Quotient Cube: How to Summarize the Semantics of a Data Cube

Laks V.S. Lakshmanan (Univ. of British Columbia)*
Jian Pei (State Univ. of New York at Buffalo)*
Jiawei Han (Univ. of Illinois at Urbana-Champaign)+

* The work is partially supported by NSERC and NCE/IRIS
+ The work is partially supported by NSF, UI, and Microsoft Research
Outline

• Introduction and motivation
• Cube lattice partitions
• Semantics preserving partitions
• Algorithms
• Experimental results
• Discussion and summary
### Data Cube

#### Base table

<table>
<thead>
<tr>
<th>Dimensions</th>
<th>Measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Store</td>
<td>Product</td>
</tr>
<tr>
<td>S1</td>
<td>P1</td>
</tr>
<tr>
<td>S1</td>
<td>P2</td>
</tr>
<tr>
<td>S2</td>
<td>P1</td>
</tr>
</tbody>
</table>

#### Aggregation

<table>
<thead>
<tr>
<th>Dimensions</th>
<th>Measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Store</td>
<td>Product</td>
</tr>
<tr>
<td>S1</td>
<td>P1</td>
</tr>
<tr>
<td>S1</td>
<td>P2</td>
</tr>
<tr>
<td>S2</td>
<td>P1</td>
</tr>
<tr>
<td>S1</td>
<td>*</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>*</td>
<td>*</td>
</tr>
</tbody>
</table>
Previous Work: Efficient Cube Computation

• Compute a cube from a base table: e.g. (Agarwal et al. 98), (Zhao et al. 97)
• View materialization with space constraint: e.g. Harinarayann et al. 96
• Handling scarcity (Ross & Srivastava 97)
• Cube compression: e.g. (Sismanis et al. 02), (Shanmugasundaram et al. 99), (Want et al. 02)
• Approximation: e.g. (Barbara & Sullivan 97), (Barbara & Xu 00), (Vitter et al. 98)
• Constrained cube construction: e.g. (Beyer & Ramakrishnan 99)
Previous Work: Extracting Semantics From Cubes

• General contexts of patterns (Sathe & Sarawagi 01)
• Generalize association rules (Imielinski et al. 00)
• Cube gradient analysis (Dong et al. 01)
Cube (Cell) Lattice

- Many cells have same aggregate values
- *Can we summarize the semantics of the cube by grouping cells by aggregate values?*

```plaintext
(S1,P1,s):6   (S1,P2,s):12   (S2,P1,f):9
(S1,*,s):9    (S1,P1,*):6    (S1,P2,*):12    (S2,*,f):9    (S2,P1,*):9    (S2,*,*):9
(*,P1,s):6    (*,P1,*):7.5   (*,P2,s):12    (*,P2,*):12   (*,*,f):9    (S2,*,*):9
(S1,*,*):9    (*,*,s):9     (*,P1,*):9      (*,P2,*):12   (*,*,f):9     (*,*,*):9
(*)          (*)          (*)          (*)          (*)          (*)
```
A Naïve Attempt

• Put all cells having same aggregate value in a class
Problems w/ the Naïve Attempt

• The result is not a lattice anymore!
  – Anomaly
  – The rollup/drilldown semantics is lost
A Better Partitioning

• Quotient cube: partitioning reserving the rollup/drilldown semantics
Problem Statement

• Given a cube, characterize a good way (quotient cube) of partitioning its cells into classes such that
  – The partition generates a reduced lattice preserving the rollup/drilldown semantics
  – The partition is optimal: # classes as small as possible

• Compute quotient cubes efficiently
Why A Quotient Cube Useful?

• Semantic compression
• Semantic OLAP browsing
Why A Quotient Cube Useful?

- Semantic compression
- Semantic OLAP browsing
Outline

• Introduction and motivation
• **Cube lattice partitions**
• Semantics preserving partitions
• Algorithms
• Experimental results
• Discussion and summary
Convex Partitions

• A convex partition retains semantics

\[ c_1 \xrightarrow{\text{rollup}} c_2 \xrightarrow{\text{rollup}} c_3, \quad c_1, c_3 \in CLS \Rightarrow c_2 \in CLS \]
A Non-convex Partition

- Anomaly
- The rollup/drilldown semantics is lost
Connected Partitions

- Cells c1 and c2 are connected if a series of rollup/drilldown operation starting from c1 can touch c2
- Intuitively, (each class of) a partition should be connected
Cover Partition

• For a cell c, a tuple t in base table is in c’s cover if t can be rolled up to c
  – E.g., Cov(S1,*,spring)=\{(S1,P1,spring), (S1,P2,spring)\}

<table>
<thead>
<tr>
<th>Dimensions</th>
<th>Measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Store</td>
<td>Product</td>
</tr>
<tr>
<td>S1</td>
<td>P1</td>
</tr>
<tr>
<td>S1</td>
<td>P2</td>
</tr>
<tr>
<td>S2</td>
<td>P1</td>
</tr>
</tbody>
</table>
Cover Partitions Are Convex

- All cells having the same cover are in a class
- \( (S1,P2,s) \) and \( (*,P2,*) \) cover same tuples in the base table \( \rightarrow \) \( (S1,P2,*) \) and \( (*,P2,s) \) are in the same class.
Cover Partitions Are Connected

- Cells c1 and c2 have the same cover → there must be some common ancestor c3 of c1 and c2 st c3 has the same cover
  - Cells c1 and c2 are in the same class and connected
Cover Partitions & Aggregates

• All cells in a cover partition carry the same aggregate value w.r.t. any aggregate function
  – But cells in a class of MIN() may have different covers

• For COUNT() and SUM() (positive), cover equivalence coincides with aggregate equivalence
Outline

- Introduction and motivation
- Cube lattice partitions
- Semantics preserving partitions
- Algorithms
- Experimental results
- Discussion and summary
Weak Congruence

- Weak congruence preserves semantics
Weak Congruence = Convex

- Convex $\Leftrightarrow$ no “hole” in the class $\Leftrightarrow$ weak congruence
- They preserve the rollup/drilldown semantics
- Quotient cube lattice is the lattice of convex classes
- How to derive the coarsest quotient cube?
Monotone Aggregate Functions

• Monotone functions
  – $S \subseteq T \rightarrow f(S) \geq f(T)$
  – $S \subseteq T \rightarrow f(S) \leq f(T)$
  – MIN(), MAX(), COUNT(), PSUM(), …

• The aggregate function $f$ is monotone $\rightarrow$$\equiv_f$ is the unique coarsest partition
  – MIN(): put all cells having the same MIN() value into a class
Non-monotone Functions

• Bad news: $\equiv_f$ may or may not be a convex/weak congruence. 😞
• Good news: cover partition is convex (i.e., weak congruence) and always yields a quotient cube w.r.t. any aggregate function! 😊
Outline

• Introduction and motivation
• Cube lattice partitions
• Semantics preserving partitions
• Algorithms
• Experimental results
• Discussion and summary
How to Compute A QC

- Aggregate functions
  - Monotone functions
  - Non-monotone functions

- Settings
  - The cube is available
  - Only the base table is available
Monotone Functions

• The cube is available → grab all cells with the same aggregate value and put them into a class

• Only the base table is available → bottom-up, depth-first search
  – For a cell, compute its cover, find the upper bound having the same aggregate value
  – Group lower bounds by upper bounds
Example: Cover QC
Non-monotone Functions

• Class merging
• Find cover partition classes
• Merge classes as long as convexity is retained
Example: AVG QC
Outline

• Introduction and motivation
• Cube lattice partitions
• Semantics preserving partitions
• Algorithms
• Experimental results
• Discussion and summary
Reduction Ratio vs. Dimensionality

# base tuples = 200k Zipf factor = 2.0
Reduction Ratio vs. Zipf Factor

![Graph showing reduction ratio vs. Zipf factor for MinCube, QC_Cov, and QC_MIN.](image)

- MinCube
- QC_Cov
- QC_MIN

# base tuples = 200k  # dimensions = 6
Reduction Ratio vs. Base Table Size

- MinCube
- QC_Cov
- QC_MIN

Zipf factor = 2.0 # dimensions = 6

Lakshmanan, Pei & Han. Quotient Cube: How to Summarize the Semantics of a Data Cube
Zipf factor = 2.0 # dimensions = 6
Compression Ratio on Weather Data Set

[Graph showing compression ratio vs number of dimensions for MinCube and QC_Cov.
Graph showing reduction ratio vs number of dimensions for QC_Cov and QC_Avg.]
Outline

• Introduction and motivation
• Cube lattice partitions
• Semantics preserving partitions
• Algorithms
• Experimental results
• Discussion and summary
Semantic Cube Exploration

• Theoretical foundation for semantic summarization in data cube
  – concept and properties of quotient cubes

• Efficient algorithms for quotient cube construction
  – Quotient cubes can be computed directly from base tables
Ongoing Research

• Efficient implementation of quotient cube-based OLAP system
  – Data warehouse built using quotient cubes
• Hierarchies and constraints
• Incremental maintenance
• Semantics based OLAP and mining
• Efficient query answering
References (1)

Reference (2)

- J. Han, J. Pei, G. Dong, and K. Wang. Efficient Computation of Iceberg Cubes with Complex Measures. In SIGMOD’01.
Reference (3)

• V. Harinarayan, A. Rajaraman, and J. D. Ullman. Implementing data cubes efficiently. In SIGMOD'96.
• G. Sathe and S. Sarawagi. Intelligent Rollups in Multidimensional OLAP Data. VLDB'01.
Reference (4)

- Y. Zhao, P. M. Deshpande, and J. F. Naughton. An array-based algorithm for simultaneous multidimensional aggregates. In SIGMOD'97.