Chapter 4: Data Cube Computation and Data Generalization

- Efficient Computation of Data Cubes
- Exploration and Discovery in Multidimensional Databases
- Attribute-Oriented Induction — An Alternative Data Generalization Method
Efficient Computation of Data Cubes

- Preliminary cube computation tricks (Agarwal et al.’96)
- Computing full/iceberg cubes: 3 methodologies
  - Top-Down: **Multi-Way** array aggregation (Zhao, Deshpande & Naughton, SIGMOD’97)
  - Bottom-Up:
    - **Bottom-up** computation: 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.
Preliminary Tricks (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看待s:** 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.
**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 re-used 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
Multi-way Array Aggregation for Cube Computation
Multi-Way Array Aggregation for Cube Computation (Cont.)

- 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
**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 \( \text{min}_\text{sup} \), its descendants can be pruned
  - If \( \text{min}_\text{sup} = 1 \) ⇒ 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
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., \( b_2, b_3, b_4, c_1, c_2, c_4, d_1, d_2, d_3 \)

- **Solution:** Replace such attributes by a *. Such attributes are **star attributes**, and the corresponding nodes in the cell tree are star nodes.
Example: Star Reduction

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

![Graph showing cube size vs dimensionality](image-url)
Motivation of High-D OLAP

- 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

- **Observation**: 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 high-dimensional 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

<table>
<thead>
<tr>
<th>tid</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>a1</td>
<td>b1</td>
<td>c1</td>
<td>d1</td>
<td>e1</td>
</tr>
<tr>
<td>2</td>
<td>a1</td>
<td>b2</td>
<td>c1</td>
<td>d2</td>
<td>e1</td>
</tr>
<tr>
<td>3</td>
<td>a1</td>
<td>b2</td>
<td>c1</td>
<td>d1</td>
<td>e2</td>
</tr>
<tr>
<td>4</td>
<td>a2</td>
<td>b1</td>
<td>c1</td>
<td>d1</td>
<td>e2</td>
</tr>
<tr>
<td>5</td>
<td>a2</td>
<td>b1</td>
<td>c1</td>
<td>d1</td>
<td>e3</td>
</tr>
</tbody>
</table>

- Divide the 5 dimensions into 2 shell fragments:
  - (A, B, C) and (D, E)
Chapter 4: Data Cube Computation and Data Generalization

- Efficient Computation of Data Cubes
- Exploration and Discovery in Multidimensional Databases
- Attribute-Oriented Induction — An Alternative Data Generalization Method
Chapter 4: Data Cube Computation and Data Generalization

- Efficient Computation of Data Cubes
- Exploration and Discovery in Multidimensional Databases
- Attribute-Oriented Induction — An Alternative Data Generalization Method
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

### Sum of sales

<table>
<thead>
<tr>
<th>Month</th>
<th>Jan</th>
<th>Feb</th>
<th>Mar</th>
<th>Apr</th>
<th>May</th>
<th>Jun</th>
<th>Jul</th>
<th>Aug</th>
<th>Sep</th>
<th>Oct</th>
<th>Nov</th>
<th>Dec</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total</td>
<td>1%</td>
<td>-1%</td>
<td>0%</td>
<td>1%</td>
<td>3%</td>
<td>-1</td>
<td>-9%</td>
<td>-1%</td>
<td>2%</td>
<td>-4%</td>
<td>3%</td>
<td></td>
</tr>
</tbody>
</table>

### Avg sales

<table>
<thead>
<tr>
<th>Item</th>
<th>Jan</th>
<th>Feb</th>
<th>Mar</th>
<th>Apr</th>
<th>May</th>
<th>Jun</th>
<th>Jul</th>
<th>Aug</th>
<th>Sep</th>
<th>Oct</th>
<th>Nov</th>
<th>Dec</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sony b/w printer</td>
<td>9%</td>
<td>-8%</td>
<td>2%</td>
<td>-5%</td>
<td>14%</td>
<td>-4%</td>
<td>0%</td>
<td>4%</td>
<td>-13%</td>
<td>-15%</td>
<td>-11%</td>
<td></td>
</tr>
<tr>
<td>Sony color printer</td>
<td>0%</td>
<td>0%</td>
<td>3%</td>
<td>2%</td>
<td>4%</td>
<td>-10%</td>
<td>-13%</td>
<td>0%</td>
<td>4%</td>
<td>-6%</td>
<td>4%</td>
<td></td>
</tr>
<tr>
<td>HP b/w printer</td>
<td>-2%</td>
<td>1%</td>
<td>2%</td>
<td>3%</td>
<td>8%</td>
<td>0%</td>
<td>-12%</td>
<td>-9%</td>
<td>3%</td>
<td>-3%</td>
<td>6%</td>
<td></td>
</tr>
<tr>
<td>HP color printer</td>
<td>0%</td>
<td>0%</td>
<td>-2%</td>
<td>1%</td>
<td>0%</td>
<td>-1%</td>
<td>-7%</td>
<td>-2%</td>
<td>1%</td>
<td>-5%</td>
<td>1%</td>
<td></td>
</tr>
<tr>
<td>IBM home computer</td>
<td>1%</td>
<td>-2%</td>
<td>-1%</td>
<td>-1%</td>
<td>3%</td>
<td>3%</td>
<td>-10%</td>
<td>4%</td>
<td>1%</td>
<td>-4%</td>
<td>1%</td>
<td></td>
</tr>
<tr>
<td>IBM laptop computer</td>
<td>0%</td>
<td>0%</td>
<td>-1%</td>
<td>3%</td>
<td>4%</td>
<td>2%</td>
<td>-10%</td>
<td>-2%</td>
<td>0%</td>
<td>-9%</td>
<td>3%</td>
<td></td>
</tr>
<tr>
<td>Toshiba home computer</td>
<td>-2%</td>
<td>-5%</td>
<td>1%</td>
<td>-1%</td>
<td>1%</td>
<td>-5%</td>
<td>3%</td>
<td>-5%</td>
<td>-1%</td>
<td>-1%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Toshiba laptop computer</td>
<td>1%</td>
<td>0%</td>
<td>3%</td>
<td>0%</td>
<td>-2%</td>
<td>-2%</td>
<td>-5%</td>
<td>3%</td>
<td>2%</td>
<td>-1%</td>
<td>0%</td>
<td></td>
</tr>
<tr>
<td>Logitech mouse</td>
<td>3%</td>
<td>-2%</td>
<td>-1%</td>
<td>0%</td>
<td>4%</td>
<td>6%</td>
<td>-11%</td>
<td>2%</td>
<td>1%</td>
<td>-4%</td>
<td>0%</td>
<td></td>
</tr>
<tr>
<td>Ergo-way mouse</td>
<td>0%</td>
<td>0%</td>
<td>2%</td>
<td>3%</td>
<td>1%</td>
<td>-2%</td>
<td>-2%</td>
<td>-5%</td>
<td>0%</td>
<td>-5%</td>
<td>8%</td>
<td></td>
</tr>
</tbody>
</table>

### IBM home computer

<table>
<thead>
<tr>
<th>Region</th>
<th>Jan</th>
<th>Feb</th>
<th>Mar</th>
<th>Apr</th>
<th>May</th>
<th>Jun</th>
<th>Jul</th>
<th>Aug</th>
<th>Sep</th>
<th>Oct</th>
<th>Nov</th>
<th>Dec</th>
</tr>
</thead>
<tbody>
<tr>
<td>North</td>
<td>-1%</td>
<td>-3%</td>
<td>-1%</td>
<td>0%</td>
<td>3%</td>
<td>4%</td>
<td>-7%</td>
<td>1%</td>
<td>0%</td>
<td>-3%</td>
<td>-3%</td>
<td></td>
</tr>
<tr>
<td>South</td>
<td>-1%</td>
<td>1%</td>
<td>-9%</td>
<td>6%</td>
<td>-1%</td>
<td>-39%</td>
<td>9%</td>
<td>-34%</td>
<td>4%</td>
<td>1%</td>
<td>7%</td>
<td></td>
</tr>
<tr>
<td>East</td>
<td>-1%</td>
<td>-2%</td>
<td>2%</td>
<td>-3%</td>
<td>1%</td>
<td>18%</td>
<td>-2%</td>
<td>11%</td>
<td>-3%</td>
<td>-2%</td>
<td>-1%</td>
<td></td>
</tr>
<tr>
<td>West</td>
<td>4%</td>
<td>0%</td>
<td>-1%</td>
<td>-3%</td>
<td>5%</td>
<td>1%</td>
<td>-18%</td>
<td>8%</td>
<td>5%</td>
<td>-8%</td>
<td>1%</td>
<td></td>
</tr>
</tbody>
</table>
Complex Aggregation at Multiple Granularities: Multi-Feature Cubes


- Ex. Grouping by all subsets of \{item, region, month\}, find the maximum price in 1997 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 = 1997
  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.
Cube-Gradient (Cubegrade)

- Analysis of changes of sophisticated measures in multi-dimensional spaces
  - Query: changes of average house price in Vancouver in ‘00 comparing against ’99
  - Answer: Apts in West went down 20%, houses in Metrotown went up 10%
- Cubegrade problem by Imielinski et al.
  - Changes in dimensions $\rightarrow$ changes in measures
  - Drill-down, roll-up, and mutation
From Cubegrade to Multi-dimensional Constrained Gradients in Data Cubes

- Significantly more expressive than association rules
  - Capture trends in user-specified measures

- Serious challenges
  - Many trivial cells in a cube $\rightarrow$ "significance constraint" to prune trivial cells
  - Numerate pairs of cells $\rightarrow$ "probe constraint" to select a subset of cells to examine
  - Only interesting changes wanted $\rightarrow$ "gradient constraint" to capture significant changes
Chapter 4: Data Cube Computation and Data Generalization

- Efficient Computation of Data Cubes
- Exploration and Discovery in Multidimensional Databases
- Attribute-Oriented Induction — An Alternative Data Generalization Method
What is Concept Description?

- Descriptive vs. predictive data mining
  - **Descriptive mining**: describes concepts or task-relevant data sets in concise, summarative, informative, discriminative forms
  - **Predictive mining**: Based on data and analysis, constructs models for the database, and predicts the trend and properties of unknown data

- Concept description:
  - **Characterization**: provides a concise and succinct summarization of the given collection of data
  - **Comparison**: provides descriptions comparing two or more collections of data
Summary

- Efficient algorithms for computing data cubes
  - Multiway array aggregation
  - BUC
  - H-cubing
  - Star-cubing
  - High-D OLAP by minimal cubing
- Further development of data cube technology
  - Discovery-drive cube
  - Multi-feature cubes
  - Cube-gradient analysis
- Another generalization approach: 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
- 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
- J. Han, Y. Cai and N. Cercone, Knowledge Discovery in Databases: An Attribute-Oriented Approach, VLDB'92
References (II)

- 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
- K. Ross and D. Srivastava. Fast computation of sparse datacubes. VLDB’97
- 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
- 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
- W. Wang, H. Lu, J. Feng, J. X. Yu, Condensed Cube: An Effective Approach to Reducing Data Cube Size. ICDE’02
- Y. Zhao, P. M. Deshpande, and J. F. Naughton. An array-based algorithm for simultaneous multidimensional aggregates. SIGMOD’97