Data Mining: Concepts and Techniques

— Chapter 4-4.1.3 —
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 1996]

- 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 across 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 *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!