### Data Warehousing and On-line Analytical Processing

- Data Warehouse: Basic Concepts
- Data Warehouse Modeling: Data Cube and OLAP
- Data Warehouse Design and Usage
- Data Warehouse Implementation
- Data Generalization by Attribute-Oriented Induction
- Summary

## What is a Data Warehouse?

- Defined in many different ways, but not rigorously.
  - A decision support database that is maintained separately from the organization's operational database
  - Support information processing by providing a solid platform of consolidated, historical data for analysis.
- "A data warehouse is a <u>subject-oriented</u>, <u>integrated</u>, <u>time-variant</u>, and <u>nonvolatile</u> collection of data in support of management's decision-making process."—W. H. Inmon
- Data warehousing:
  - The process of constructing and using data warehouses

### Data Warehouse—Subject-Oriented

- Organized around major subjects, such as customer, product, sales
- Focusing on the modeling and analysis of data for decision makers, not on daily operations or transaction processing
- Provide a simple and concise view around particular subject issues by excluding data that are not useful in the decision support process

# **Data Warehouse—Integrated**

- Constructed by integrating multiple, heterogeneous data sources
  - relational databases, flat files, on-line transaction records
- Data cleaning and data integration techniques are applied.
  - Ensure consistency in naming conventions, encoding structures, attribute measures, etc. among different data sources
    - E.g., Hotel price: currency, tax, breakfast covered, etc.
  - When data is moved to the warehouse, it is converted.

## **Data Warehouse—Time Variant**

- The time horizon for the data warehouse is significantly longer than that of operational systems
  - Operational database: current value data
  - Data warehouse data: provide information from a historical perspective (e.g., past 5-10 years)
- Every key structure in the data warehouse
  - Contains an element of time, explicitly or implicitly
  - But the key of operational data may or may not contain "time element"

## **Data Warehouse–Nonvolatile**

- A physically separate store of data transformed from the operational environment
- Operational update of data does not occur in the data warehouse environment
  - Does not require transaction processing, recovery, and concurrency control mechanisms
  - Requires only two operations in data accessing:

initial loading of data and access of data

### **OLTP vs. OLAP**

	OLTP	OLAP
users	clerk, IT professional	knowledge worker
function	day to day operations	decision support
DB design	application-oriented	subject-oriented
data	current, up-to-date detailed, flat relational isolated	historical, summarized, multidimensional integrated, consolidated
usage	repetitive	ad-hoc
access	read/write index/hash on prim. key	lots of scans
unit of work	short, simple transaction	complex query
# records accessed	tens	millions
#users	thousands	hundreds
DB size	100MB-GB	100GB-TB
metric	transaction throughput	query throughput, response

# Why a Separate Data Warehouse?

- High performance for both systems
  - DBMS— tuned for OLTP: access methods, indexing, concurrency control, recovery
  - Warehouse—tuned for OLAP: complex OLAP queries, multidimensional view, consolidation
- Different functions and different data:
  - <u>missing data</u>: Decision support requires historical data which operational DBs do not typically maintain
  - <u>data consolidation</u>: DS requires consolidation (aggregation, summarization) of data from heterogeneous sources
  - <u>data quality</u>: different sources typically use inconsistent data representations, codes and formats which have to be reconciled
- Note: There are more and more systems which perform OLAP analysis directly on relational databases

#### **Data Warehouse: A Multi-Tiered Architecture**



## **Three Data Warehouse Models**

#### Enterprise warehouse

- collects all of the information about subjects spanning the entire organization
- Data Mart
  - a subset of corporate-wide data that is of value to a specific groups of users. Its scope is confined to specific, selected groups, such as marketing data mart
    - Independent vs. dependent (directly from warehouse) data mart
- Virtual warehouse
  - A set of views over operational databases
  - Only some of the possible summary views may be materialized

### Extraction, Transformation, and Loading (ETL)

#### Data extraction

get data from multiple, heterogeneous, and external sources

#### Data cleaning

detect errors in the data and rectify them when possible

#### Data transformation

 convert data from legacy or host format to warehouse format

#### Load

 sort, summarize, consolidate, compute views, check integrity, and build indicies and partitions

#### Refresh

propagate the updates from the data sources to the warehouse

# **Metadata Repository**

- **Meta data** is the data defining warehouse objects. It stores:
- Description of the structure of the data warehouse
  - schema, view, dimensions, hierarchies, derived data defn, data mart locations and contents
- Operational meta-data
  - data lineage (history of migrated data and transformation path), currency of data (active, archived, or purged), monitoring information (warehouse usage statistics, error reports, audit trails)
- The algorithms used for summarization
- The mapping from operational environment to the data warehouse
- Data related to system performance
  - warehouse schema, view and derived data definitions
- Business data
  - business terms and definitions, ownership of data, charging policies

### Chapter 4: Data Warehousing and On-line Analytical Processing

- Data Warehouse: Basic Concepts
- Data Warehouse Modeling: Data Cube and OLAP
- Data Warehouse Design and Usage
- Data Warehouse Implementation
- Data Generalization by Attribute-Oriented Induction
- Summary

### From Tables and Spreadsheets to Data Cubes

- 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
  - Dimension tables, such as item (item\_name, brand, type), or time(day, week, month, quarter, year)
  - Fact table contains measures (such as dollars\_sold) and keys to each of the related dimension tables
- 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 a data cube.

### **Cube: A Lattice of Cuboids**



### **Conceptual Modeling of Data Warehouses**

- Modeling data warehouses: dimensions & measures
  - <u>Star schema</u>: A fact table in the middle connected to a set of dimension tables
  - <u>Snowflake schema</u>: A refinement of star schema where some dimensional hierarchy is normalized into a set of smaller dimension tables, forming a shape similar to snowflake
  - Fact constellations: Multiple fact tables share dimension tables, viewed as a collection of stars, therefore called galaxy schema or fact constellation

### **Example of Star Schema**



## **Example of Snowflake Schema**



#### Example of Fact Constellation



### A Concept Hierarchy: Dimension (location)



### **Data Cube Measures: Three Categories**

- <u>Distributive</u>: if the result derived by applying the function to *n* aggregate values is the same as that derived by applying the function on all the data without partitioning
  E.g., count(), sum(), min(), max()
- <u>Algebraic</u>: if it can be computed by an algebraic function with *M* arguments (where *M* is a bounded integer), each of which is obtained by applying a distributive aggregate function
  - E.g., avg(), min\_N(), standard\_deviation()
- <u>Holistic</u>: if there is no constant bound on the storage size needed to describe a subaggregate.
  - E.g., median(), mode(), rank()

#### View of Warehouses and Hierarchies



### **Multidimensional Data**

 Sales volume as a function of product, month, and region



Month

Dimensions: *Product, Location, Time* Hierarchical summarization paths



### **A Sample Data Cube**



### Cuboids Corresponding to the Cube



# **Typical OLAP Operations**

- 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)



# **A Star-Net Query Model**



### **Browsing a Data Cube**



### Chapter 4: Data Warehousing and On-line Analytical Processing

- Data Warehouse: Basic Concepts
- Data Warehouse Modeling: Data Cube and OLAP
- Data Warehouse Design and Usage
- Data Warehouse Implementation
- Data Generalization by Attribute-Oriented Induction
- Summary

### **Design of Data Warehouse: A Business Analysis Framework**

- Four views regarding the design of a data warehouse
  - Top-down view
    - allows selection of the relevant information necessary for the data warehouse
  - Data source view
    - exposes the information being captured, stored, and managed by operational systems
  - Data warehouse view
    - consists of fact tables and dimension tables
  - Business query view
    - sees the perspectives of data in the warehouse from the view of end-user

### Data Warehouse Design Process

#### Top-down, bottom-up approaches or a combination of both

- <u>Top-down</u>: Starts with overall design and planning (mature)
- <u>Bottom-up</u>: Starts with experiments and prototypes (rapid)

#### From software engineering point of view

- <u>Waterfal</u>I: structured and systematic analysis at each step before proceeding to the next
- <u>Spiral</u>: rapid generation of increasingly functional systems, short turn around time, quick turn around

#### Typical data warehouse design process

- Choose a business process to model, e.g., orders, invoices, etc.
- Choose the <u>grain</u> (atomic level of data) of the business process
- Choose the dimensions that will apply to each fact table record
- Choose the measure that will populate each fact table record

# Development: A Recommended Approach



## **Data Warehouse Usage**

- Three kinds of data warehouse applications
  - Information processing
    - supports querying, basic statistical analysis, and reporting using crosstabs, tables, charts and graphs
  - Analytical processing
    - multidimensional analysis of data warehouse data
    - supports basic OLAP operations, slice-dice, drilling, pivoting
  - Data mining
    - knowledge discovery from hidden patterns
    - supports associations, constructing analytical models, performing classification and prediction, and presenting the mining results using visualization tools

# (OLAP) to On Line Analytical Mining (OLAM)

- Why online analytical mining?
  - High quality of data in data warehouses
    - DW contains integrated, consistent, cleaned data
  - Available information processing structure surrounding data warehouses
    - ODBC, OLEDB, Web accessing, service facilities, reporting and OLAP tools
  - OLAP-based exploratory data analysis
    - Mining with drilling, dicing, pivoting, etc.
  - On-line selection of data mining functions
    - Integration and swapping of multiple mining functions, algorithms, and tasks

### Chapter 4: Data Warehousing and On-line Analytical Processing

- Data Warehouse: Basic Concepts
- Data Warehouse Modeling: Data Cube and OLAP
- Data Warehouse Design and Usage
- Data Warehouse Implementation
- Data Generalization by Attribute-Oriented Induction
- Summary
# Efficient Data Cube Computation

- Data cube can be viewed as a lattice of cuboids
  - The bottom-most cuboid is the base cuboid
  - The top-most cuboid (apex) contains only one cell
  - How many cuboids in an n-dimensional cube with L levels?  $T = \prod_{i=1}^{n} (L_i + 1)$
- Materialization of data cube
  - Materialize <u>every</u> (cuboid) (full materialization), <u>none</u> (no materialization), or <u>some</u> (partial <u>materialization</u>)
  - Selection of which cuboids to materialize
    - Based on size, sharing, access frequency, etc.

# The "Compute Cube" Operator

Cube definition and computation in DMQL define cube sales [item, city, year]: sum (sales\_in\_dollars) compute cube sales

 Transform it into a SQL-like language (with a new operator cube by, introduced by Gray et al.'96)

SELECT item, city, year, SUM (amount) FROM SALES CUBE BY item, city, year Need compute the following Group-Bys (date, product, customer), (date, product),(date, customer), (product, (city, item) (city, item) (city, year) (city, item) (city, year) (city, yea

(city, item, year)

#### Indexing OLAP Data: Bitmap Index

- Index on a particular column
- Each value in the column has a bit vector: bit-op is fast
- The length of the bit vector: # of records in the base table
- The *i*-th bit is set if the *i*-th row of the base table has the value for the indexed column
- not suitable for high cardinality domains
- A recent bit compression technique, Word-Aligned Hybrid (WAH), makes it work for high cardinality domain as well [Wu, et al. TODS'06]

Base table			Inde	x on	Regio	Index on Type			
Cust	Region	Туре	<b>RecID</b>	Asia	Europe	<b>America</b>	RecID	Retail	Dealer
C1	Asia	Retail	1	1	0	0	1	1	0
C2	Europe	Dealer	2	0	1	0	2	0	1
C3	Asia	Dealer	3	1	0	0	3	0	1
C4	America	Retail	4	0	0	1	4	1	0
C5	Europe	Dealer	5	0	1	0	5	0	1

# **Indexing OLAP Data: Join Indices**

- Join index: JI(R-id, S-id) where R (R-id, ...) ▷⊲ S (S-id, ...)
- Traditional indices map the values to a list of record ids
  - It materializes relational join in JI file and speeds up relational join
- In data warehouses, join index relates the values of the <u>dimensions</u> of a start schema to <u>rows</u> in the fact table.
  - E.g. fact table: Sales and two dimensions city and product
    - A join index on *city* maintains for each distinct city a list of R-IDs of the tuples recording the Sales in the city
  - Join indices can span multiple dimensions



# **Efficient Processing OLAP Queries**

- **Determine which operations** should be performed on the available cuboids
  - Transform drill, roll, etc. into corresponding SQL and/or OLAP operations,
     e.g., dice = selection + projection
- Determine which materialized cuboid(s) should be selected for OLAP op.
  - Let the query to be processed be on {*brand, province\_or\_state*} with the condition "*year = 2004*", and there are 4 materialized cuboids available:
    - 1) { year, item\_name, city}
    - 2) { year, brand, country}
    - 3) { year, brand, province\_or\_state }
    - 4) {*item\_name, province\_or\_state*} where *year = 2004*

Which should be selected to process the query?

Explore indexing structures and compressed vs. dense array structs in MOLAP

# **OLAP Server Architectures**

#### Relational OLAP (ROLAP)

- Use relational or extended-relational DBMS to store and manage warehouse data and OLAP middle ware
- Include optimization of DBMS backend, implementation of aggregation navigation logic, and additional tools and services
- Greater scalability
- Multidimensional OLAP (MOLAP)
  - Sparse array-based multidimensional storage engine
  - Fast indexing to pre-computed summarized data
- Hybrid OLAP (HOLAP) (e.g., Microsoft SQLServer)
  - Flexibility, e.g., low level: relational, high-level: array
- Specialized SQL servers (e.g., Redbricks)
  - Specialized support for SQL queries over star/snowflake schemas

#### Chapter 4: Data Warehousing and On-line Analytical Processing

- Data Warehouse: Basic Concepts
- Data Warehouse Modeling: Data Cube and OLAP
- Data Warehouse Design and Usage
- Data Warehouse Implementation
- Data Generalization by Attribute-Oriented Induction
- Summary

#### Attribute-Oriented Induction

- Proposed in 1989 (KDD '89 workshop)
- Not confined to categorical data nor particular measures
- How it is done?
  - Collect the task-relevant data (*initial relation*) using a relational database query
  - Perform generalization by <u>attribute removal</u> or <u>attribute generalization</u>
  - Apply aggregation by merging identical, generalized tuples and accumulating their respective counts
  - Interaction with users for knowledge presentation

# Example

Example: Describe general characteristics of graduate students in the University database

Step 1. Fetch relevant set of data using an SQL statement, e.g.,

Select \* (i.e., name, gender, major, birth\_place, birth\_date, residence, phone#, gpa)

from student

where student\_status in {"Msc", "MBA", "PhD" }

- Step 2. Perform attribute-oriented induction
- Step 3. Present results in generalized relation, cross-tab, or rule forms

### **Class Characterization: An Example**

	Name	Gen	der Ma	ajor	Birth-Place Vancouver,BC, Canada Montreal, Que, Canada Seattle, WA, USA 		Birth_date C, 8-12-76 e, 28-7-75 A 25-8-70 		Residence 3511 Main St., Richmond 345 1st Ave., Richmond 125 Austin Ave., Burnaby 		Phone #	GPA 3.67 3.70 3.83 
Initial Relation	Jim Woodma Scott Lachance	M n M	C C	S S							687-4598 253-9106	
	Laura Le 	e F	Ph 	ysics							420-5232 	
	Removed	Retai	ned Sci Bu	,Eng, s	Country		Age	range	City	7	Removed	Excl, VG,
		Gender	ler Major		Birth_region		Age_range Res		ence GPA		Count	
Prime Generalized Relation		М	Science	ence Canada		20-	-25 Richr		nond	Very-good	1 16	
		F	Science	e Foreign		25-	-30	Burnaby		Excellent	22	
		•••					••	•••				
			Ge	Birth_Region Gender			Canada         Foreig           16         14           10         22		n	Total		
				M F		16			30			
			10						32			
				Т	<b>`otal</b>	26		36		62		

#### Basic Principles of Attribute-Oriented Induction

- <u>Data focusing</u>: task-relevant data, including dimensions, and the result is the *initial relation*
- Attribute-removal: remove attribute A if there is a large set of distinct values for A but (1) there is no generalization operator on A, or (2) A's higher level concepts are expressed in terms of other attributes
- Attribute-generalization: If there is a large set of distinct values for A, and there exists a set of generalization operators on A, then select an operator and generalize A
- <u>Attribute-threshold control</u>: typical 2-8, specified/default
- <u>Generalized relation threshold control</u>: control the final relation/rule size

#### Attribute-Oriented Induction: Basic Algorithm

- <u>InitialRel</u>: Query processing of task-relevant data, deriving the *initial relation*.
- PreGen: Based on the analysis of the number of distinct values in each attribute, determine generalization plan for each attribute: removal? or how high to generalize?
- <u>PrimeGen</u>: Based on the PreGen plan, perform generalization to the right level to derive a "prime generalized relation", accumulating the counts.
- Presentation: User interaction: (1) adjust levels by drilling,
   (2) pivoting, (3) mapping into rules, cross tabs,
   visualization presentations.

#### Presentation of Generalized Results

- Generalized relation:
  - Relations where some or all attributes are generalized, with counts or other aggregation values accumulated.
- Cross tabulation:
  - Mapping results into cross tabulation form (similar to contingency tables).
  - Visualization techniques:
  - Pie charts, bar charts, curves, cubes, and other visual forms.
- Quantitative characteristic rules:
  - Mapping generalized result into characteristic rules with quantitative information associated with it, e.g.,

 $grad(x) \land male(x) \Rightarrow$ 

 $birth\_region(x) = "Canada"[t:53\%] \lor birth\_region(x) = "foreign"[t:47\%].$ 

# **Mining Class Comparisons**

- <u>Comparison</u>: Comparing two or more classes
- Method:
  - Partition the set of relevant data into the target class and the contrasting class(es)
  - Generalize both classes to the same high level concepts
  - Compare tuples with the same high level descriptions
  - Present for every tuple its description and two measures
    - support distribution within single class
    - comparison distribution between classes
  - Highlight the tuples with strong discriminant features
- Relevance Analysis:
  - Find attributes (features) which best distinguish different classes

### Concept Description vs. Cube-Based OLAP

#### Similarity:

- Data generalization
- Presentation of data summarization at multiple levels of abstraction
- Interactive drilling, pivoting, slicing and dicing

#### Differences:

- OLAP has systematic preprocessing, query independent, and can drill down to rather low level
- AOI has automated desired level allocation, and may perform dimension relevance analysis/ranking when there are many relevant dimensions
- AOI works on the data which are not in relational forms

### Chapter 4: Data Warehousing and On-line Analytical Processing

- Data Warehouse: Basic Concepts
- Data Warehouse Modeling: Data Cube and OLAP
- Data Warehouse Design and Usage
- Data Warehouse Implementation
- Data Generalization by Attribute-Oriented Induction



# **Summary**

- Data warehousing: A multi-dimensional model of a data warehouse
  - A data cube consists of *dimensions* & *measures*
  - Star schema, snowflake schema, fact constellations
  - OLAP operations: drilling, rolling, slicing, dicing and pivoting
- Data Warehouse Architecture, Design, and Usage
  - Multi-tiered architecture
  - Business analysis design framework
  - Information processing, analytical processing, data mining, OLAM (Online Analytical Mining)
- Implementation: Efficient computation of data cubes
  - Partial vs. full vs. no materialization
  - Indexing OALP data: Bitmap index and join index
  - OLAP query processing
  - OLAP servers: ROLAP, MOLAP, HOLAP
- Data generalization: Attribute-oriented induction

# **References (I)**

- S. Agarwal, R. Agrawal, P. M. Deshpande, A. Gupta, J. F. Naughton, R. Ramakrishnan, and S. Sarawagi. On the computation of multidimensional aggregates. VLDB'96
- D. Agrawal, A. E. Abbadi, A. Singh, and T. Yurek. Efficient view maintenance in data warehouses. SIGMOD'97
- R. Agrawal, A. Gupta, and S. Sarawagi. Modeling multidimensional databases. ICDE'97
- S. Chaudhuri and U. Dayal. An overview of data warehousing and OLAP technology. ACM SIGMOD Record, 26:65-74, 1997
- E. F. Codd, S. B. Codd, and C. T. Salley. Beyond decision support. Computer World, 27, July 1993.
- J. Gray, et al. Data cube: A relational aggregation operator generalizing group-by, cross-tab and sub-totals. Data Mining and Knowledge Discovery, 1:29-54, 1997.
- A. Gupta and I. S. Mumick. Materialized Views: Techniques, Implementations, and Applications. MIT Press, 1999.
- J. Han. Towards on-line analytical mining in large databases. *ACM SIGMOD Record*, 27:97-107, 1998.
- V. Harinarayan, A. Rajaraman, and J. D. Ullman. Implementing data cubes efficiently. SIGMOD'96

# **References (II)**

- C. Imhoff, N. Galemmo, and J. G. Geiger. Mastering Data Warehouse Design: Relational and Dimensional Techniques. John Wiley, 2003
- W. H. Inmon. Building the Data Warehouse. John Wiley, 1996
- R. Kimball and M. Ross. The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling. 2ed. John Wiley, 2002
- P. O'Neil and D. Quass. Improved query performance with variant indexes. SIGMOD'97
- Microsoft. OLEDB for OLAP programmer's reference version 1.0. In http://www.microsoft.com/data/oledb/olap, 1998
- A. Shoshani. OLAP and statistical databases: Similarities and differences. PODS'00.
- S. Sarawagi and M. Stonebraker. Efficient organization of large multidimensional arrays. ICDE'94
- P. Valduriez. Join indices. ACM Trans. Database Systems, 12:218-246, 1987.
- J. Widom. Research problems in data warehousing. CIKM'95.
- K. Wu, E. Otoo, and A. Shoshani, Optimal Bitmap Indices with Efficient Compression, ACM Trans. on Database Systems (TODS), 31(1), 2006, pp. 1-38.

#### ALMA MATER

TO THY HAPPY CHILDREN OF THE FUTURE THOSE OF THE PAST SEND GREETINGS

20.0M1 101.510

ANTER DUNC

1



#### Chapter 4: Data Warehousing and On-line Analytical Processing

- Data Warehouse: Basic Concepts
  - (a) What Is a Data Warehouse?
  - (b) Data Warehouse: A Multi-Tiered Architecture
  - (c) Three Data Warehouse Models: Enterprise Warehouse, Data Mart, ad Virtual Warehouse
  - (d) Extraction, Transformation and Loading
  - (e) Metadata Repository
- Data Warehouse Modeling: Data Cube and OLAP
  - (a) Cube: A Lattice of Cuboids
  - (b) Conceptual Modeling of Data Warehouses
  - (c) Stars, Snowflakes, and Fact Constellations: Schemas for Multidimensional Databases
  - (d) Dimensions: The Role of Concept Hierarchy
  - (e) Measures: Their Categorization and Computation
  - (f) Cube Definitions in Database systems
  - (g) Typical OLAP Operations
  - (h) A Starnet Query Model for Querying Multidimensional Databases
- Data Warehouse Design and Usage
  - (a) Design of Data Warehouses: A Business Analysis Framework
  - (b) Data Warehouses Design Processes
  - (c) Data Warehouse Usage
  - (d) From On-Line Analytical Processing to On-Line Analytical Mining
- Data Warehouse Implementation
  - (a) Efficient Data Cube Computation: Cube Operation, Materialization of Data Cubes, and Iceberg Cubes
  - (b) Indexing OLAP Data: Bitmap Index and Join Index
  - (c) Efficient Processing of OLAP Queries
  - (d) OLAP Server Architectures: ROLAP vs. MOLAP vs. HOLAP
- Data Generalization by Attribute-Oriented Induction
  - (a) Attribute-Oriented Induction for Data Characterization
  - (b) Efficient Implementation of Attribute-Oriented Induction
  - (c) Attribute-Oriented Induction for Class Comparisons
  - (d) Attribute-Oriented Induction vs. Cube-Based OLAP
- Summary

# **Compression of Bitmap Indices**

- Bitmap indexes must be compressed to reduce I/O costs and minimize CPU usage—majority of the bits are 0's
- Two compression schemes:
  - Byte-aligned Bitmap Code (BBC)
  - Word-Aligned Hybrid (WAH) code
- Time and space required to operate on compressed bitmap is proportional to the total size of the bitmap
- Optimal on attributes of low cardinality as well as those of high cardinality.
- WAH out performs BBC by about a factor of two



# **Concepts and Techniques**

(3<sup>rd</sup> ed.)

#### - Chapter 5 -

Jiawei Han, Micheline Kamber, and Jian Pei University of Illinois at Urbana-Champaign & Simon Fraser University ©2011 Han, Kamber & Pei. All rights reserved.



# **Chapter 5: Data Cube Technology**

- Data Cube Computation: Preliminary Concepts
- Data Cube Computation Methods
- Processing Advanced Queries by Exploring Data
   Cube Technology
- Multidimensional Data Analysis in Cube Space
- Summary

# **Data Cube: A Lattice of Cuboids**



time, item, location, supplierc

# **Data Cube: A Lattice of Cuboids**



- Base vs. aggregate cells; ancestor vs. descendant cells; parent vs. child cells
  - 1. (9/15, milk, Urbana, Dairy\_land)
  - 2. (9/15, milk, Urbana, \*)
  - 3. (\*, milk, Urbana, \*)
  - 4. (\*, milk, Urbana, \*)
  - 5. (\*, milk, Chicago, \*)
  - 6. (\*, milk, \*, \*)

# **Cube Materialization: Full Cube vs. Iceberg Cube**

Full cube vs. iceberg cube

 compute cube sales iceberg as
 select month, city, customer group, count(\*)
 from salesInfo
 cube by month, city, customer group
 having count(\*) >= min support



- Computing *only* the cuboid cells whose measure satisfies the iceberg condition
- Only a small portion of cells may be "above the water" in a sparse cube
- Avoid explosive growth: A cube with 100 dimensions
  - 2 base cells: (a1, a2, ...., a100), (b1, b2, ..., b100)
  - How many aggregate cells if "having count >= 1"?
  - What about "having count >= 2"?

### Iceberg Cube, Closed Cube & Cube Shell

- Is iceberg cube good enough?
  - 2 base cells: {(a<sub>1</sub>, a<sub>2</sub>, a<sub>3</sub>..., a<sub>100</sub>):10, (a<sub>1</sub>, a<sub>2</sub>, b<sub>3</sub>, ..., b<sub>100</sub>):10}
  - How many cells will the iceberg cube have if having count(\*) >= 10? Hint: A huge but tricky number!
- Close cube:
  - Closed cell c: if there exists no cell d, s.t. d is a descendant of c, and d has the same measure value as c.
  - Closed cube: a cube consisting of only closed cells
  - What is the closed cube of the above base cuboid? Hint: only 3 cells
- Cube Shell
  - Precompute only the cuboids involving a small # of dimensions,
     e.g., 3 For (A<sub>1</sub>, A<sub>2</sub>, ... A<sub>10</sub>), how many combinations to compute?
  - More dimension combinations will need to be computed on the fly

# **Roadmap for Efficient Computation**

- General cube computation heuristics (Agarwal et al.'96)
- Computing full/iceberg cubes: 3 methodologies
  - Bottom-Up: Multi-Way array aggregation (Zhao, Deshpande & Naughton, SIGMOD'97)
  - Top-down:
    - BUC (Beyer & Ramarkrishnan, SIGMOD'99)
    - H-cubing technique (Han, Pei, Dong & Wang: SIGMOD'01)
  - Integrating Top-Down and Bottom-Up:
    - Star-cubing algorithm (Xin, Han, Li & Wah: VLDB'03)
- High-dimensional OLAP: A Minimal Cubing Approach (Li, et al. VLDB'04)
- Computing alternative kinds of cubes:
  - Partial cube, closed cube, approximate cube, etc.

### General Heuristics (Agarwal et al. VLDB'96)

- Sorting, hashing, and grouping operations are applied to the dimension attributes in order to reorder and cluster related tuples
- Aggregates may be computed from previously computed aggregates, rather than from the base fact table
  - Smallest-child: computing a cuboid from the smallest, previously computed cuboid
  - Cache-results: caching results of a cuboid from which other cuboids are computed to reduce disk I/Os
  - Amortize-scans: computing as many as possible cuboids at the same time to amortize disk reads
  - Share-sorts: sharing sorting costs cross multiple cuboids when sort-based method is used
  - Share-partitions: sharing the partitioning cost across multiple cuboids when hash-based algorithms are used

# **Chapter 5: Data Cube Technology**

- Data Cube Computation: Preliminary Concepts
- Data Cube Computation Methods
- Processing Advanced Queries by Exploring Data
   Cube Technology
- Multidimensional Data Analysis in Cube Space
- Summary

# **Data Cube Computation Methods**

- Multi-Way Array Aggregation
- BUC
- Star-Cubing
- High-Dimensional OLAP

# **Multi-Way Array Aggregation**

- Array-based "bottom-up" algorithm
- Using multi-dimensional chunks
- No direct tuple comparisons
- Simultaneous aggregation on multiple dimensions
- Intermediate aggregate values are reused for computing ancestor cuboids
- Cannot do *Apriori* pruning: No iceberg optimization



#### Multi-way Array Aggregation for Cube Computation (MOLAP)

- Partition arrays into chunks (a small subcube which fits in memory).
- Compressed sparse array addressing: (chunk\_id, offset)
- Compute aggregates in "multiway" by visiting cube cells in the order which minimizes the # of times to visit each cell, and reduces memory access and storage cost.



What is the best traversing order to do multi-way aggregation?

#### Multi-way Array Aggregation for Cube Computation (3-D to 2-D)


#### Multi-way Array Aggregation for Cube Computation (2-D to 1-D)



\*

\*

\*

\*

\*

В

73

#### Multi-Way Array Aggregation for Cube Computation (Method Summary)

- Method: the planes should be sorted and computed according to their size in ascending order
  - Idea: keep the smallest plane in the main memory, fetch and compute only one chunk at a time for the largest plane
- Limitation of the method: computing well only for a small number of dimensions
  - If there are a large number of dimensions, "top-down" computation and iceberg cube computation methods can be explored

# **Data Cube Computation Methods**

Multi-Way Array Aggregation



- Star-Cubing
- High-Dimensional OLAP

# **Bottom-Up Computation (BUC)**

- BUC (Beyer & Ramakrishnan, SIGMOD'99)
- Bottom-up cube computation (Note: top-down in our view!)
- Divides dimensions into partitions and facilitates iceberg pruning
  - If a partition does not satisfy min\_sup, its descendants can be pruned
  - If *minsup* = 1 ⇒compute full CUBE!
- No simultaneous aggregation



# **BUC: Partitioning**

- Usually, entire data set can't fit in main memory
- Sort *distinct* values
  - partition into blocks that fit
- Continue processing
- Optimizations
  - Partitioning
    - External Sorting, Hashing, Counting Sort
  - Ordering dimensions to encourage pruning
    - Cardinality, Skew, Correlation
  - Collapsing duplicates
    - Can't do holistic aggregates anymore!



# **Data Cube Computation Methods**

- Multi-Way Array Aggregation
- BUC
- Star-Cubing
- High-Dimensional OLAP

# **Star-Cubing: An Integrating Method**

 D. Xin, J. Han, X. Li, B. W. Wah, Star-Cubing: Computing Iceberg Cubes by Top-Down and Bottom-Up Integration, VLDB'03

#### Explore shared dimensions

- E.g., dimension A is the shared dimension of ACD and AD
- ABD/AB means cuboid ABD has shared dimensions AB
- Allows for shared computations
  - e.g., cuboid AB is computed simultaneously as ABD
- Aggregate in a top-down manner but with the bottom-up sub-layer underneath which will allow Apriori pruning
- Shared dimensions grow in bottom-up fashion



# **Iceberg Pruning in Shared Dimensions**

- Anti-monotonic property of shared dimensions
  - If the measure is *anti-monotonic*, and if the aggregate value on a shared dimension does not satisfy the *iceberg condition*, then all the cells extended from this shared dimension cannot satisfy the condition either
- Intuition: if we can compute the shared dimensions before the actual cuboid, we can use them to do Apriori pruning
- Problem: how to prune while still aggregate simultaneously on multiple dimensions?

# **Cell Trees**



- Traverse the tree to retrieve
  - a particular tuple

# **Star Attributes and Star Nodes**

- Intuition: If a single-dimensional aggregate on an attribute value p does not satisfy the iceberg condition, it is useless to distinguish them during the iceberg computation
  - E.g., b<sub>2</sub>, b<sub>3</sub>, b<sub>4</sub>, c<sub>1</sub>, c<sub>2</sub>, c<sub>4</sub>, d<sub>1</sub>, d<sub>2</sub>,
     d<sub>3</sub>
- Solution: Replace such attributes by a \*. Such attributes are <u>star</u> <u>attributes</u>, and the corresponding nodes in the cell tree are <u>star nodes</u>

А	В	С	D	Count
a1	b1	c1	d1	1
a1	b1	c4	d3	1
a1	b2	c2	d2	1
a2	b3	c3	d4	1
a2	b4	c3	d4	1

# **Example: Star Reduction**

- Suppose minsup = 2
- Perform one-dimensional aggregation. Replace attribute values whose count < 2 with \*. And collapse all \*'s together
- Resulting table has all such attributes replaced with the starattribute
- With regards to the iceberg computation, this new table is a *lossless compression* of the original table

А	В	С	D	Count
a1	b1	*	*	1
a1	b1	*	*	1
a1	*	*	*	1
a2	*	c3	d4	1
a2	*	c3	d4	1



# **Star Tree**

- Given the new compressed table, it is possible to construct the corresponding cell tree—called <u>star tree</u>
- Keep a <u>star table</u> at the side for easy lookup of star attributes
- The star tree is a *lossless compression* of the original cell tree



#### Star-Cubing Algorithm—DFS on Lattice Tree





#### Star-Cubing Algorithm—DFS on Star-Tree



### Multi-Way Star-Tree Aggregation

- Start depth-first search at the root of the base star tree
- At each new node in the DFS, create corresponding star tree that are descendents of the current tree according to the integrated traversal ordering
  - E.g., in the base tree, when DFS reaches a1, the ACD/A tree is created
  - When DFS reaches b\*, the ABD/AD tree is created
- The counts in the base tree are carried over to the new trees

# Multi-Way Aggregation (2)

- When DFS reaches a leaf node (e.g., d\*), start backtracking
- On every backtracking branch, the count in the corresponding trees are output, the tree is destroyed, and the node in the base tree is destroyed
- Example
  - When traversing from d\* back to c\*, the alb\*c\*/alb\*c\* tree is output and destroyed
  - When traversing from c\* back to b\*, the alb\*D/alb\* tree is output and destroyed
  - When at b\*, jump to b1 and repeat similar process

# **Data Cube Computation Methods**

- Multi-Way Array Aggregation
- BUC
- Star-Cubing
- High-Dimensional OLAP

## **The Curse of Dimensionality**

- None of the previous cubing method can handle high dimensionality!
- A database of 600k tuples. Each dimension has cardinality of 100 and *zipf* of 2.



### **Motivation of High-D OLAP**

- X. Li, J. Han, and H. Gonzalez, High-Dimensional OLAP: A Minimal Cubing Approach, VLDB'04
- Challenge to current cubing methods:
  - The "curse of dimensionality" problem
  - Iceberg cube and compressed cubes: only delay the inevitable explosion
  - Full materialization: still significant overhead in accessing results on disk
- High-D OLAP is needed in applications
  - Science and engineering analysis
  - Bio-data analysis: thousands of genes
  - Statistical surveys: hundreds of variables

# **Fast High-D OLAP with Minimal Cubing**

- <u>Observation</u>: OLAP occurs only on a small subset of dimensions at a time
- Semi-Online Computational Model
  - 1. Partition the set of dimensions into **shell fragments**
  - 2. Compute data cubes for each shell fragment while retaining **inverted indices** or **value-list indices**
  - 3. Given the pre-computed **fragment cubes**, dynamically compute cube cells of the highdimensional data cube *online*

### **Properties of Proposed Method**

- Partitions the data vertically
- Reduces high-dimensional cube into a set of lower dimensional cubes
- Online re-construction of original high-dimensional space
- Lossless reduction
- Offers tradeoffs between the amount of pre-processing and the speed of online computation

#### **Example Computation**

Let the cube aggregation function be count

tid	А	В	С	D	E
1	al	b1	c1	d1	e1
2	al	b2	c1	d2	e1
3	al	b2	c1	d1	e2
4	a2	b1	c1	d1	e2
5	a2	b1	c1	d1	e3

Divide the 5 dimensions into 2 shell fragments:
(A, B, C) and (D, E)

#### **1-D Inverted Indices**

#### Build traditional invert index or RID list

Attribute Value	TID List	List Size
al	123	3
a2	4 5	2
b1	145	3
b2	2 3	2
c1	12345	5
dl	1345	4
d2	2	1
el	1 2	2
e2	3 4	2
e3	5	1

# **Shell Fragment Cubes: Ideas**

- Generalize the 1-D inverted indices to multi-dimensional ones in the data cube sense
- Compute all cuboids for data cubes ABC and DE while retaining the inverted indices
- For example, shell fragment cube ABC contains 7 cuboids:
  - A, B, C
  - AB, AC, BC
  - ABC
- This completes the offline<sup>a2 k</sup> computation stage

Cell	Intersection	TID List	List Size
al bl	123 145	1	1
a1 b2	1 2 3 \cap 2 3	2 3	2
a2 b1	45 145	4 5	2
e <sup>a2 b2</sup>	4 5 ∩ 2 3	$\otimes$	0

#### Shell Fragment Cubes: Size and Design

- Given a database of T tuples, D dimensions, and F shell fragment size, the fragment cubes' space requirement is:
  - For F < 5, the growth is sub-linear</p>

$$O\left(T\left\lceil\frac{D}{F}\right\rceil(2^F-1)\right)$$

- Shell fragments do not have to be disjoint
- Fragment groupings can be arbitrary to allow for maximum online performance
  - Known common combinations (e.g.,<city, state>) should be grouped together.
- Shell fragment sizes can be adjusted for optimal balance between offline and online computation

#### **ID\_Measure Table**

If measures other than count are present, store in ID\_measure table separate from the shell fragments

tid	count	sum
1	5	70
2	3	10
3	8	20
4	5	40
5	2	30

## **The Frag-Shells Algorithm**

- Partition set of dimension  $(A_1,...,A_n)$  into a set of *k* fragments  $(P_1,...,P_k)$ .
- 2. Scan base table once and do the following
- insert <tid, measure > into ID\_measure table.
- 4. for each attribute value a<sub>i</sub> of each dimension A<sub>i</sub>
- 5. build inverted index entry  $\langle a_i, tidlist \rangle$
- 6. For each fragment partition P<sub>i</sub>
- build local fragment cube S<sub>i</sub> by intersecting tid-lists in bottomup fashion.

#### Frag-Shells (2)



## **Online Query Computation: Query**

- A query has the general form  $\langle a_1, a_2, ..., a_n : M \rangle$
- Each a<sub>i</sub> has 3 possible values
  - Instantiated value 1
  - 2. Aggregate \* function
  - Inquire ? function 3.
- For example,  $\langle 3 ? ? * 1: count \rangle$  returns a 2-D data cube.

## **Online Query Computation: Method**

- Given the fragment cubes, process a query as follows
  - 1. Divide the query into fragment, same as the shell
  - Fetch the corresponding TID list for each fragment from the fragment cube
  - 3. Intersect the TID lists from each fragment to construct **instantiated base table**
  - 4. Compute the data cube using the base table with any cubing algorithm

#### **Online Query Computation: Sketch**



#### Experiment: Size vs. Dimensionality (50 and 100 cardinality)



• (50-C): 10<sup>6</sup> tuples, 0 skew, 50 cardinality, fragment size 3.

• (100-C): 10<sup>6</sup> tuples, 2 skew, 100 cardinality, fragment size 2.

#### **Experiments on Real World Data**

- UCI Forest CoverType data set
  - 54 dimensions, 581K tuples
  - Shell fragments of size 2 took 33 seconds and 325MB to compute
  - 3-D subquery with 1 instantiate D: 85ms~1.4 sec.
- Longitudinal Study of Vocational Rehab. Data
  - 24 dimensions, 8818 tuples
  - Shell fragments of size 3 took 0.9 seconds and 60MB to compute
  - 5-D query with 0 instantiated D: 227ms~2.6 sec.



# **Chapter 5: Data Cube Technology**

- Data Cube Computation: Preliminary Concepts
- Data Cube Computation Methods
- Processing Advanced Queries by Exploring Data Cube
   Technology
  - Sampling Cube
  - Ranking Cube
- Multidimensional Data Analysis in Cube Space
- Summary
#### **Processing Advanced Queries by Exploring Data Cube Technology**

#### Sampling Cube

- X. Li, J. Han, Z. Yin, J.-G. Lee, Y. Sun, "Sampling Cube: A Framework for Statistical OLAP over Sampling Data", SIGMOD'08
- Ranking Cube
  - D. Xin, J. Han, H. Cheng, and X. Li. Answering top-k queries with multi-dimensional selections: The ranking cube approach. VLDB'06
- Other advanced cubes for processing data and queries
  - Stream cube, spatial cube, multimedia cube, text cube, RFID cube, etc. — to be studied in volume 2

## **Statistical Surveys and OLAP**

- Statistical survey: A popular tool to collect information about a population based on a sample
  - Ex.: TV ratings, US Census, election polls
- A common tool in politics, health, market research, science, and many more
- An efficient way of collecting information (Data collection is expensive)
- Many **statistical tools** available, to determine validity
  - Confidence intervals
  - Hypothesis tests
- OLAP (multidimensional analysis) on survey data
  - highly desirable but can it be done well?

#### Surveys: Sample vs. Whole Population

Data is only a sample of **population** 

Age\Education	High-school	College	Graduate				
18							
19							
20							

### Problems for Drilling in Multidim. Space

Data is only a **sample** of population but samples could be small when drilling to certain multidimensional space

<b>Age\Education</b>	High-school	College	Graduate		
18		<b>A</b>			
19					
20	<b>A</b>	<b>^</b>	<b>A</b>		

#### OLAP on Survey (i.e., Sampling) Data

- Semantics of query is unchanged
- Input data has changed

Age/Education	High-school	College	Graduate		
18		<b>A</b>			
19		<b>\$</b>			
20			<b>A</b>		

## Challenges for OLAP on Sampling Data

- Computing confidence intervals in OLAP context
  No data?
  - Not exactly. No data in subspaces in cube
  - Sparse data
  - Causes include sampling bias and query selection bias
- Curse of dimensionality
  - Survey data can be high dimensional
  - Over 600 dimensions in real world example
  - Impossible to fully materialize

## **Example 1: Confidence Interval**

What is the average income of 19-year-old high-school students? Return not only query result but also confidence interval

Age/Education	High-school	College	Graduate		
18		<b>A</b>			
19					
20	<b>A</b>	<b>A</b>	<b>A</b>		

# **Confidence Interval**

- Confidence interval at  $ar{x}$ :  $ar{x} \pm t_c \hat{\sigma}_{ar{x}}$ 
  - x is a sample of data set;  $\bar{x}$  is the mean of sample
  - t<sub>c</sub> is the critical t-value, calculated by a look-up
  - $\hat{\sigma}_{\bar{x}} = \frac{s}{\sqrt{l}}$  is the estimated standard error of the mean
- Example: \$50,000 ± \$3,000 with 95% confidence
  - Treat points in cube cell as samples
  - Compute confidence interval as traditional sample set
- Return answer in the form of confidence interval
  - Indicates **quality** of query answer
  - User selects desired confidence interval

#### **Efficient Computing Confidence Interval Measures**

- Efficient computation in all cells in data cube
  - Both mean and confidence interval are algebraic
  - Why confidence interval measure is algebraic?

$$\bar{x} \pm t_c \hat{\sigma}_{\bar{x}}$$

 $\bar{x}$  is algebraic

 $\hat{\sigma}_{\bar{x}} = rac{s}{\sqrt{l}}$  where both *s* and *l (count)* are algebraic

 Thus one can calculate cells efficiently at more general cuboids without having to start at the base cuboid each time

## **Example 2: Query Expansion**

What is the average income of 19-year-old college students?

Age/Education	High-school	College	Graduate
18		<b>A</b>	
19		<b>Å</b> .	
20			

## **Boosting Confidence by Query Expansion**

- From the example: The queried cell "19-year-old college students" contains only 2 samples
- Confidence interval is large (i.e., low confidence). why?
  - Small sample size
  - High standard deviation with samples
- Small sample sizes can occur at relatively low dimensional selections
  - Collect more data?— expensive!
  - Use data in other cells? Maybe, but have to be careful

## Intra-Cuboid Expansion: Choice 1

Expand query to include **18** and **20** year olds?

Age/Education	High-school	College	Graduate
18		<b>A</b>	
19		<b>\$</b> , <b>\$</b> ,	
20	<b>A</b>	<b>A</b>	

## Intra-Cuboid Expansion: Choice 2

Expand query to include high-school and graduate students?

Age/Education	High-school	College	Graduate
18		<b>A</b>	
19		<b>^</b>	
20			<b>A</b>

## **Query Expansion**



## **Intra-Cuboid Expansion**

- Combine other cells' data into own to "boost" confidence
  - If share semantic and cube similarity
  - Use only if necessary
  - Bigger sample size will decrease confidence interval
  - Cell segment similarity
    - Some dimensions are clear: Age
    - Some are fuzzy: Occupation
    - May need domain knowledge
- Cell value similarity
  - How to determine if two cells' samples come from the same population?
  - Two-sample t-test (confidence-based)

# **Inter-Cuboid Expansion**

- If a query dimension is
  - Not correlated with cube value
  - But is causing small sample size by drilling down too much
- Remove dimension (i.e., generalize to \*) and move to a more general cuboid
- Can use two-sample t-test to determine similarity between two cells across cuboids
- Can also use a different method to be shown later

## **Query Expansion Experiments**

- Real world sample data: 600 dimensions and 750,000 tuples
- 0.05% to simulate "sample" (allows error checking)

(a) Intra-Cuboid Expansion with Age dimension and Average Number of Children cube measure

Q	uery	Average Q	Query Ans	wer Error	Sampling Sizes		
Gender	Marital	No Expand	Expand	% Improve	Population	Sample	Expanded
FEMALE	MARRIED	0.48	0.32	33%	2473.0	2.2	28.3
FEMALE	SINGLE	0.31	0.21	30%	612.6	0.6	6.4
FEMALE	DIVORCED	0.49	0.43	11%	321.1	0.3	3.4
MALE	MARRIED	0.42	0.21	49%	4296.8	4.4	37.6
MALE	SINGLE	0.26	0.21	16%	571.8	0.5	3.6
MALE	DIVORCED	0.33	0.27	19%	224.7	0.2	1.2
	Average	0.38	0.27	26%	1416.7	1.4	13.4

# **Chapter 5: Data Cube Technology**

- Data Cube Computation: Preliminary Concepts
- Data Cube Computation Methods
- Processing Advanced Queries by Exploring Data Cube
   Technology
  - Sampling Cube
  - Ranking Cube



- Multidimensional Data Analysis in Cube Space
- Summary

## **Ranking Cubes – Efficient Computation of Ranking queries**

- Data cube helps not only OLAP but also ranked search
- (top-k) ranking query: only returns the best k results according to a user-specified preference, consisting of (1) a *selection condition* and (2) a *ranking function*
- Ex.: Search for apartments with expected price 1000 and expected square feet 800
  - Select top 1 from Apartment
  - where City = "LA" and Num\_Bedroom = 2
  - order by [price 1000]^2 + [sq feet 800]^2 asc
- Efficiency question: Can we only search what we need?
  - Build a ranking cube on both selection dimensions and ranking dimensions

#### Ranking Cube: Partition Data on Both Selection and Ranking Dimensions

One single data partition as the template

Slice the data partition by selection conditions





for BR=2

## **Materialize Ranking-Cube**



#### Simultaneously Push Selection and Ranking



## **Processing Ranking Query: Execution Trace**

Select top 1 from Apartment where city = "LA" order by [price – 1000]<sup>2</sup> + [sq feet - 800]<sup>2</sup> asc

Bin boundary for price	[500, 600, 800, 1100,1350]
Bin boundary for sq feet	[200, 400, 600, 800, 1120]

f=[price-1000]^2 + [sq feet - 800]^2



#### **Execution Trace:**

- 1. Retrieve High-level measure for LA {11, 15}
- 2. Estimate *lower bound score* for block 11, 15

f(block 11) = 40,000, f(block 15) = 160,000

- 3. Retrieve block 11
- 4. Retrieve low-level measure for block 11
- 5. f(t6) = 130,000, f(t7) = 97,600

#### Output t7, done!

## **Ranking Cube: Methodology and Extension**

- Ranking cube methodology
  - Push selection and ranking simultaneously
  - It works for many sophisticated ranking functions
- How to support high-dimensional data?
  - Materialize only those *atomic* cuboids that contain single selection dimensions
    - Uses the idea similar to high-dimensional OLAP
    - Achieves low space overhead and high performance in answering ranking queries with a high number of selection dimensions

## **Chapter 5: Data Cube Technology**

- Data Cube Computation: Preliminary Concepts
- Data Cube Computation Methods
- Processing Advanced Queries by Exploring Data
   Cube Technology
- Multidimensional Data Analysis in Cube Space
- Summary

## Multidimensional Data Analysis in Cube Space

- Prediction Cubes: Data Mining in Multi Dimensional Cube Space
- Multi-Feature Cubes: Complex Aggregation at Multiple Granularities
- Discovery-Driven Exploration of Data Cubes

## **Data Mining in Cube Space**

- Data cube greatly increases the analysis bandwidth
- Four ways to interact OLAP-styled analysis and data mining
  - Using cube space to define data space for mining
  - Using OLAP queries to generate features and targets for mining, e.g., multi-feature cube
  - Using data-mining models as building blocks in a multistep mining process, e.g., prediction cube
  - Using data-cube computation techniques to speed up repeated model construction
    - Cube-space data mining may require building a model for each candidate data space
    - Sharing computation across model-construction for different candidates may lead to efficient mining

## **Prediction Cubes**

- Prediction cube: A cube structure that stores prediction models in multidimensional data space and supports prediction in OLAP manner
- Prediction models are used as building blocks to define the interestingness of subsets of data, i.e., to answer which subsets of data indicate better prediction

### How to Determine the Prediction Power of an Attribute?

- Ex. A customer table **D**:
  - Two dimensions Z: *Time* (*Month*, *Year*) and *Location* (*State, Country*)
  - Two features **X**: *Gender* and *Salary*
  - One class-label attribute Y: Valued Customer
- Q: "Are there times and locations in which the value of a customer depended greatly on the customers gender (i.e., Gender: predictiveness attribute V)?"
- Idea:
  - Compute the difference between the model built on that using X to predict Y and that built on using X – V to predict Y
  - If the difference is large, V must play an important role at predicting Y

## **Efficient Computation of Prediction Cubes**

- Naïve method: Fully materialize the prediction cube, i.e., exhaustively build models and evaluate them for each cell and for each granularity
- Better approach: Explore score function decomposition that reduces prediction cube computation to data cube computation

## Multidimensional Data Analysis in Cube Space

- Prediction Cubes: Data Mining in Multi-Dimensional Cube Space
- Multi-Feature Cubes: Complex Aggregation at 
   Multiple Granularities
- Discovery-Driven Exploration of Data Cubes

### **Complex Aggregation at Multiple Granularities: Multi-Feature Cubes**

- Multi-feature cubes (Ross, et al. 1998): Compute complex queries involving multiple dependent aggregates at multiple granularities
- Ex. Grouping by all subsets of {item, region, month}, find the maximum price in 2010 for each group, and the total sales among all maximum price tuples

select item, region, month, max(price), sum(R.sales)

from purchases

where year = 2010

cube by item, region, month: R

such that R.price = max(price)

 Continuing the last example, among the max price tuples, find the min and max shelf live, and find the fraction of the total sales due to tuple that have min shelf life within the set of all max price tuples

## Multidimensional Data Analysis in Cube Space

- Prediction Cubes: Data Mining in Multi-Dimensional Cube Space
- Multi-Feature Cubes: Complex Aggregation at Multiple Granularities
- Discovery-Driven Exploration of Data Cubes

### **Discovery-Driven Exploration of Data Cubes**

- Hypothesis-driven
  - exploration by user, huge search space
- Discovery-driven (Sarawagi, et al.'98)
  - Effective navigation of large OLAP data cubes
  - pre-compute measures indicating exceptions, guide user in the data analysis, at all levels of aggregation
  - Exception: significantly different from the value anticipated, based on a statistical model
  - Visual cues such as background color are used to reflect the degree of exception of each cell

## **Kinds of Exceptions and their Computation**

#### Parameters

- SelfExp: surprise of cell relative to other cells at same level of aggregation
- InExp: surprise beneath the cell
- PathExp: surprise beneath cell for each drill-down path
- Computation of exception indicator (modeling fitting and computing SelfExp, InExp, and PathExp values) can be overlapped with cube construction
- Exception themselves can be stored, indexed and retrieved like precomputed aggregates

#### **Examples: Discovery-Driven Data Cubes**

item	all
region	all

Sum of sales	mont	h										
	Jan	Feb	Mar	Арг	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
Total		1%	-1%	0%	1%	3%	-1	-9%	-1%	2%	-4%	3%

Avg sales	топ	ntla										
item	Jan	Feb	Mar	Арг	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
											ļ	
Sony b/w printer		9%	-8%	2%	-5%	14%	-4%	0%	41%	-13%	-15%	-11%
Sony color printer		0%	0%	3%	2%	4%	-10%	-13%	0%	4%	-6%	4%
HP b/w printer		-2%	1%	2%	3%	8%	0%	-12%	-9%	3%	-3%	6%
HP color printer		0%	0%	-2%	1%	0%	-1%	-7%	-2%	1%	-5%	1%
IBM home computer		1%	-2%	-1%	-1%	3%	3%	-10%	4%	1%	-4%	-1%
IBM laptop computer		0%	0%	-1%	3%	4%	2%	-1 <b>0</b> %	-2%	0%	-9%	3%
Toshiba home computer		-2%	-5%	1%	1%	-1%	1%	5%	-3%	-5%	-1%	-1%
Toshiba laptop computer		1%	0%	3%	0%	-2%	-2%	-5%	3%	2%	-1%	0%
Logitech mouse		3%	-2%	-1%	0%	4%	6%	-11%	2%	1%	-4%	0%
Ergo-way mouse		0%	0%	2%	3%	1%	-2%	-2%	-5%	0%	-5%	8%

item	IBM home computer											
Avg sales	month											
region	Jan	Feb	Mar	Арг	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
North South East West		-1% -1% -1% 4%	-3% 1% -2% <b>0</b> %	-1% -9% 2% -1%	0% 6% -3% -3%	3% -1% 1% 5%	<b>4%</b> -39% 18% 1%	-7% 9% -2% -18%	1% -34% 11% 8%	0% 4% -3% 5%	-3% 1% -2% -8%	-3% 7% -1% 1%
# **Chapter 5: Data Cube Technology**

- Data Cube Computation: Preliminary Concepts
- Data Cube Computation Methods
- Processing Advanced Queries by Exploring Data
  Cube Technology
- Multidimensional Data Analysis in Cube Space



### Data Cube Technology: Summary

- Data Cube Computation: Preliminary Concepts
- Data Cube Computation Methods
  - MultiWay Array Aggregation
  - BUC
  - Star-Cubing
  - High-Dimensional OLAP with Shell-Fragments
- Processing Advanced Queries by Exploring Data Cube Technology
  - Sampling Cubes
  - Ranking Cubes
- Multidimensional Data Analysis in Cube Space
  - Discovery-Driven Exploration of Data Cubes
  - Multi-feature Cubes
  - Prediction Cubes

### Ref.(I) Data Cube Computation Methods

- S. Agarwal, R. Agrawal, P. M. Deshpande, A. Gupta, J. F. Naughton, R. Ramakrishnan, and S. Sarawagi. On the computation of multidimensional aggregates. VLDB'96
- D. Agrawal, A. E. Abbadi, A. Singh, and T. Yurek. Efficient view maintenance in data warehouses. SIGMOD'97
- K. Beyer and R. Ramakrishnan. Bottom-Up Computation of Sparse and Iceberg CUBEs.. SIGMOD'99
- M. Fang, N. Shivakumar, H. Garcia-Molina, R. Motwani, and J. D. Ullman. Computing iceberg queries efficiently. VLDB'98
- J. Gray, S. Chaudhuri, A. Bosworth, A. Layman, D. Reichart, M. Venkatrao, F. Pellow, and H. Pirahesh. Data cube: A relational aggregation operator generalizing group-by, cross-tab and sub-totals. Data Mining and Knowledge Discovery, 1:29–54, 1997.
- J. Han, J. Pei, G. Dong, K. Wang. Efficient Computation of Iceberg Cubes With Complex Measures. SIGMOD'01
- L. V. S. Lakshmanan, J. Pei, and J. Han, Quotient Cube: How to Summarize the Semantics of a Data Cube, VLDB'02
- X. Li, J. Han, and H. Gonzalez, High-Dimensional OLAP: A Minimal Cubing Approach, VLDB'04
- Y. Zhao, P. M. Deshpande, and J. F. Naughton. An array-based algorithm for simultaneous multidimensional aggregates. SIGMOD'97
- K. Ross and D. Srivastava. Fast computation of sparse datacubes. VLDB'97
- D. Xin, J. Han, X. Li, B. W. Wah, Star-Cubing: Computing Iceberg Cubes by Top-Down and Bottom-Up Integration, VLDB'03
- D. Xin, J. Han, Z. Shao, H. Liu, C-Cubing: Efficient Computation of Closed Cubes by Aggregation-Based Checking, ICDE'06

#### Ref. (II) Advanced Applications with Data Cubes

- D. Burdick, P. Deshpande, T. S. Jayram, R. Ramakrishnan, and S. Vaithyanathan. OLAP over uncertain and imprecise data. VLDB'05
- X. Li, J. Han, Z. Yin, J.-G. Lee, Y. Sun, "Sampling Cube: A Framework for Statistical OLAP over Sampling Data", SIGMOD'08
- C. X. Lin, B. Ding, J. Han, F. Zhu, and B. Zhao. Text Cube: Computing IR measures for multidimensional text database analysis. ICDM'08
- D. Papadias, P. Kalnis, J. Zhang, and Y. Tao. Efficient OLAP operations in spatial data warehouses. SSTD'01
- N. Stefanovic, J. Han, and K. Koperski. Object-based selective materialization for efficient implementation of spatial data cubes. IEEE Trans. Knowledge and Data Engineering, 12:938–958, 2000.
- T. Wu, D. Xin, Q. Mei, and J. Han. Promotion analysis in multidimensional space. VLDB'09
- T. Wu, D. Xin, and J. Han. ARCube: Supporting ranking aggregate queries in partially materialized data cubes. SIGMOD'08
- D. Xin, J. Han, H. Cheng, and X. Li. Answering top-k queries with multi-dimensional selections: The ranking cube approach. VLDB'06
- J. S. Vitter, M. Wang, and B. R. Iyer. Data cube approximation and histograms via wavelets. CIKM'98
- D. Zhang, C. Zhai, and J. Han. Topic cube: Topic modeling for OLAP on multidimensional text databases. SDM'09

#### Ref. (III) Knowledge Discovery with Data Cubes

- R. Agrawal, A. Gupta, and S. Sarawagi. Modeling multidimensional databases. ICDE'97
- B.-C. Chen, L. Chen, Y. Lin, and R. Ramakrishnan. Prediction cubes. VLDB'05
- B.-C. Chen, R. Ramakrishnan, J.W. Shavlik, and P. Tamma. Bellwether analysis: Predicting global aggregates from local regions. VLDB'06
- Y. Chen, G. Dong, J. Han, B. W. Wah, and J. Wang, Multi-Dimensional Regression Analysis of Time-Series Data Streams, VLDB'02
- G. Dong, J. Han, J. Lam, J. Pei, K. Wang. Mining Multi-dimensional Constrained Gradients in Data Cubes. VLDB' 01
- R. Fagin, R. V. Guha, R. Kumar, J. Novak, D. Sivakumar, and A. Tomkins. Multistructural databases. PODS'05
- J. Han. Towards on-line analytical mining in large databases. SIGMOD Record, 27:97– 107, 1998
- T. Imielinski, L. Khachiyan, and A. Abdulghani. Cubegrades: Generalizing association rules. Data Mining & Knowledge Discovery, 6:219–258, 2002.
- R. Ramakrishnan and B.-C. Chen. Exploratory mining in cube space. Data Mining and Knowledge Discovery, 15:29–54, 2007.
- K. A. Ross, D. Srivastava, and D. Chatziantoniou. Complex aggregation at multiple granularities. EDBT'98
- S. Sarawagi, R. Agrawal, and N. Megiddo. Discovery-driven exploration of OLAP data cubes. EDBT'98
- G. Sathe and S. Sarawagi. Intelligent Rollups in Multidimensional OLAP Data. *VLDB'01*



# **Chapter 5: Data Cube Technology**

- Efficient Methods for Data Cube Computation
  - Preliminary Concepts and General Strategies for Cube Computation
  - Multiway Array Aggregation for Full Cube Computation
  - BUC: Computing Iceberg Cubes from the Apex Cuboid Downward
  - H-Cubing: Exploring an H-Tree Structure
  - Star-cubing: Computing Iceberg Cubes Using a Dynamic Star-tree Structure
  - Precomputing Shell Fragments for Fast High-Dimensional OLAP
- Data Cubes for Advanced Applications
  - Sampling Cubes: OLAP on Sampling Data
  - Ranking Cubes: Efficient Computation of Ranking Queries
- Knowledge Discovery with Data Cubes
  - Discovery-Driven Exploration of Data Cubes
  - Complex Aggregation at Multiple Granularity: Multi-feature Cubes
  - Prediction Cubes: Data Mining in Multi-Dimensional Cube Space
- Summary

### H-Cubing: Using H-Tree Structure

- Bottom-up computation
- Exploring an H-tree structure
- If the current computation of an H-tree cannot pass min\_sup, do not proceed further (pruning)
- No simultaneous aggregation



## **H-tree: A Prefix Hyper-tree**

	Attr. Val.		Quant-Info		Side-	link				
	Edu		Sum:2285							
	Hhd									
	Bus									
Heade	r						edu		hhd	bus
table	Jan						$\wedge$		$\backslash$	
	Feb							$\mathbf{i}$	$\backslash$	
							, I and		\ I.a. in	\ E-1-
	Tor						Jan	Mar	Jan	Feb
	Van		•••						$\backslash$	
	Mon									
_							$\sum_{\text{Tor}} \times$	→ Van	Tor	→ Mon
Month	City	Cust_grp	Prod	Cost	Price					
Jan	Tor	Edu	Printer	500	485	Г	Quant Info		OI	<b>O</b> I
Jan	Tor	Hhd	TV	800	1200	Ľ			<b>X</b>	<b>X</b>
Jan	Tor	Edu	Camera	1160	1280		Sum: 1765			
Feb	Mon	Bus	Laptop	1500	2500	(	Cnt: 2			
Mar	Van	Edu	HD	540	520		oins			
								-		

## **Computing Cells Involving "City"**

Header		Attr. Val. Q.I.		Side	e-link	l Fr	From (*, *, Tor) to (*, Jan, To						
		Eau Ubd						,,		/	,,		
								ro	ot				
Table		DUS											
										•			
<b>H</b> <sub>Tor</sub>		<u>Jan</u> Eeb						Edu	H	hd.	Bus.		
		гер						Luu.		1			
Attr. Val.	Qua	nt-Info	Side-li	nk			Jan.	Mar.		Jan.	Feb.		
Edu	Sum	:2285					1	1		1	$\setminus$		
Hhd													
Bus							<b>I</b>				$\backslash$		
							- Tor	Van	_	Tor	Mon.		
Jan													
Feb						<b>/</b>			-				
							Quant-Info	O.I.		Q.I.	Q.I.		
Tor								4 🕒					
Van							Sum: 1765						
Mon							Cnt: 2						
								-					
							bins						

#### **Computing Cells Involving Month But No City**

- 1. Roll up quant-info
- 2. Compute cells involving month but no city

Attr. Val.	Quant-Info	Side-link			
Edu.	Sum:2285				
Hhd.					
Bus.					
Jan.	•••				
Feb.					
Mar.					
Tor.					
Van.					
Mont.					



## **Computing Cells Involving Only Cust\_grp**

