)
FROM CT
```

- In this example the table used to train the model is again used for an application of the model.
- Method DM_applyClasModel returns the result of applying the classification model contained in MyModel to a given value of DM_ApplicationData retrieved from table CT.

Overview

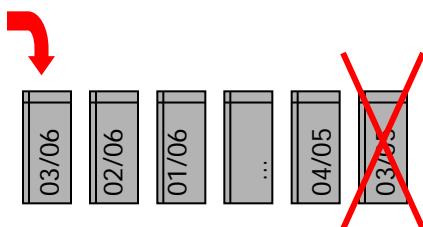
- Support for OLAP and Data Mining in SQL
 - OLAP:
ROLLUP, CUBE, WINDOW, OLAP Functions
 - Data Mining (SQL/MM)
- Database Support for OLAP
 - Partitioning
 - Materialized Views
 - Indexes
 - Optimization of OLAP Queries (Star Joins)
- Performance Evaluation: TPC Benchmarks

Levels of Database Support



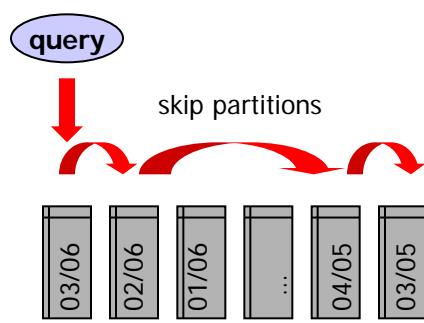
Partitioning

- A table is split into several partitions.
- The content of the table is provided by combining the content of all partitions.
- Support for data warehousing:



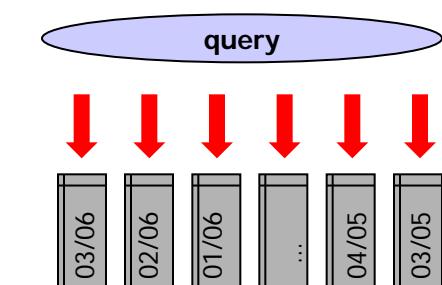
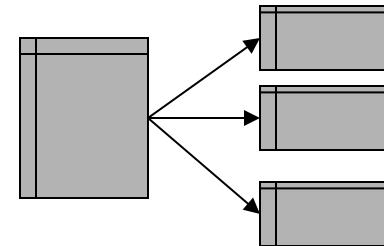
management of
huge data sets

- rolling data warehouse scenario
- add and drop partitions
- add table as partition



efficient query
processing

- partition pruning

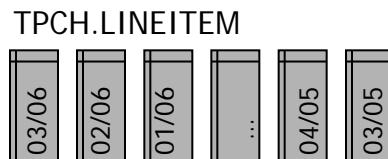


intra-query-parallelism

- parallel hash joins

Partitioning Methods

- horizontal partitioning vs. vertical partitioning
- range partitioning:
 - each row is assigned to a certain partition according to the value of its partitioning attribute



```
CREATE TABLE TPCH.LINEITEM ( L_ORDERKEY INTEGER NOT NULL,  
                            L_PARTKEY    INTEGER NOT NULL,  
                            L_SUPPKEY    INTEGER NOT NULL,  
                            ...  
                            L_SHIPDATE   DATE NOT NULL,  
                            ...  
PARTITION BY RANGE (L_SHIPDATE)  
( PARTITION LINEITEM032005 VALUES LESS THAN '2005-04-01'),  
( PARTITION LINEITEM042005 VALUES LESS THAN '2005-05-01'),  
( PARTITION LINEITEM052005 VALUES LESS THAN '2005-06-01'),  
( PARTITION LINEITEM062005 VALUES LESS THAN '2005-07-01'),  
...  
(Oracle)
```

Partitioning Methods

- hash partitioning:
 - each row is assigned to a certain partition according to the value of a linear hash function applied to the partitioning attribute(s) x:
$$h(x) := x \bmod p$$

p: number of partitions
- combinations:
 - e.g., range partitioning on the first level and hash partitioning on the second level

```
CREATE TABLE TPCH.LINEITEM ( L_ORDERKEY INTEGER NOT NULL,  
                           L_PARTKEY  INTEGER NOT NULL,  
                           L_SUPPKEY  INTEGER NOT NULL,  
                           ...  
                           L_SHIPDATE DATE NOT NULL,  
                           ...  
PARTITION BY HASH (L_ORDERKEY, L_PARTKEY, L_SUPPKEY) PARTITIONS 4
```

...

(Oracle)

Overview

- Support for OLAP and Data Mining in SQL
 - OLAP:
ROLLUP, CUBE, WINDOW, OLAP Functions
 - Data Mining (SQL/MM)
- Database Support for OLAP
 - Partitioning
 - Materialized Views
 - Indexes
 - Optimization of OLAP Queries (Star Joins)
- Performance Evaluation: TPC Benchmarks

Materialized Summary Data

- Detailed data as well as summarized data are physically stored in the data warehouse.
- Advantages:
 - Performance enhancement for queries that access summarized data.
- Challenges:
 - How to identify the suitable set of materialized summary data?
 - How to efficiently refresh materialized summary data?
 - How to use summary data implicitly?

Sales

country	month	group	sales
F	01/04	100	500
F	02/04	100	1000
F	03/04	100	250
F	04/04	100	750
F	05/04	200	1250
F	02/04	200	2000
F	03/04	200	500
GB	01/04	100	400
GB	02/04	100	800
GB	03/04	100	300
GB	04/04	100	2000
GB	05/04	100	2500
GB	02/04	200	100
GB	03/04	200	100
GB	04/04	200	100
GB	05/04	200	100
I	01/04	300	250
I	02/04	300	750
I	03/04	300	200
I	04/04	300	1500
I	05/04	300	800

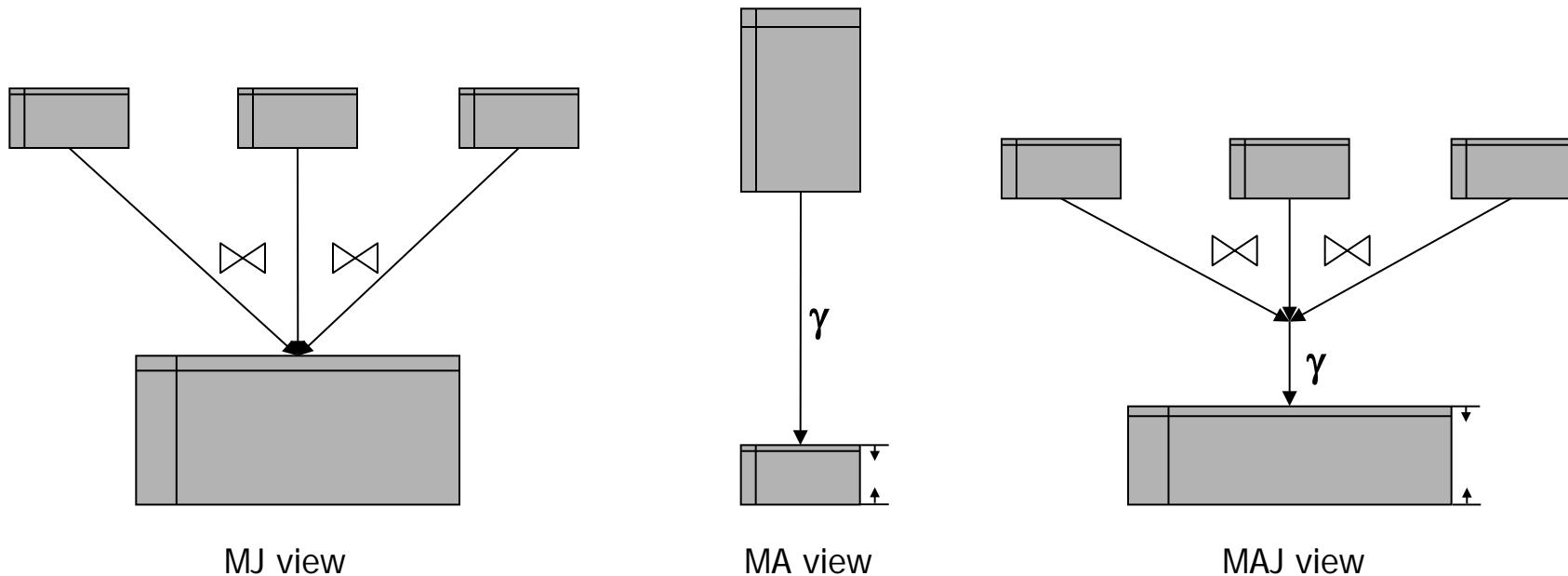
Sales_per_country

country	sales
F	6250
GB	6400
I	3500



Types of Summary Data

- Classification based on operations



- Other important characteristics:
 - Linkage between detailed data and summary data
 - Responsibility for refreshing summary data
 - Usage of summary data in query processing

Explicit vs. Implicit Summary Data

Explicit Summary Data

- Provided as additional tables
 - no link to detailed data
- Refreshing summary data:
 - part of ETL processing
- Usage:
 - Applications include queries that:
 - refer to detailed data
 - refer to summary data

Implicit Summary Data

- Provided by materialized views
 - materialized view is linked to detailed data
- Refreshing summary data:
 - part of updates on detailed data
 - DB system is responsible for refreshing
- Usage:
 - Applications include queries that:
 - always refer to detailed data
 - DB system automatically rewrites the query to allow usage of summary data.

Explicit Summary Data

```
SELECT month, SUM(sales)  
FROM Sales  
WHERE group = 100  
GROUP BY month
```



Sales			
country	month	group	sales
F	01/04	100	500
F	02/04	100	1000
F	03/04	100	250
F	04/04	100	750
F	05/04	200	1250
F	02/04	200	2000
F	03/04	200	500
GB	01/04	100	400
GB	02/04	100	800
GB	03/04	100	300
GB	04/04	100	2000
GB	05/04	100	2500
GB	02/04	200	100
GB	03/04	200	100
GB	04/04	200	100
GB	05/04	200	100
I	01/04	300	250
I	02/04	300	750
I	03/04	300	200
I	04/04	300	1500
I	05/04	300	800

ETL
processing

```
SELECT country, SUM(sales)  
FROM Sales_per_country  
WHERE country<>'I'  
GROUP BY country
```



query
summary
data

Sales_per_country	
country	sales
F	6250
GB	6400
I	3500

Explicit Summary Data

Fact table contains materialized summary data

Sales

region_dim	region_type	time_dim	time_type	customer_dim	customer_type	sales
50	1	401	3	10501	1	500
50	1	402	3	10501	1	1000
50	1	403	3	10501	1	250
50	1	404	3	10501	1	750
50	1	402	3	10502	1	1250
50	1	403	3	10502	1	2000
50	1	405	3	10502	1	500
51	1	401	3	10501	1	400
51	1	402	3	10501	1	800
51	1	403	3	10501	1	300
51	1	404	3	10501	1	2000
51	1	405	3	10501	1	2500
51	1	402	3	10502	1	100
51	1	403	3	10502	1	100
51	1	404	3	10502	1	100
51	1	405	3	10502	1	100
53	1	401	3	10505	1	250
53	1	402	3	10505	1	750
53	1	403	3	10505	1	200
53	1	404	3	10505	1	1500
53	1	405	3	10505	1	800
50	1	2004	4	100	2	6250
51	1	2004	4	100	2	6400
53	1	2004	4	100	2	3500

Region Dimension

region_type	country_id	country_desc	region_id	region_desc
1	50	F	70	EU
1	51	GB	70	EU
1	53	I	70	EU
2	-	-	70	EU
...				

region dimension
3: all
2: region
1: country

Time Dimension

time_type	month_id	month_desc	year_id	year_desc	...
...					
3	401	01/04	2004	2004	
3	402	02/04	2004	2004	
3	403	03/04	2004	2004	
3	404	04/04	2004	2004	
3	405	05/04	2004	2004	
...					
4	-	-	2004	2004	
...					

time dimension
4: year
3: month
2: week
1: day

Customer Dimension

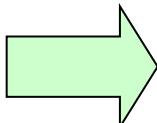
customer_type	customer_id	customer_desc	group_id	group_desc	...
...					
1	10501	name1	100	key cust.	
1	10502	name2	100	key cust.	
1	10505	name5	100	key cust.	
...					
2	-	-	100	key cust.	
...					

customer dimension
3: all
2: group
1: customer

Explicit Summary Data

```

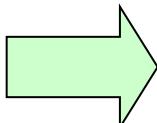
SELECT T.month_desc, SUM(S.sales)
FROM Sales S, Time T
WHERE S.customer_dim = 10501
AND S.region_type = 1
AND S.time_type = 3
AND S.customer_type = 1
AND S.time_type = T.time_type
AND S.time_dim = T.month_id
GROUP BY T.month_desc
    
```



month_desc	sales
01/04	900
02/04	1800
03/04	550
04/04	2750
05/04	2500

```

SELECT R.country_desc, T.year_desc, C.group_desc, SUM(S.sales)
FROM Sales S, Region R, Time T, Customer C
WHERE R.country_desc<>'I'
AND S.region_type = 1
AND S.time_type = 4
AND S.customer_type = 2
AND S.region_type = R.region_type
AND S.region_dim = R.country_id
AND S.time_type = T.time_type
AND S.time_dim = T.year_id
AND S.customer_type = C.customer_type
AND S.customer_dim = C.group_id
GROUP BY R.country_desc, T.year_desc, C.group_desc
    
```



country_desc	year_desc	group_desc	sales
F	2004	key cust.	6250
GB	2004	key cust.	6400

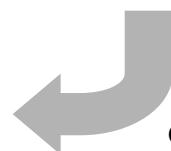
Implicit Summary Data

```
SELECT month, SUM(sales)  
FROM Sales  
WHERE group = 100  
GROUP BY month
```



Sales			
country	month	group	sales
F	01/04	100	500
F	02/04	100	1000
F	03/04	100	250
F	04/04	100	750
F	05/04	200	1250
F	02/04	200	2000
F	03/04	200	500
GB	01/04	100	400
GB	02/04	100	800
GB	03/04	100	300
GB	04/04	100	2000
GB	05/04	100	2500
GB	02/04	200	100
GB	03/04	200	100
GB	04/04	200	100
GB	05/04	200	100
I	01/04	300	250
I	02/04	300	750
I	03/04	300	200
I	04/04	300	1500
I	05/04	300	800

```
SELECT country, SUM(sales)  
FROM Sales  
WHERE country<>'I'  
GROUP BY country
```



query detailed data



db system internally uses summary



automatic refresh

Sales_per_country	
country	sales
F	6250
GB	6400
I	3500

Derivability

- **Problem statement:**
Given a query Q and a set of summary data M . Under which conditions is the result of Q derivable from M ?
- **Aspects of derivability:**
 - predicates
 - aggregate functions
 - base tables
 - groupings
- Knowledge on derivability is used in two ways:
 - processing of query Q : decide whether M can be used or not
 - selecting the most appropriate summary data

Derivability and Predicates

- P_Q predicate of query Q, P_M predicate of summary data M
- Conditions for derivability:
 - all attributes of P_Q are provided by M
 - $P_Q \subseteq P_M$
- Example:

Query

```
SELECT country, SUM(sales)  
  
FROM Sales  
  
WHERE country<>'I'  
  
GROUP BY country
```

Summary Data

```
CREATE VIEW Sales_per_country_month  
  
SELECT country, month, SUM(sales) AS sales  
  
FROM Sales  
  
GROUP BY country, month
```

Classification of Aggregate Functions

Additive aggregate functions:

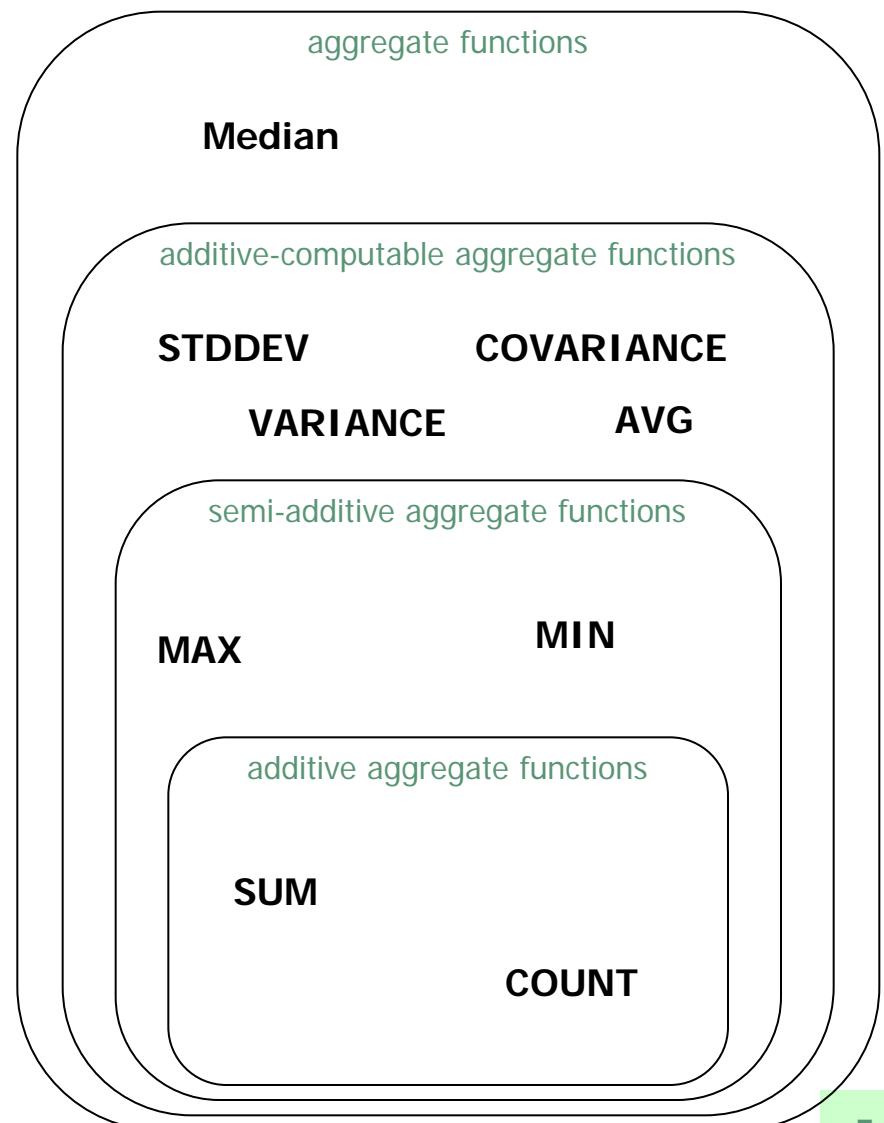
- Conditions:
 - $F(x_1 \cup x_2) = F(F(x_1), F(x_2))$
 - $F^{-1}()$ exists

Semi-additive aggregate functions:

- Conditions:
 - $F(x_1 \cup x_2) = F(F(x_1), F(x_2))$

Additive-computable functions:

- Conditions:
 - $F(x) = G(F_1(x), \dots, F_n(x))$
where G is a algebraic expression
and F_i are additive or semi-additive aggregate functions

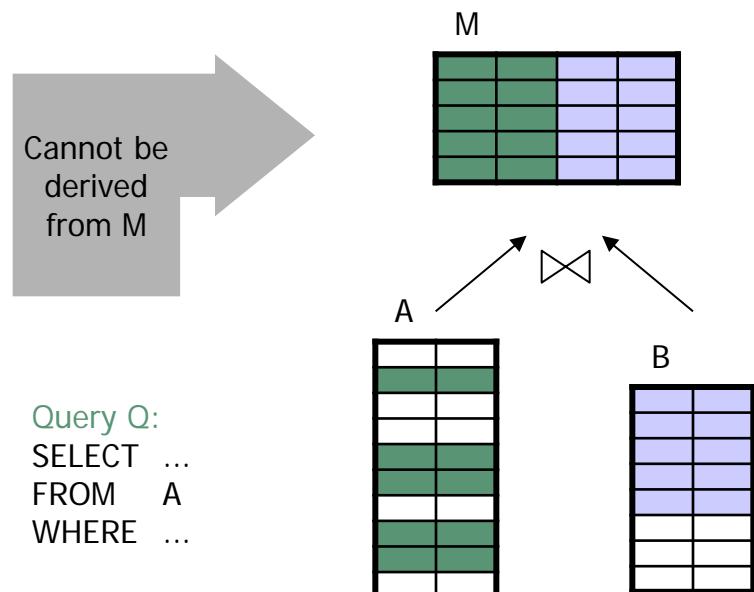


Derivability and Aggregate Functions

- Additive aggregate function:
 - is used in query Q and in summary data M
 - Q is derivable from M
- Semi-additive aggregate function:
 - is used in query Q and in summary data M
 - Q is derivable from M
 - incremental refreshing of M not possible for DELETE operations
- Additive-computable aggregate functions:
 - is used in query Q and based on additive or semi-additive functions used in M
 - Q is derivable from M
 - incremental refreshing of M not possible for DELETE operations
- All other aggregate functions:
 - no derivability

Derivability and Base Tables

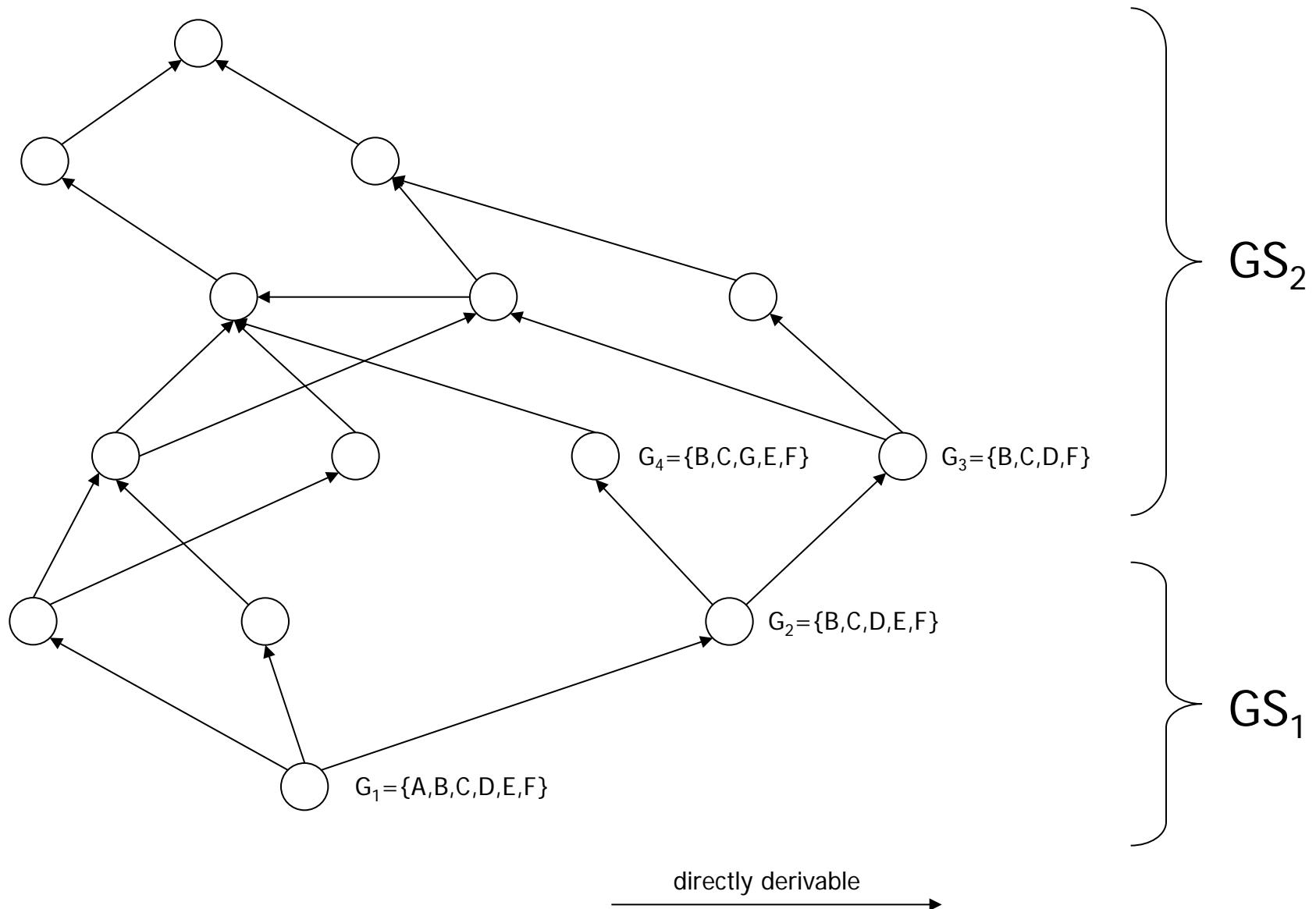
- Summary data M includes a join on tables A and B and a join predicate P_{AB} .
- Condition for derivability:
 - Query Q is derivable from M if Q also includes a join on tables A and B and the join predicate P_{AB} .
- Condition can be relaxed:
 - based on assertions on the data warehouse schema
 - e.g., if A and B build a lossless join, B is not necessary in query Q.



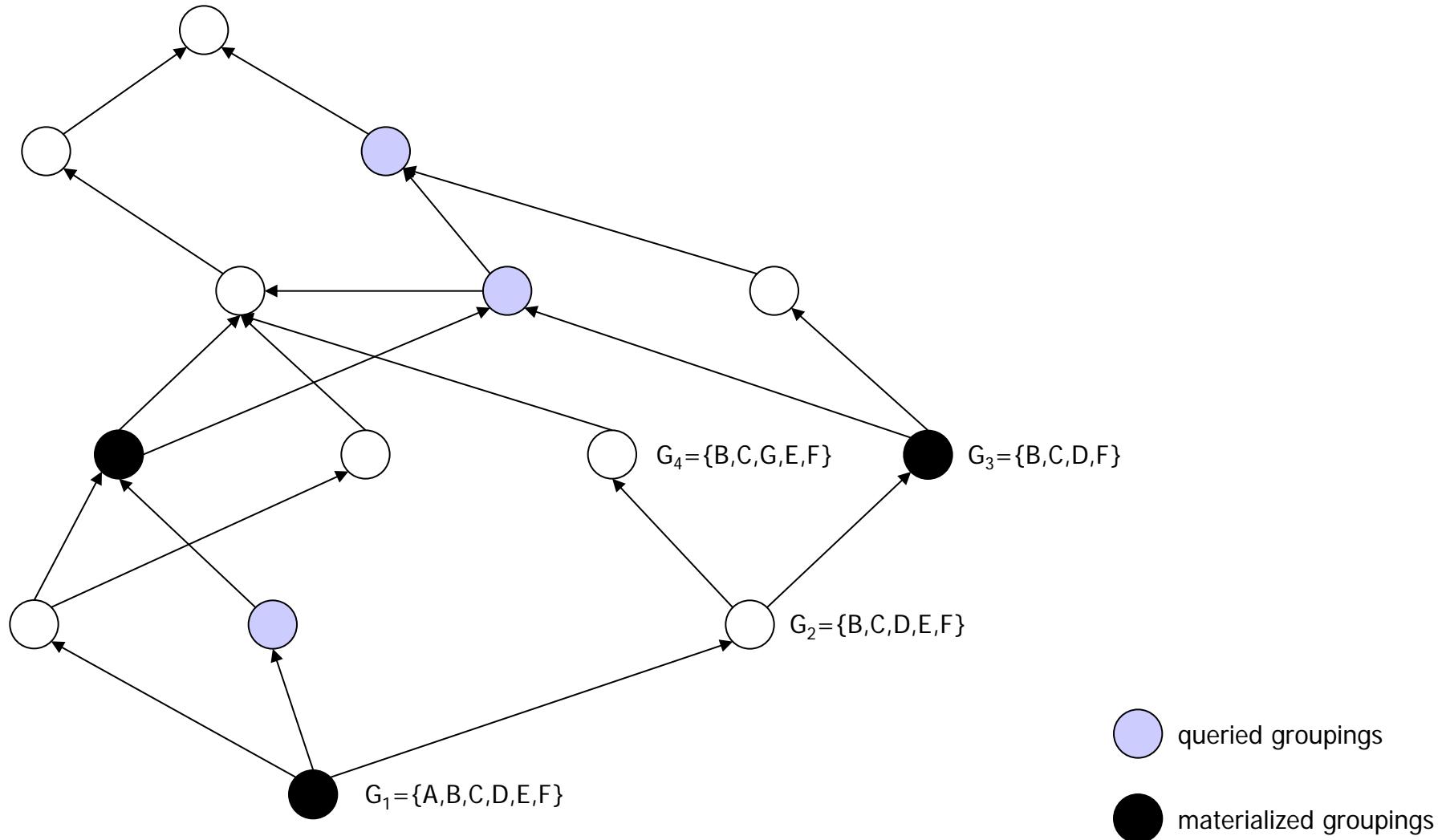
Derivability and Groupings

- $G_i = \{A_1, \dots, A_n\}$ is a grouping with grouping attributes A_1, \dots, A_n and $\forall k, l \quad 1 \leq k, l \leq n, k \neq l : \neg(A_k \rightarrow A_l)$
- Grouping G_2 is **directly derivable** from G_1 if one of the following conditions holds:
 - the grouping attributes of G_2 contain the grouping attributes of G_1 except one, i.e., $G_2 \subset G_1$ and $|G_2| = |G_1| - 1$
 - exactly one grouping attribute A_k of G_1 is replaced in G_2 by another attribute A_o such that $A_k \rightarrow A_o$.
- An acyclic dependency graph is build from groupings that are directly derivable.
- Grouping G_2 is **derivable** from G_1 if a path from G_1 to G_2 exists in the dependency graph ($G_1 < G_2$).
- A grouping set $GS_2 = \{G_1^2, \dots, G_m^2\}$ is derivable from grouping set $GS_1 = \{G_1^1, \dots, G_n^1\}$ if each grouping of GS_2 is derivable from at least one grouping of GS_1 .

Sample Dependency Graph



Sample Dependency Graph

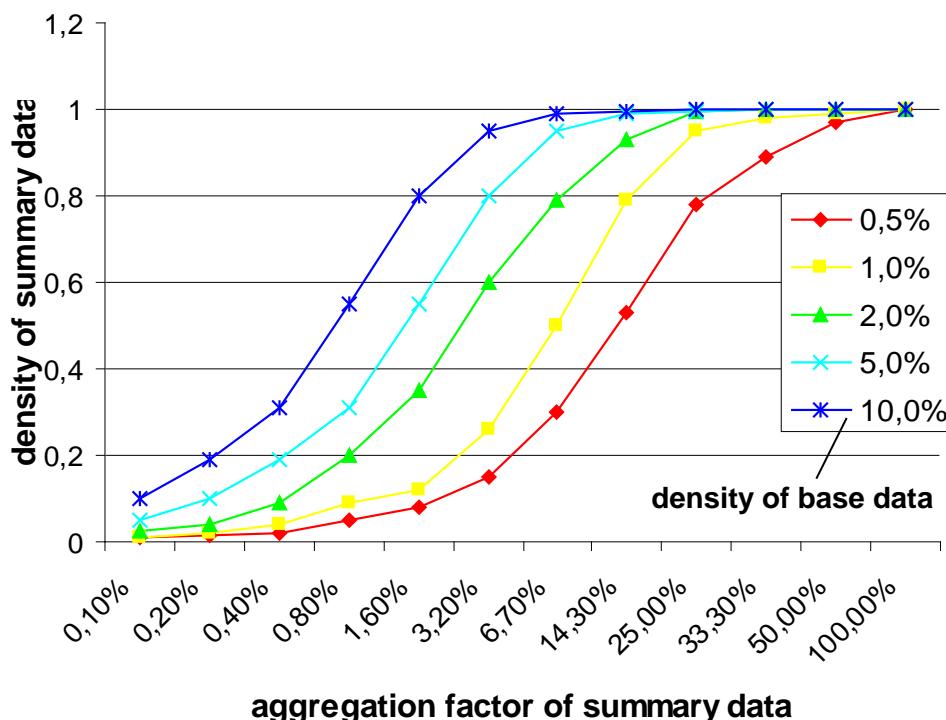
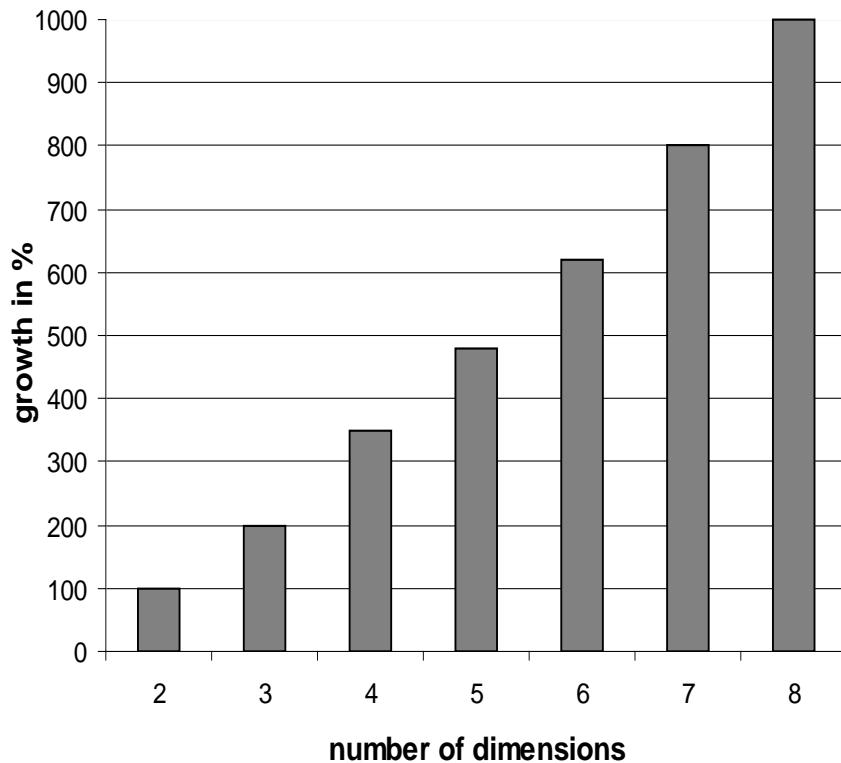


Number of Groupings

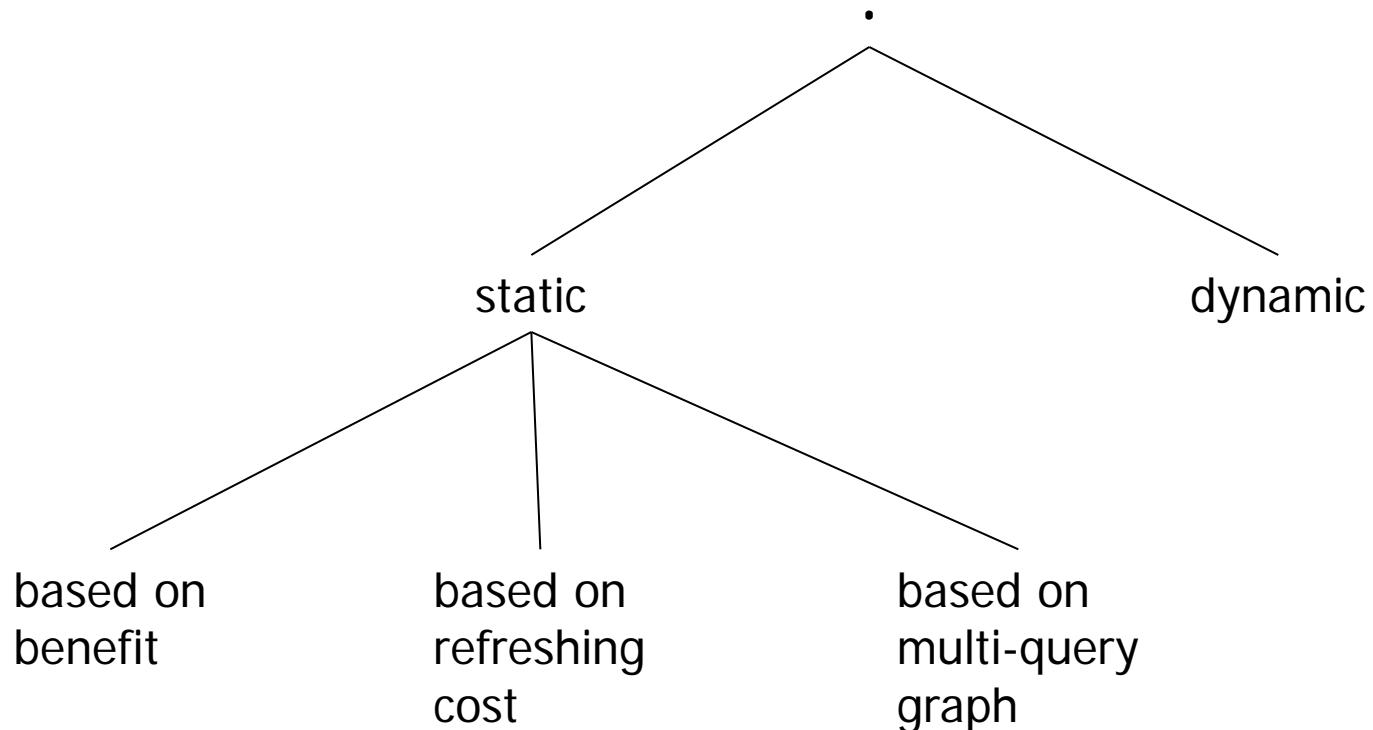
- The number of groupings that have to be considered for materialized summary data grows exponentially with the number of available grouping attributes.
- For n attributes, 2^n groupings have to be considered:
 - 3 attributes → 8 combinations
 - 10 attributes → 1024 combinations
 - 25 attributes → 33554432 combinations

Groupings and Sparsity

- Growth of data volume density: 2%; aggregation factor 1:20
- Density of summary data depending on aggregation factor and density of base data



Materialized View Selection Approaches



Materialized View Selection

- Set of n queries (workload): $Q = \{q_1, \dots, q_n\}$
- Set of m materialized views: $M = \{v_1, \dots, v_m\}$
- Execution cost for query q based on materialized view v : $C(q, v)$
- Execution cost for query q based on materialized views M : $C(q, M) = \min_{v \in M} (C(q, v))$
- Frequency of query q_i in workload: f_i
- Execution cost for all queries in Q based on materialized views M :
$$C(Q, M) = \sum_{q_i \in Q} f_i \cdot C(q_i, M)$$
- **Benefit** of additional materialized views u_1, \dots, u_k :
$$B(\{u_1, \dots, u_k\}, M) = C(Q, M) - C(Q, M \cup \{u_1, \dots, u_k\})$$

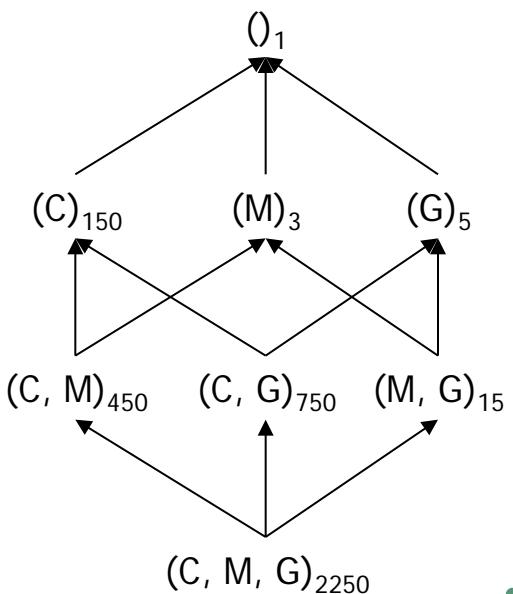
View Selection Algorithm

```
A: Set of all possible materialized views
S: Space for materialized views
M: Set of selected materialized views
select_views(A, S) begin
    M = {most granular materialized view}           // adjust available space S
    while S > 0 do begin
        uopt = ∅;
        BPUS({uopt}, M) = 0;
        for each u ∈ (A-M) do begin
            B({u}, M) = 0;
            for each q ∈ descendants(u) do
                B({u}, M) = B({u}, M) + C(q, M) - C(q, M ∪ {u})
            BPUS({u}, M) = B({u}, M) / u.size
            if BPUS({u}, M) > BPUS({uopt}, M) then
                uopt = u;
        end;
        if S - uopt.size > 0 then
            M = M ∪ {uopt};
            S = S - uopt.size;
        else
            S = 0;
        end;
    end;
    return M;
end;
```

Example: View Selection Algorithm

Sales

country	month	group	sales
F	01/04	100	500
F	02/04	100	1000
F	03/04	100	250
F	04/04	100	750
F	05/04	200	1250
F	02/04	200	2000
F	03/04	200	500
GB	01/04	100	400
GB	02/04	100	800
GB	03/04	100	300
GB	04/04	100	2000
GB	05/04	100	2500
GB	02/04	200	100
GB	03/04	200	100
GB	04/04	200	100
GB	05/04	200	100
I	01/04	300	250
I	02/04	300	750
I	03/04	300	200
I	04/04	300	1500
I	05/04	300	800



- Possible groupings and their cardinality.

grouping	cardinality
none	1
C	150
M	3
G	5
C, M	450
C, G	750
M, G	15
C, M, G	2250

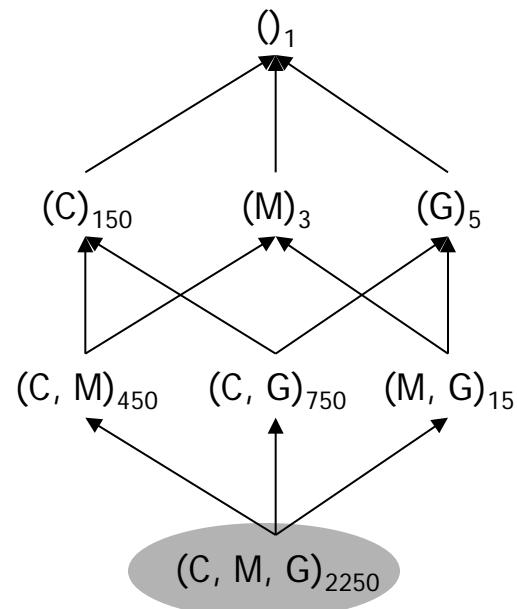
- Cardinality reflects the query execution cost.

Example: View Selection Algorithm

- $A = \{(), (C), (M), (G), (C, M), (C, G), (M, G), (C, M, G)\}$
- first step:
 $M = \{(C, M, G)\}, S = 50$

$S = 2300$

$M = \{\}$

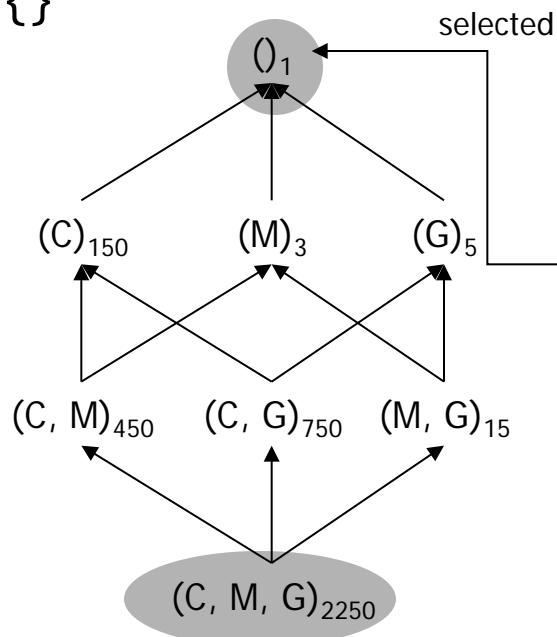


Example: View Selection Algorithm

- $A = \{(), (C), (M), (G), (C, M), (C, G), (M, G), (C, M, G)\}$

$S = 2300$

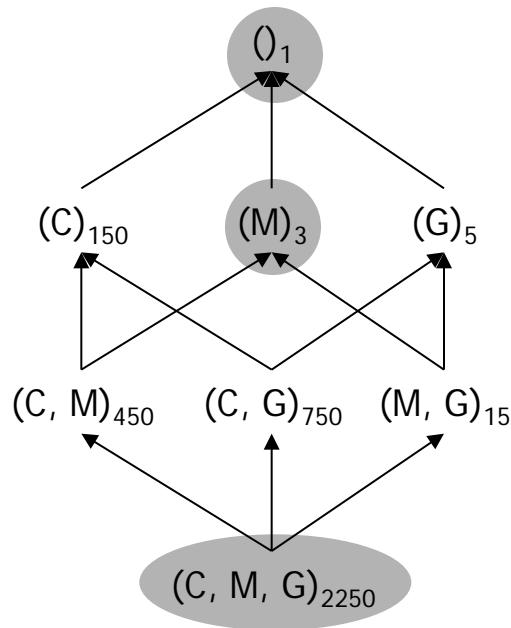
$M = \{\}$



u	$B(\{u\}, M)$	$BPUS(\{u\}, M)$
(C, M)	$(2250-450) + (2250-450) + (2250-450) + (2250-450)$	16
(C, G)	$(2250-750) + (2250-750) + (2250-750) + (2250-750)$	8
(M, G)	$(2250-15) + (2250-15) + (2250-15) + (2250-15)$	596
(C)	$(2250-150) + (2250-150)$	28
(M)	$(2250-3) + (2250-3)$	1498
(G)	$(2250-5) + (2250-5)$	898
()	$(2250-1)$	2249

- second step:
 $M = \{(C, M, G), ()\}$, $S = 49$

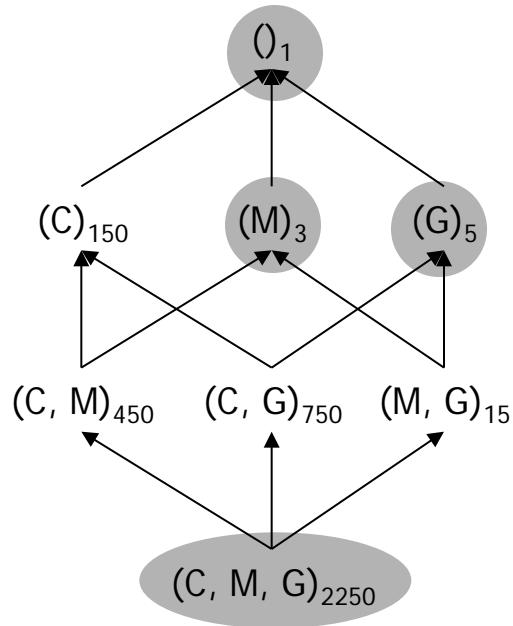
Example: View Selection Algorithm



u	B({u}, M)	BPUS({u}, M)
(C, M)	$(2250-450)+(2250-450)+(2250-450) = 5400$	12
(C, G)	$(2250-750)+(2250-750)+(2250-750) = 4500$	6
(M, G)	$(2250-15)+(2250-15)+(2250-15) = 6705$	447
(C)	$(2250-150) = 2100$	14
(M)	$(2250-3) = 2247$	749
(G)	$(2250-5) = 2245$	449

- third step:
 $M=\{(C, M, G), (), (M)\}$, $S=46$

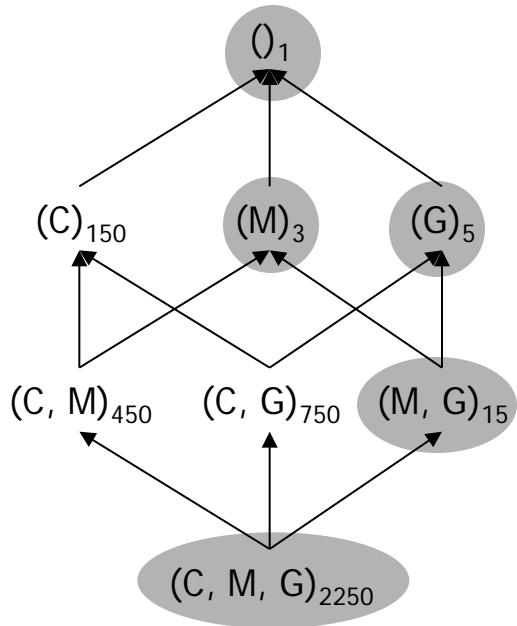
Example: View Selection Algorithm



u	B({u}, M)	BPUS({u}, M)
(C, M)	$(2250-450)+(2250-450) = 1800$	4
(C, G)	$(2250-750)+(2250-750)+(2250-750) = 4500$	6
(M, G)	$(2250-15)+(2250-15) = 4470$	298
(C)	$(2250-150) = 2100$	14
(G)	$(2250-5) = 2245$	449

- fourth step:
 $M=\{(C, M, G), (), (M), (G)\}, S=41$

Example: View Selection Algorithm



u	B({u}, M)	BPUS({u}, M)
(C, M)	$(2250-450)+(2250-450) = 1800$	4
(C, G)	$(2250-750)+(2250-750) = 3000$	4
(M, G)	$(2250-15) = 2235$	149
(C)	$(2250-150) = 2100$	14

- fifth step:
 $M=\{(C, M, G), (), (M), (G), (M, G)\}$
 $S=41$
- not enough space for further materialization
- final result:
 $\{(), (M), (G), (M, G), (C, M, G)\}$

View Selection Algorithm (by Size)

A: Set of all possible materialized views

S: Space for materialized views

M: Set of selected materialized views

```
select_views_sorted(A, S) begin
    A = sort_desc_by_size(A);
    M = {most granular materialized view}
    while S > 0 do begin
        umin = min(A);
        if S - umin.size > 0 then
            M = M ∪ {umin};
            S = S - umin.size;
        else
            S = 0;
        end;
    end;
    return M;
end;
```

Dynamic Approaches

- Static approaches assume a constant query profile for the data warehouse.
- Dynamic approaches:
 - Store query results as materialized summary data (similar to caching)
 - If not enough space is left
-> choose replacement victim
 - General conditions to be considered in replacement strategies:
 - size of materialized summary data
 - dependencies between materialized summary data
 - invalidation versus refreshing

Refreshing Summary Data

- immediate refresh
 - on commit refresh
 - deferred refresh
- incremental maintenance
(self maintainability required)
 - complete refresh

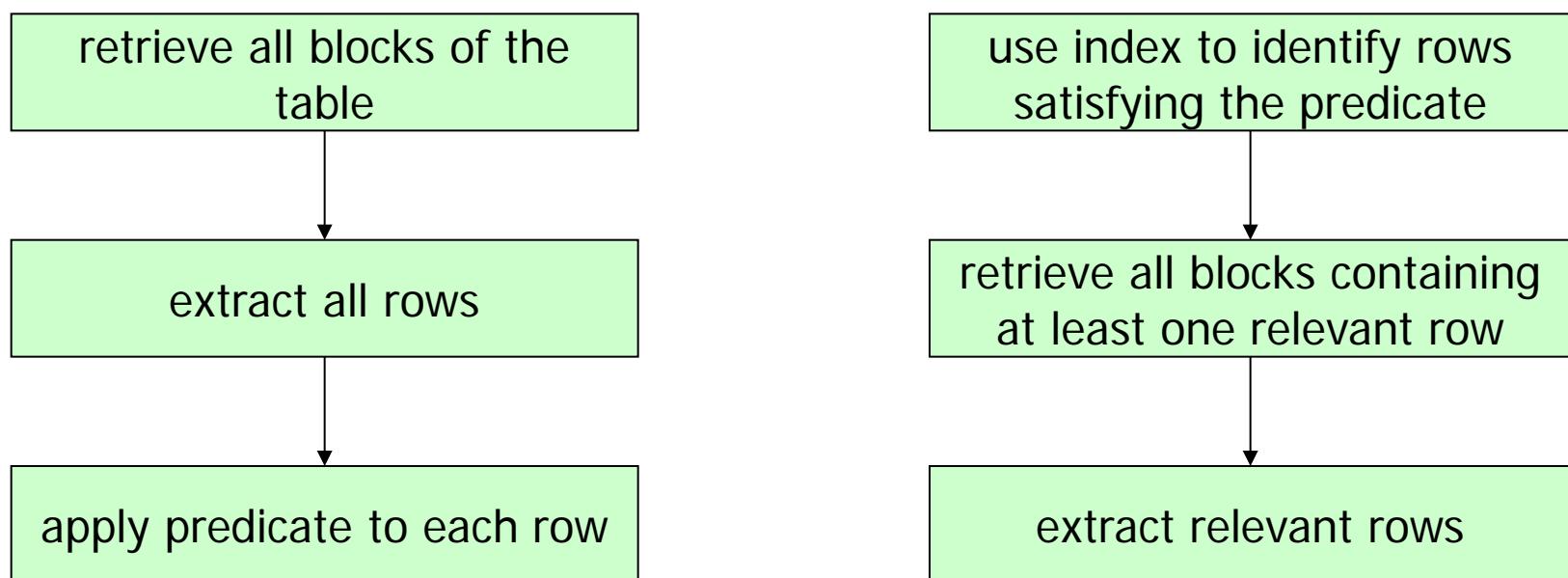
complete	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
incremental	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
	immediate	on-commit	deferred

Overview

- Support for OLAP and Data Mining in SQL
 - OLAP:
ROLLUP, CUBE, WINDOW, OLAP Functions
 - Data Mining (SQL/MM)
- Database Support for OLAP
 - Partitioning
 - Materialized Views
 - Indexes
 - Optimization of OLAP Queries (Star Joins)
- Performance Evaluation: TPC Benchmarks

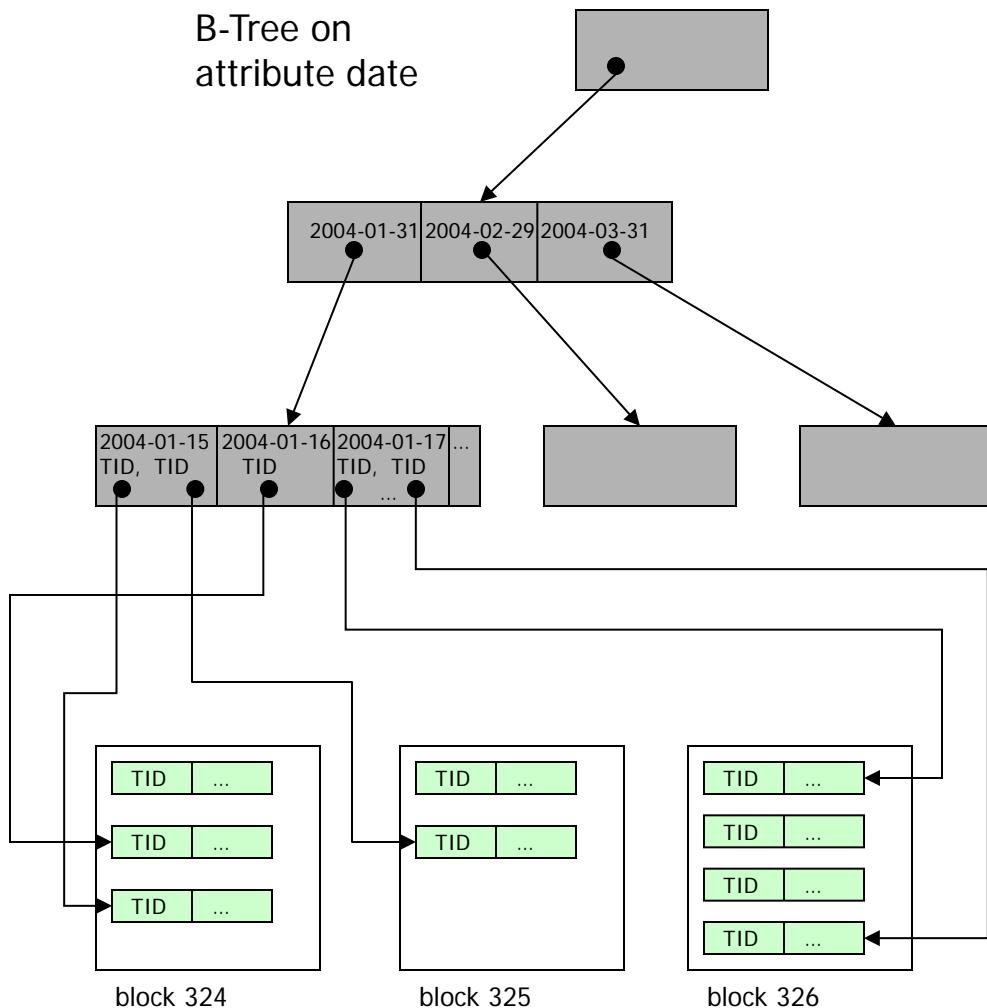
Scan

- All rows of a relation are stored in blocks on external memory:
 - hard drive
 - RAID
 - ...
- Table scan
 - Scan operation:
Retrieve all rows of a table that satisfy a predicate.
 - Index scan



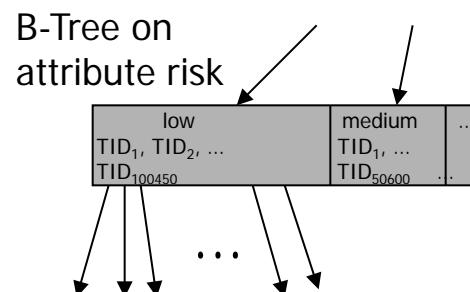
Indexes

- Each tuple is identified by a tuple identifier (TID) (also RID = row identifier)
- Indexes support the efficient mapping of attribute values to the corresponding rows in a table.



Indexes on Dimension Tables

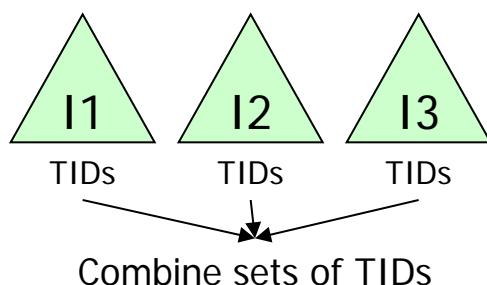
- OLAP-queries typically provide restrictions on dimension attributes.
- index allows to identify qualified rows in a dimension
- **problem:**
 - low-cardinality dimension attributes



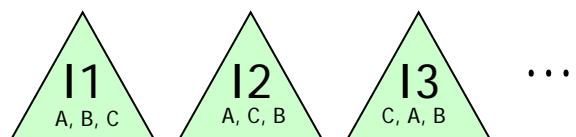
Indexes on Fact Tables

- OLAP-queries typically provide restrictions on the primary key of a fact table.
- combined index on primary key attributes allows to identify relevant rows of the fact table
- **problems:**
 - $n!$ possible index structures
 - size of the index

- Use multiple indexes on single attributes

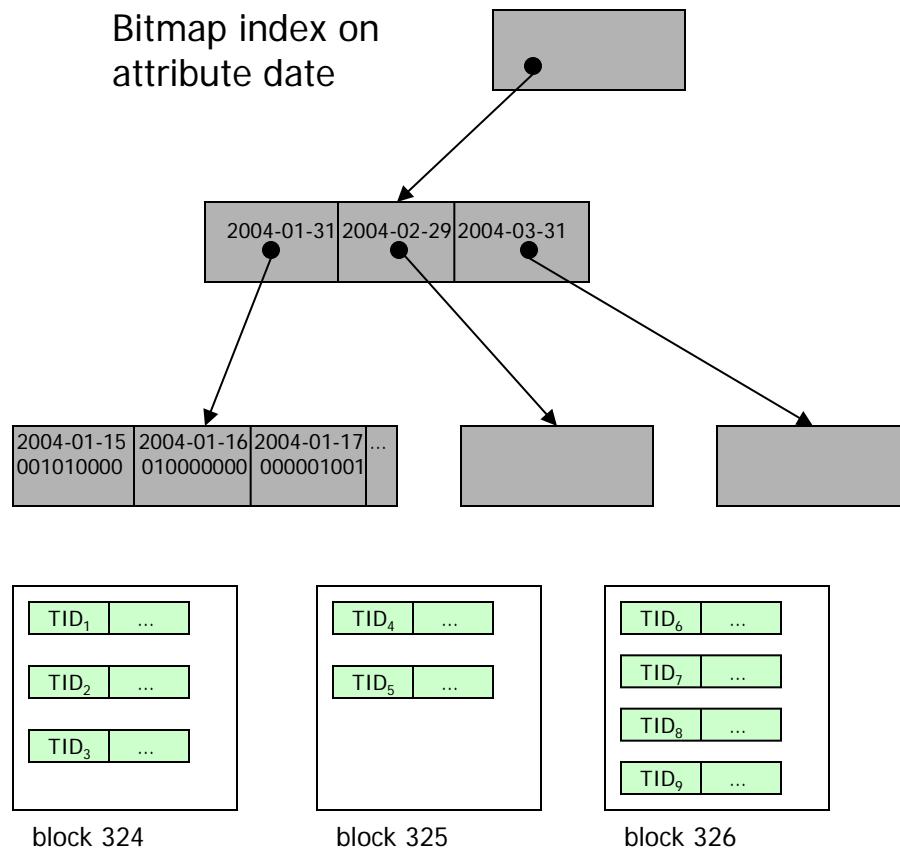


- Use multi-attribute indexes



Bitmap Indexes

- Basic idea:
 - replace TIDs in a B-Tree by a bitmap for key k
 - the bitmap contains a '1' at position i, if tuple t_i has value k for the index attribute
- Advantages:
 - less index space for low-cardinality attributes
 - bitmap compression
 - efficient combination of bitmaps possible



Sample Bitmaps

Orders

o_orderkey	o_custkey	o_orderstatus	o_orderprice	o_orderday	o_orderpriority	o_clerk	o_shippriority
1	3691	O	194029.55	1996-01-02	5-LOW	Clerk#000000951	0
2	7801	O	60951.63	1996-12-01	1-URGENT	Clerk#000000880	0
3	12332	F	247296.05	1993-10-14	5-LOW	Clerk#000000955	0
4	13678	O	53829.87	1995-10-11	5-LOW	Clerk#000000124	0
5	4450	F	139660.54	1994-07-30	5-LOW	Clerk#000000925	0
6	5563	F	65843.52	1992-02-21	4-NOT SPECIFIED	Clerk#000000058	0
7	3914	O	231037.28	1996-01-10	2-HIGH	Clerk#000000470	0
32	13006	O	166802.63	1995-07-16	2-HIGH	Clerk#000000616	0
33	6697	F	118518.56	1993-10-27	3-MEDIUM	Clerk#000000409	0
34	6101	O	75662.77	1998-07-21	3-MEDIUM	Clerk#000000223	0
35	12760	O	192885.43	1995-10-23	4-NOT SPECIFIED	Clerk#000000604	0
36	11527	O	72196.43	1995-11-03	1-URGENT	Clerk#000000358	0
37	8612	F	156440.15	1992-06-03	3-MEDIUM	Clerk#000000456	0
38	12484	O	64695.26	1996-08-21	4-NOT SPECIFIED	Clerk#000000604	0
39	8177	O	307811.89	1996-09-20	3-MEDIUM	Clerk#000000659	0
...

B_{o_clerk} :

Clerk#000000951: 1000000000000000...

Clerk#000000604: 00000000010010...

...

$B_{o_orderpriority}$:

1-URGENT: 01000000001000...

2-HIGH: 0000011000000...

3-MEDIUM: 00000001100101...

4-NOT SPECIFIED: 00001000010010...

5-LOW: 10111000000000...

$B_{o_orderstatus}$:

O: 110100110111011...

F: 001011001000100...

$B_{o_shippriority}$:

O: 111111111111111...

Bitmap Usage

- using bitmaps for exact match queries:
 $X = D$
- using bitmaps for range queries:
 $X \text{ IN}(A, B, C, D, E)$

result bitmap B:

$$B[i] = B_{X,D}[i]$$

result bitmap B:

```
for i=1 to length(BX)
    B[i] = BX,A[i]
    OR  BX,B[i]
    OR  BX,c[i]
    OR  BX,D[i]
    OR  BX,E[i] )
```

end for

Sample Query

```
SELECT SUM(l_quantity) AS sum_quan
FROM tpch.lineitem, tpch.orders
WHERE l_orderkey = o_orderkey
AND o_orderstatus = 'F'
AND o_shippriority = '0'
AND (o_orderpriority IN ('5-LOW', '4-NOT SPECIFIED')
OR o_clerk = 'Clerk#000000604');
```

```
for i=1 to length(tpch.orders)
    B[i] =      B_o_orderstatus,F[i]
    AND      B_o_shippriority,0[i]
    AND (      B_o_orderpriority,4 [i]
    OR       B_o_orderpriority,5 [i]
    OR       B_o_Clerk#000000604 [i] )
end for
```

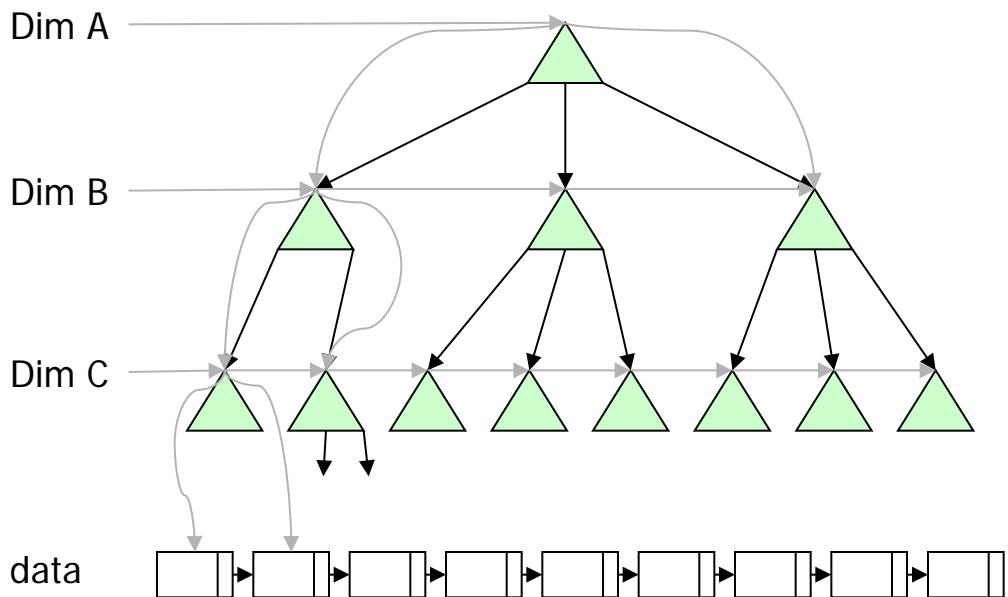


B:

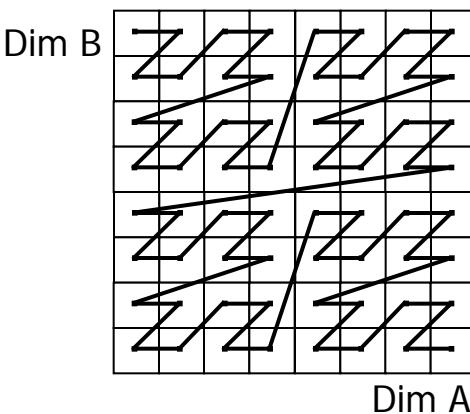
```
001011001000100...
AND 111111111111111...
AND (
00001000010010...
OR 10111000000000...
OR 00000000010010...
)
=
001011000000000...
```

Combining One-dimensional Indexes

- MDB tree
 - combines several one-dimensional indexes



- UB tree
 - uses a space-filling curve to map a multidimensional universe to one-dimensional space
 - Z-Curve preserves the multidimensional proximity
 - uses standard B*-Tree to index the tuples taking the Z-Address as keys
 - provides multidimensional clustering of the data



Multidimensional Indexes

- Symmetric treatment of all dimensions
- Examples:
 - Quadtree (2 dimensions)
 - Octtree (3 dimensions)
 - K-D-B-Tree
 - hB-Tree (holy brick)
 - R-Trees: R-tree, R⁺-tree, R^{*}-tree, packed R-tree
 - ...

Overview

- Support for OLAP and Data Mining in SQL
 - OLAP:
ROLLUP, CUBE, WINDOW, OLAP Functions
 - Data Mining (SQL/MM)
- Database Support for OLAP
 - Partitioning
 - Materialized Views
 - Indexes
 - Optimization of OLAP Queries (Star Joins)
- Performance Evaluation: TPC Benchmarks

Star Queries

SELECT D1.a, D2.d, ..., Dn.x, f(F факт)

FROM F, D1, D2, ..., Dn

WHERE F.b = D1.b

AND F.e = D2.e

...

AND D1.c = ... $\longrightarrow \sigma_1$

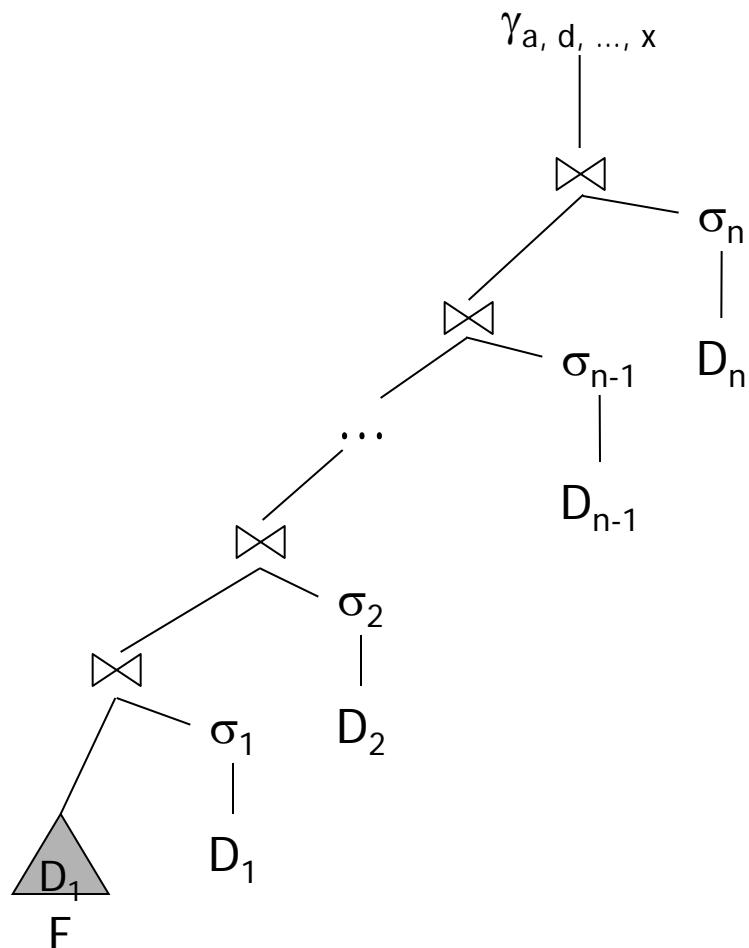
AND D2.f = ... $\longrightarrow \sigma_2$

...

GROUP BY D1.a, D2.d, ..., Dn.x

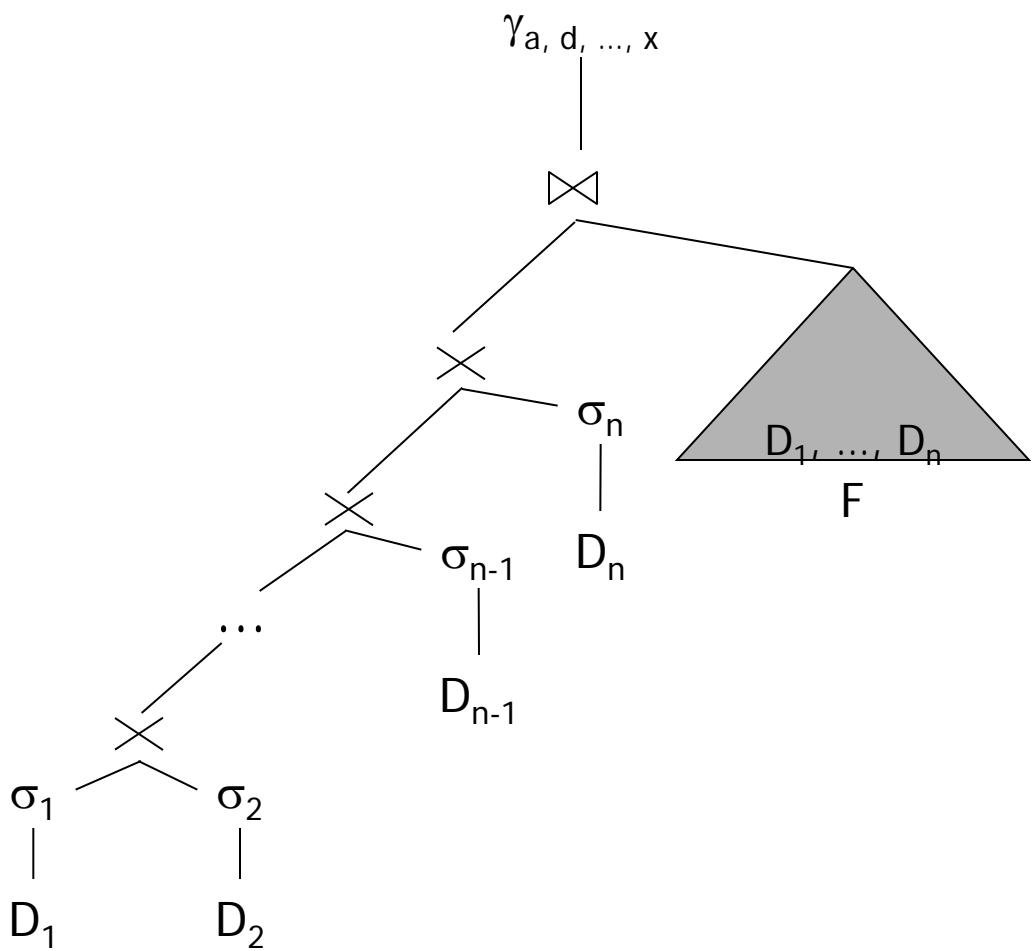
- Fact table F
- Dimension tables D1, ..., Dn
- Aggregation f applied to fact attribute F факт
- Optimizer has to decide on:
 - join order
 - index usage
 - predicate push-down
 - ...

Evaluating Star Queries



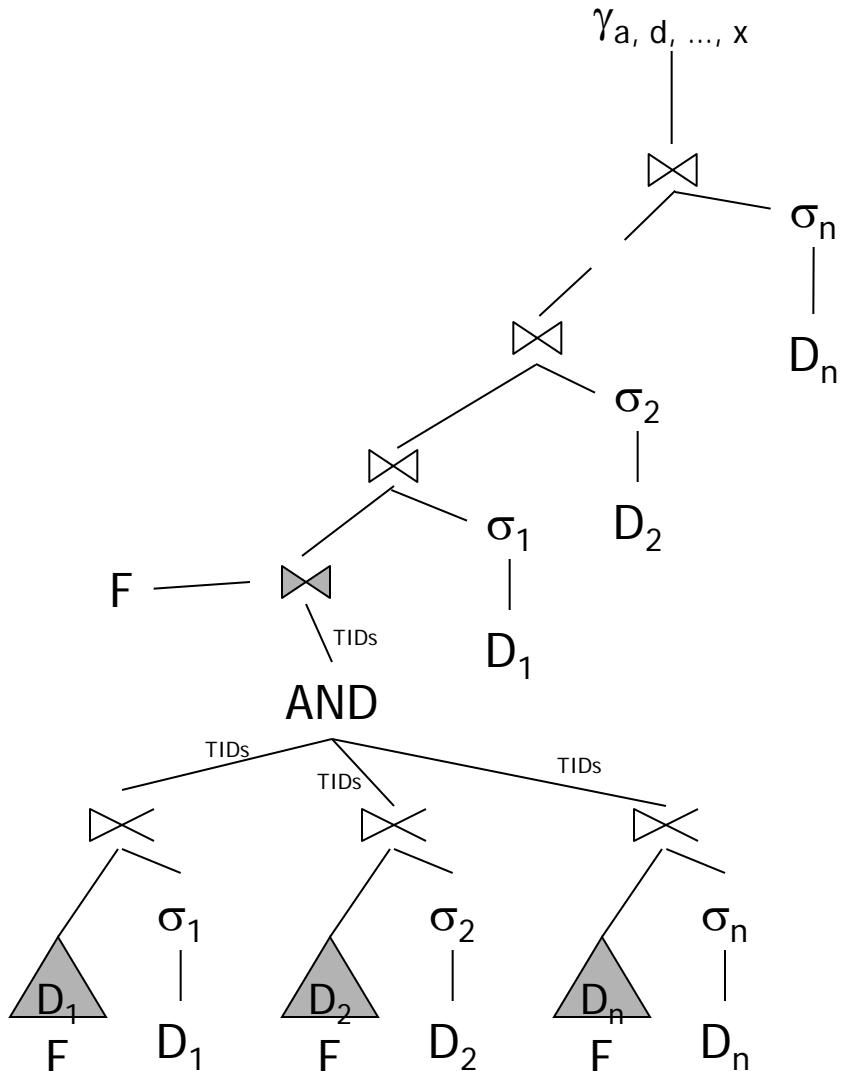
- Basic approach:
 - join fact table and first dimension table.
 - use index on fact table (join attribute) if available
 - join result with further dimension tables until last dimension is reached
- Predicates on dimensions reduce size of intermediate results.
- Join order for dimensions depends on:
 - available indexes
 - selectivity of predicates

Evaluating Star Queries



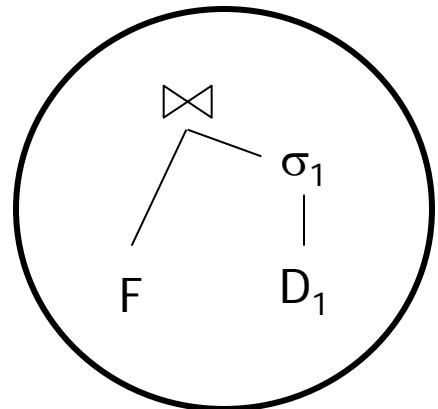
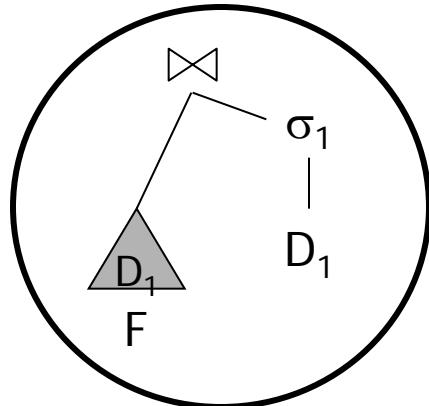
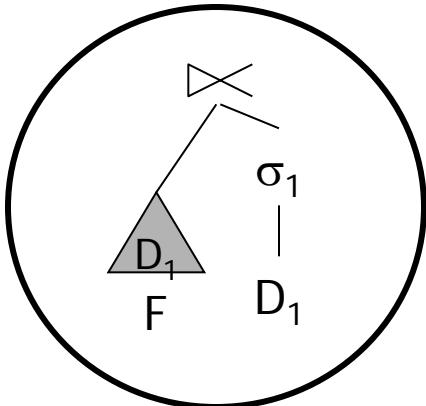
- Basic approach:
 - join all dimension tables
 - join resulting cartesian product to the fact table
 - use combined index on fact table if available
- Reduces size of intermediate results

Evaluating Star Queries



- Basic approach:
 - semi-join of fact table and each dimension table (results in n sets of TIDs for qualified rows)
 - index on fact table (join attribute) is mandatory
 - combine sets of TIDs
 - fetch qualified rows from fact table
 - join intermediate result with each dimension table
- Reduces size of intermediate results
- Efficient algorithm needed for intersection of sets of TIDs

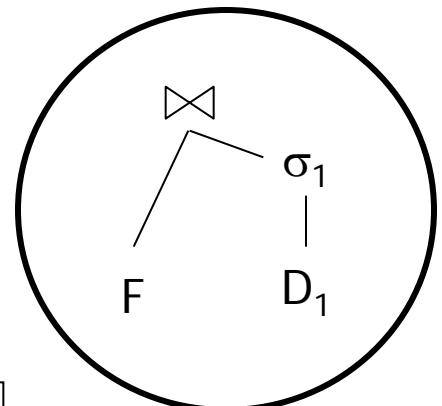
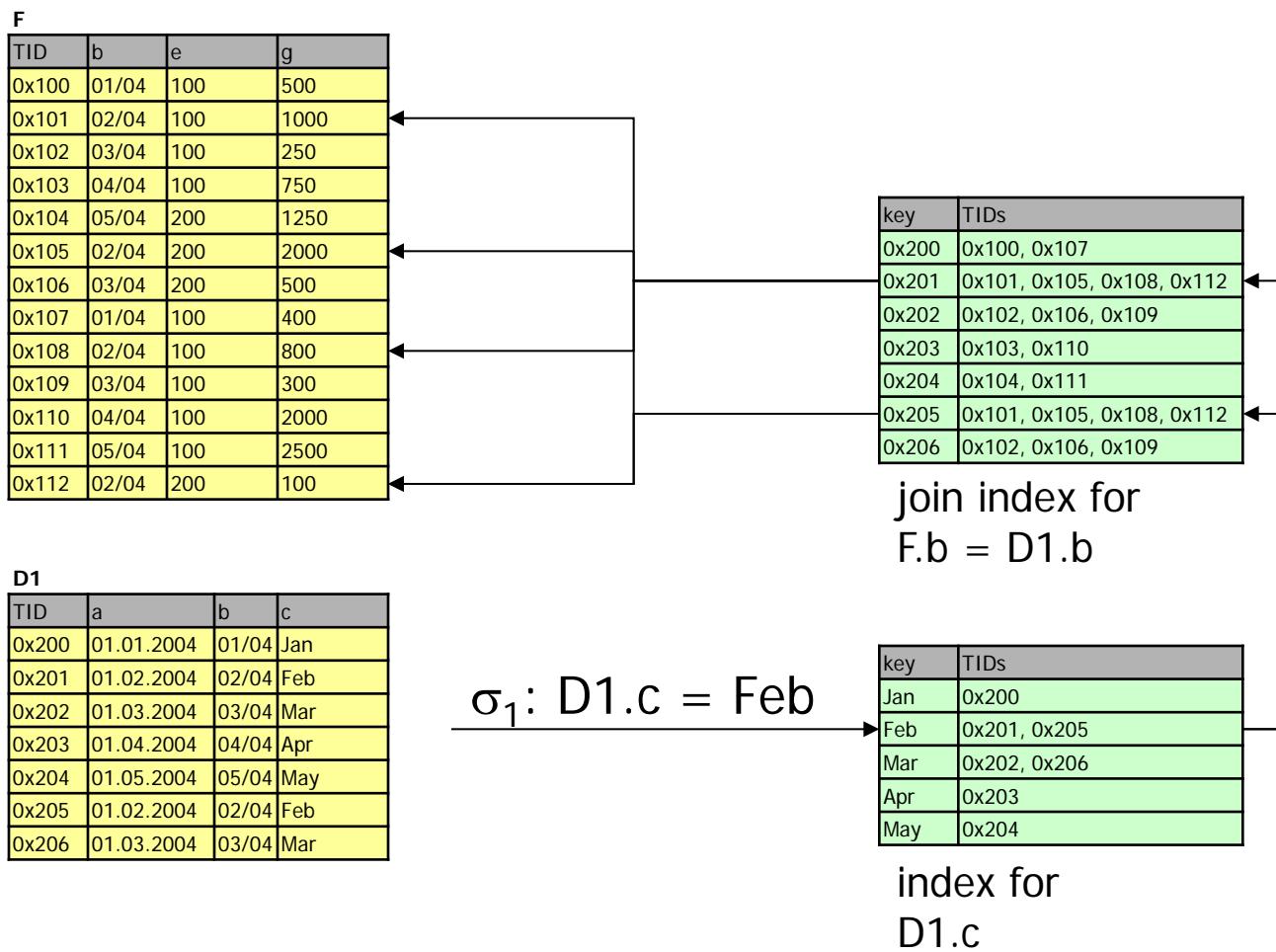
Join Indexes



- **Goal:**
Provide join between fact table F and dimension table D1 without access to both tables.
- **Join Index:**
Index structure covering the pre-computed join between two tables.

```
SELECT D1.a, D2.d, ..., Dn.x, f(F факт)  
FROM F, D1, D2, ..., Dn  
WHERE F.b = D1.b  
AND F.e = D2.e  
...  
AND D1.c = ...  
AND D2.f = ...  
...  
GROUP BY D1.a, D2.d, ..., Dn.x
```

Join Indexes



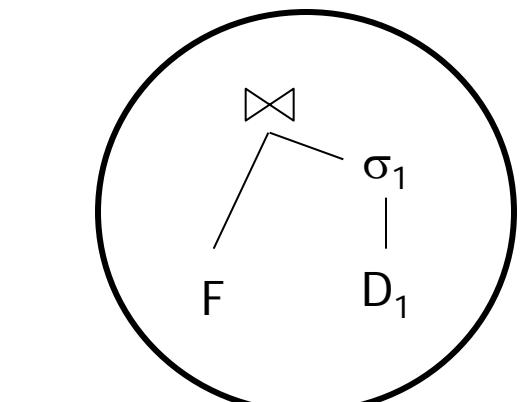
Join Indexes based on Bitmaps

F	TID	b	e	g
	0x100	01/04	100	500
	0x101	02/04	100	1000
	0x102	03/04	100	250
	0x103	04/04	100	750
	0x104	05/04	200	1250
	0x105	02/04	200	2000
	0x106	03/04	200	500
	0x107	01/04	100	400
	0x108	02/04	100	800
	0x109	03/04	100	300
	0x110	04/04	100	2000
	0x111	05/04	100	2500
	0x112	02/04	200	100

D1	TID	a	b	c
	0x200	01.01.2004	01/04	Jan
	0x201	01.02.2004	02/04	Feb
	0x202	01.03.2004	03/04	Mar
	0x203	01.04.2004	04/04	Apr
	0x204	01.05.2004	05/04	May
	0x205	01.02.2004	02/04	Feb
	0x206	01.03.2004	03/04	Mar

D2	TID	d	e	f
	0x300	E	100	very low
	0x301	D	200	low
	0x302	C	300	medium
	0x303	B	400	high
	0x304	A	500	very high

key	Bitmap
Jan	1000001000000
Feb	0100010010001
Mar	0010001001000
Apr	0001000000100
May	0000100000010



key	Bitmap
very low	1111000111110
low	0000111000001
medium	0000000000000
high	0000000000000
very high	0000000000000

multi table join index

Algorithm for GROUP BY

G_i : GROUP BY attributes
AGG: aggregation function

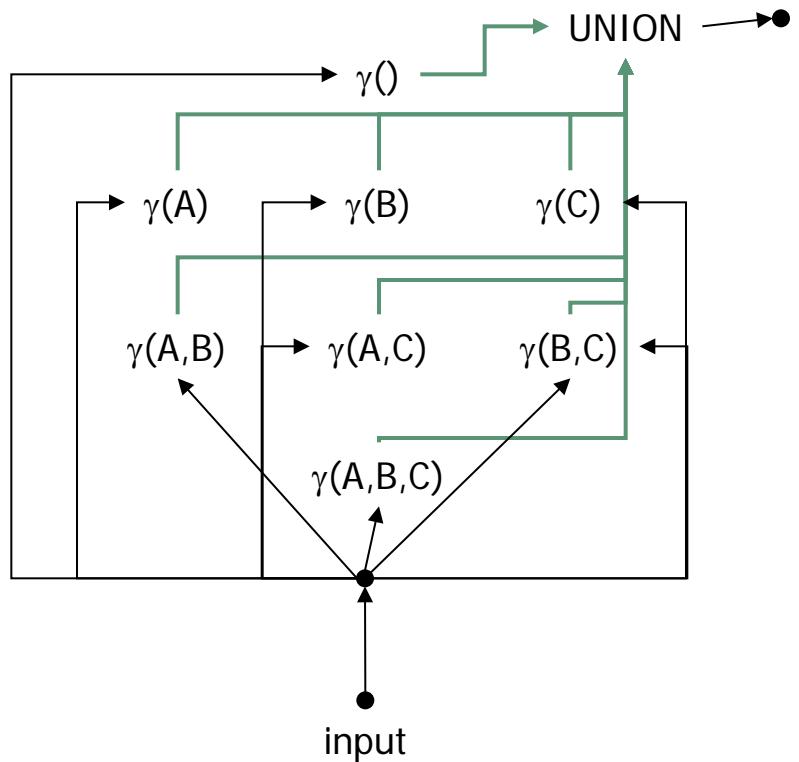
Begin

```
SORT( $G_1, \dots, G_n$ );  
While (Input not empty)  
    ( $\$g_1, \dots, \$g_n, \$val$ ) = ReadNextTuple(Input);  
    If (current and last tuple are equal for  $G_1, \dots, G_n$ ) Then  
        $aggrset = $aggrset  $\cup$  {$val};  
    Else  
        $aggrval = AGG($aggrset);  
        WriteNextTuple(Output,  $g_1, \dots, g_n, \$aggrval$ );  
        $aggrset = ($val);  
    End If  
End While  
$aggrval = AGG($aggrset);  
WriteNextTuple(Output,  $g_1, \dots, g_n, \$aggrval$ );  
End
```

Complex Groupings

- Naïve approach:
Build all groupings from detailed data.
- Drawbacks:
 - input of grouping might be derived from base tables for each grouping individually
 - no reuse of intermediate grouping results

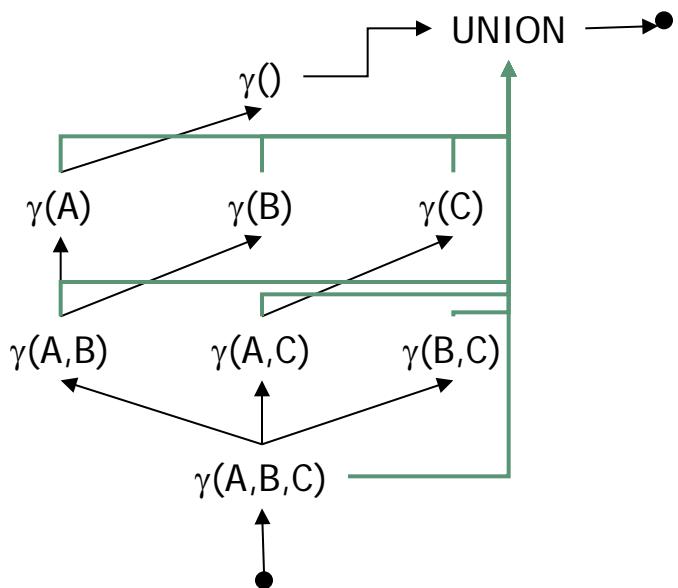
- CUBE(A,B,C)



Complex Groupings Based on Derivability

- Optimization heuristics:
 - build grouping from smallest parent
- Derivability has to be analyzed
- Advantages:
 - Cardinality of data streams is reduced

- CUBE (A,B,C)

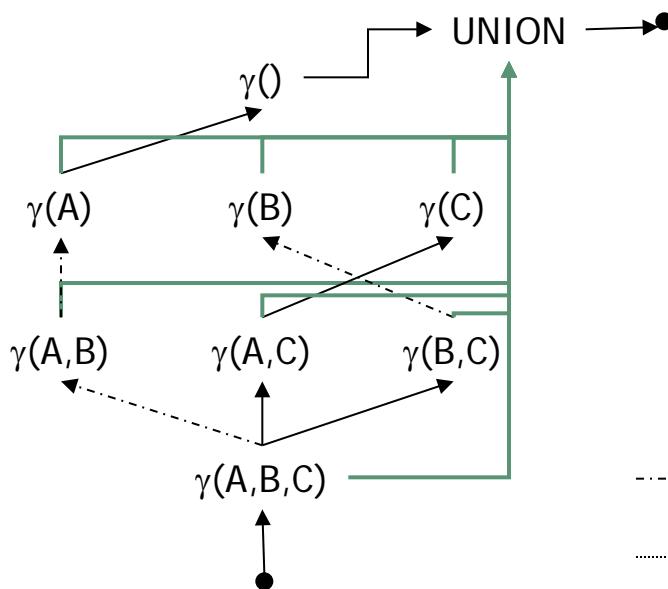


Exploiting Sort Order for Complex Groupings

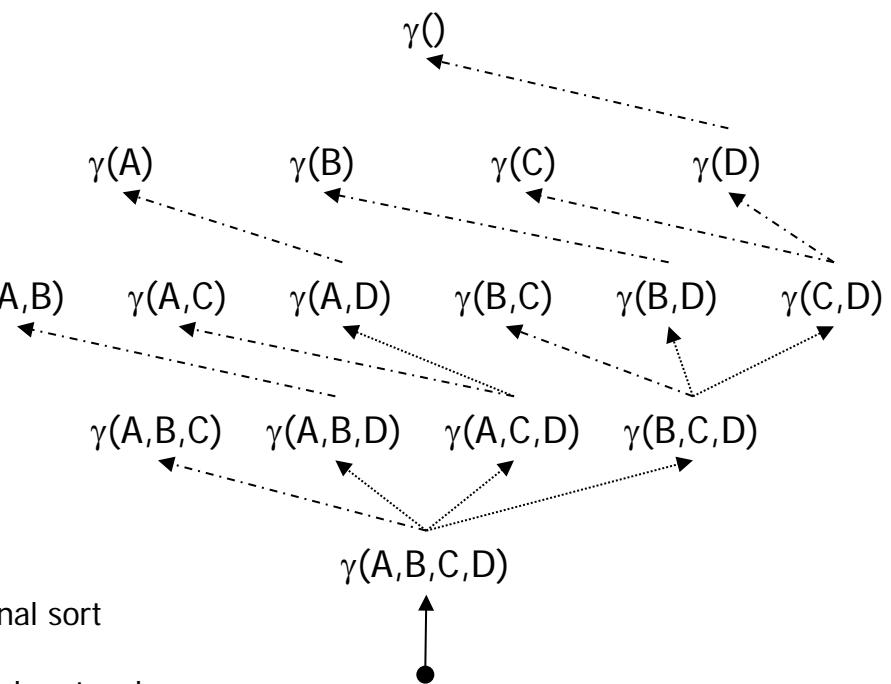
- Optimization heuristics:

- use existing ordering
- e.g. sort data on A,B,C and derive $\gamma(A,B)$, $\gamma(A)$, and $\gamma()$ without additional sorts

- CUBE (A,B,C)



→ no additional sort
 → keep partial sort order



(Source: [Len03])

Algorithm for OLAP Functions

Begin

SORT($G_1, \dots, G_n, P_1, \dots, P_m, O_1, \dots, O_p$);

While (Input not empty)

\$tuple = ReadNextTuple(Input);

 If (current and last tuple are equal for $G_1, \dots, G_n, P_1, \dots, P_m$) Then

\$agglist = \$agglist + (\$tuple);

Else /* process partition */

For \$pos = 1 To Length(\$agglist)

 Switch(W_L)

Case UNBOUNDED PRECEDING: \$low = 1;

Case CURRENT: \$low = \$pos;

 Default: \$low = Max(1, \$pos- W_L);

End Switch

 Switch (W_H)

Case UNBOUNDED FOLLOWING: \$high = Length(\$agglist);

Case CURRENT: \$high = \$pos;

 Default: \$high = Min(Length(\$agglist), \$pos+ W_H);

End Switch

\$aggval = AGG(\$agglist[\$low], ..., \$agglist[\$high]);

 WriteNextTuple(Output, $G_1, \dots, G_n, P_1, \dots, P_m, O_1, \dots, O_p, \$aggval$);

End For

\$agglist = (\$tuple);

End If

End While

/* process last partition similar to else path */

End

 G_i : GROUP BY attributes O_j : WINDOW ordering (OVER clause) P_k : WINDOW partitioning W_L, W_H : lower and upper bound of WINDOW

AGG: aggregation function

Overview

- Support for OLAP and Data Mining in SQL
 - OLAP:
ROLLUP, CUBE, WINDOW, OLAP Functions
 - Data Mining (SQL/MM)
- Database Support for OLAP
 - Partitioning
 - Materialized Views
 - Indexes
 - Optimization of OLAP Queries (Star Joins)
- Performance Evaluation: TPC Benchmarks

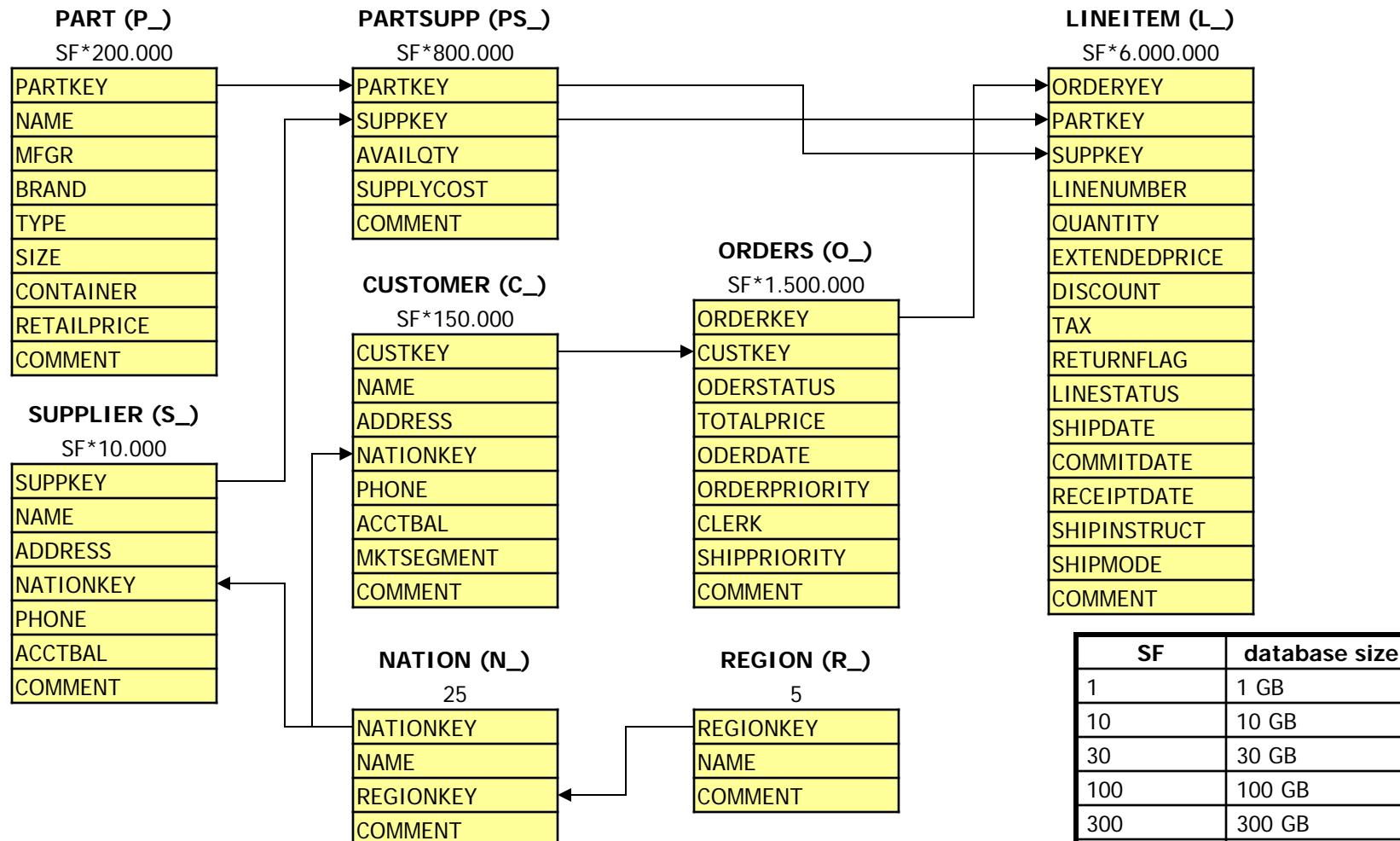
Key Criteria for Benchmarks

- Relevant:
 - It must measure the peak performance and price/performance of systems.
- Portable:
 - It should be easy to implement the benchmark on many different systems and architectures.
- Scaleable:
 - The benchmark should apply to small and large computer systems. It should be possible to scale the benchmark up to larger systems, and to parallel computer systems as computer performance and architecture evolve.
- Simple:
 - The benchmark must be understandable, otherwise it will lack credibility.

TPC-H

- Decision support benchmark.
- Business oriented ad-hoc queries (Q1 - Q22) and concurrent data modifications (RF1 - RF2).
- Benchmark illustrates decision support systems that examine large volumes of data, execute queries with a high degree of complexity, and give answers to critical business questions.
- Performance metric: TPC-H Composite Query-per-Hour Performance Metric (QphH@Size).
- TPC-H Price/Performance metric is expressed as \$/QphH@Size.

TPCH Benchmark Schema



Database Scaling:

SF	database size
1	1 GB
10	10 GB
30	30 GB
100	100 GB
300	300 GB
1000	1000 GB = 1 TB
3000	3000 GB
10000	10000 GB
30000	30000 GB
100000	100000 GB

Minimum Cost Supplier Query (Q2)

- **Business Question:**

The Minimum Cost Supplier Query finds, in a given region, for each part of a certain type and size, the supplier who can supply it at minimum cost. If several suppliers in that region offer the desired part type and size at the same (minimum) cost, the query lists the parts from suppliers with the 100 highest account balances. For each supplier, the query lists the supplier's account balance, name and nation; the part's number and manufacturer; the supplier's address, phone number and comment information.

```
SELECT s_acctbal, s_name, n_name, p_partkey,
       p_mfgr, s_address, s_phone, s_comment
  FROM part, supplier, partsupp, nation, region
 WHERE p_partkey = ps_partkey
   AND s_suppkey = ps_suppkey
   AND p_size = [SIZE]
   AND p_type like '%[TYPE]'
   AND s_nationkey = n_nationkey
   AND n_regionkey = r_regionkey
   AND r_name = '[REGION]'
   AND ps_supplycost = (
        SELECT min(ps_supplycost)
        FROM partsupp, supplier, nation, region
        WHERE p_partkey = ps_partkey
          AND s_suppkey = ps_suppkey
          AND s_nationkey = n_nationkey
          AND n_regionkey = r_regionkey
          AND r_name = '[REGION]'

        )
 ORDER BY s_acctbal desc, n_name, s_name,
          p_partkey;
```

Refresh Functions

- **Business Rationale (RF1):**
The New Sales refresh function inserts new rows into the ORDERS and LINEITEM tables in the database following the scaling and data generation methods used to populate the database.
- Refresh Function Definition:
LOOP (SF * 1500) TIMES
 INSERT a new row into table ORDERS
 LOOP RANDOM(1, 7) TIMES
 INSERT a new row into LINEITEM
 END LOOP
END LOOP
- **Business Rationale (RF2):**
The Old Sales refresh function removes rows from the ORDERS and LINEITEM tables in the database to emulate the removal of stale or obsolete information.
- Refresh Function Definition:
LOOP (SF * 1500) TIMES
 DELETE FROM ORDERS
 WHERE O_ORDERKEY = [value]
 DELETE FROM LINEITEM
 WHERE L_ORDERKEY = [value]
END LOOP

Metrics

- **Power Test:**
 - Queries submitted through a single session
 - Session executes queries one after another.
 - Used to measure the raw query execution power of the system.
- **Throughput test:**
 - Queries submitted through two or more sessions.
 - One session per query stream.
 - Each stream must execute queries serially.
 - Executed in parallel with a single refresh stream.
- **TPC-H Power:**
$$\text{Power@SIZE} = \frac{3600 \cdot SF}{\sqrt[24]{\prod_{i=1}^{22} QI(i,0) \cdot \prod_{j=1}^2 RI(j,0)}}$$

QI(i,0): timing interval of query Qi
RI(j,0): timing interval of refresh function RFj
- **TPC-H Throughput Numerical Quantity:**
$$\text{Throughput@SIZE} = (S \cdot 22 \cdot 3600) / T_s \cdot SF$$

T_s: measurement interval
S: number of query streams in the throughput test
- **TPC-H Composite Query-Per-Hour Performance Metric:**
$$\text{QphH@SIZE} = \sqrt{\text{Power@SIZE} \cdot \text{Throughput@SIZE}}$$
- **TPC-H Price/Performance Metric:**
$$\text{Price - per - QphH@SIZE} = \$ / \text{QphH@SIZE}$$

TPC-H Results: 3 TB

3,000 GB Results											
Rank	Company	System	QphH	Price / QphH	Watts / KQphH	System Availability	Database	Operating System	Date Submitted	Cluster	
1	FUJITSU	PRIMERGY RX300 S4	1,608,920	1.36 USD	NR	08/01/08	EXASOL EXASolution 2.1	EXASOL EXACluster OS 2.1	06/02/08	Y	
2	ORACLE	Sun SPARC Enterprise M9000 Server	198,907	16.58 USD	NR	04/05/11	Oracle Database 11g Release 2 Enterprise Edt.	Oracle Solaris 10	10/05/10	N	
3	hp invent	HP ProLiant DL980 G7	162,601	2.68 USD	NR	10/13/10	Microsoft SQL Server 2008 R2 Enterprise Edition	Microsoft Windows Server 2008 R2 Enterprise Edition	06/21/10	N	
4	IBM	IBM Power 595 Model 9119-FHA	156,537	20.60 USD	NR	11/24/09	Sybase IQ Single Application Server Edition v.15.1 ESD #1.2	AIX Version 6.1	11/24/09	N	
5	ORACLE	Sun Fire[TM] E25K server	114,713	36.68 USD	NR	04/09/07	Oracle Database 10g R2 Enterprise Edt w/Partitioning	Sun Solaris 10	04/09/07	N	
6	hp invent	HP BladeSystem ProLiant BL25p cluster 64p DC	110,576	37.80 USD	NR	06/08/06	Oracle Database 10g R2 Enterprise Edt w/Partitioning	Red Hat Enterprise Linux 4 ES	06/08/06	Y	
7	ORACLE	Sun Fire[TM] E25K server	105,430	54.87 USD	NR	01/27/06	Oracle Database 10g R2 Enterprise Edt w/Partitioning	Sun Solaris 10	01/27/06	N	
8	UNISYS	Unisys ES7000 Model 7600R Enterprise Server(16s)	102,778	21.05 USD	NR	05/06/10	Microsoft SQL Server 2008 R2 Datacenter Edition	Microsoft Windows Server 2008 R2 Datacenter Edition	11/02/09	N	
9	IBM	IBM eServer p5 595	100,512	53.00 USD	NR	03/01/06	Oracle 10g Enterprise Ed R2 w/ Partitioning	IBM AIX 5L V5.3	09/19/05	N	
10	hp invent	HP Integrity Superdome	60,359	32.60 USD	NR	05/21/07	Microsoft SQL Server 2005 Enterprise Edt Itanium SP2	Microsoft Windows Server 2003 Datacenter Ed.(64-bit)SP1	05/21/07	N	

TPC-H Results: 10TB + 30TB

10,000 GB Results

Rank	Company	System	QphH	Price/QphH	Watts/KQphH	System Availability	Database	Operating System	Date Submitted	Cluster
1	 IBM	IBM System p 570	343,551	32.89 USD	NR	04/15/08	IBM DB2 Warehouse 9.5	IBM AIX 5L V5.3	10/15/07	Y
2	 hp invent	HP Integrity Superdome/Dual-Core Itanium/1.6 GHz	208,457	27.97 USD	NR	09/10/08	Oracle Database 11g Enterprise Edition	HP-UX 11i v3 64 bit	03/10/08	N
3	 IBM	IBM System p5 575 with DB2 UDB 8.2	180,108	47.00 USD	NR	08/30/06	IBM DB2 UDB 8.2	IBM AIX 5L V5.3	07/14/06	Y
4	 hp invent	HP Integrity Superdome-DC Itanium2/1.6GHz/64p/128c	171,380	32.91 USD	NR	04/01/07	Oracle Database 10g R2 Enterprise Edt w/Partitioning	HP-UX 11i v3 64 bit	11/30/06	N
5	 ORACLE	Sun Fire[TM] E25K server	108,099	53.80 USD	NR	01/23/06	Oracle 10g Enterprise Ed R2 w/ Partitioning	Sun Solaris 10	11/29/05	N
6	 hp invent	HP Integrity Superdome - Itanium2/1.5 GHz-128p/128	86,282	161.24 USD	NR	04/06/05	Oracle Database 10g Enterprise Edition	HP UX 11.i V2 64 bit	10/07/04	Y
7	 UNISYS	Unisys ES7000 Model 7600R Enterprise Server(16s)	80,172	18.95 USD	NR	02/17/09	Microsoft SQL Server 2008 Enterprise x64 Edition	Microsoft Windows Server 2008 Datacenter x64 Edition	02/17/09	N
8	 hp invent	HP Integrity Superdome	63,650	38.54 USD	NR	08/30/08	Microsoft SQL Server 2008 Enterprise Edition	Microsoft Windows Server 2008 Intanium based Systems	02/27/08	N
9	 hp invent	HP Integrity Superdome - Itanium2/1.5 GHz-64p/64c	49,104	118.13 USD	NR	03/25/04	Oracle Database 10g Enterprise Edition	HP UX 11.i, 64-bit Base OS	01/05/04	N

30,000 GB Results

Rank	Company	System	QphH	Price/QphH	Watts/KQphH	System Availability	Database	Operating System	Date Submitted	Cluster
1	 hp invent	HP Integrity Superdome - Itanium2/1.6 GHz/18MB iL3	150,960	46.69 USD	NR	06/18/07	Oracle Database 10g release2 Enterprise Edt	HP-UX 11i v3 64 bit	06/18/07	N

*** indicates a duplicate result

'NR' in the Watts/KQphH column indicates that no energy data was reported for that benchmark.

TPC-R

- Decision support benchmark similar to TPC-H.
- Allows additional optimizations based on advance knowledge of the queries.
- Business oriented queries and concurrent data modifications.
- Performance metric: TPC-R Composite Query-per-Hour Performance Metric (QphR@Size)-
- TPC-R Price/Performance metric is expressed as \$/QphR@Size.

Summary

- SQL supports typical OLAP queries by:
 - ROLLUP, CUBE, WINDOWs, Ranking, OLAP functions
- SQL/MM Data Mining supports:
 - build data mining models (clustering, classification, regression, AR)
 - test and apply data mining models
 - import and export data mining models
- Database systems support performance of data warehouse systems by, e.g.:
 - data partitioning
 - materialized views
 - indexes: bitmap indexes, join indexes, ...
 - query processing for star queries
- Benchmark for data warehousing: TPC-H

Papers & Books

- [Bay97] R. Bayer: The Universal B-Tree for Multidimensional Indexing: general Concepts In: Proc. of the Worldwide Computing and Its Applications, International Conference. WWCA '97, Tsukuba, Japan, March 10-11, 1997.
- [BP+97] E. Baralis, S. Paraboschi, E. Teniente: Materialized View Selection in a Multidimensional Database. In: Proc. of the 23rd Int. Conf. on Very Large Databases, Athens, Greece, 1997.
- [HR+96] V. Harinarayan, A. Rajaraman, J. Ullman: Implementing Data Cubes Efficiently. In: Proc. of the Int. Conf. on Management of Data, Montreal, Quebec, 1996.
- [ScOu82] P. Scheuermann, M. Ouksel: Multidimensional B-trees for Associative Searching in Database Systems. In: Information Systems, vol. 7, no. 2, 1982.
- [MS02] J. Melton, A. Simon: SQL:1999, Understanding Relational Language Components. Morgan Kaufmann, 2002.
- [Mel03] J. Melton: SQL:1999, Understanding Object-Relational and other Advanced Features. Morgan Kaufmann, 2003.