

PROPERTIES AND APPLICATIONS OF DIAMOND CUBES

Hazel Webb

Computer Science and Applied Statistics, University of New Brunswick Saint John, Canada
hazel.webb@unb.ca

Keywords: OLAP, Decision Support, Data Storage.

Abstract: We introduce the diamond cube operator which, we believe, will aid decision support and analysis. In the future, we intend to show that it is also a valuable tool in areas of scientific analysis and data visualization. An algorithm has been designed, implemented and tested on a real-world, terrorism data set. The preliminary results confirm that this operation can be used to find a dense subset within a data cube and that our multi-linked list implementation generates the diamond cube. Our implementation allows the user to constrain the diamond cube by requiring that specific attribute values are included or excluded. Several mathematical properties of the diamond cube have been identified. We intend to investigate its interaction with typical On-Line Analytical Processing (OLAP) operators, slice, roll-up and drill-down; and to compare and contrast the diamond cube with other structures, e.g. the iceberg cube.

1 INTRODUCTION

OLAP is a database acceleration technique used for deductive analysis. It is used in decision support systems to provide answers to “big picture” queries such as

What were our average quarterly sales in 2006 for each region?

The main objective of OLAP is to have constant-time, or near constant-time, answers for many typical queries. To achieve such acceleration one can create a *cube* of data, a map from all attribute values to a given measure. For the purpose of this paper, we use the term *cube* to be one member of the lattice of the data cube that has a fixed set of dimensions and attribute values. Figure 1 illustrates a four-dimensional data cube lattice under roll-ups. The arrow between $ABCD$ and ABC indicates that the data is stored at a coarser granularity, i.e. the measures at level 1 do not distinguish different attribute values for dimension D .

Typical OLAP operators, slice, roll-up and drill-down [6], allow the user to extract subsets of the data cube with varying degrees of granularity. We propose a new OLAP operator, diamond dice, as a way to cluster allocated cells in a data cube. Figure 2 illustrates

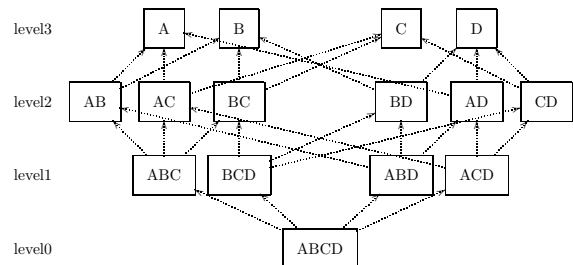


Fig. 1. 4-dimensional lattice under roll-ups

the diamond dice operation applied to the real world data set used in our experiments [8]. In this example the data were rolled up to two dimensions, *country* and *action*.

The diamond dice operation is formally defined in Section 4; methodology and experimental results are discussed in Sections 6 and 7 respectively.

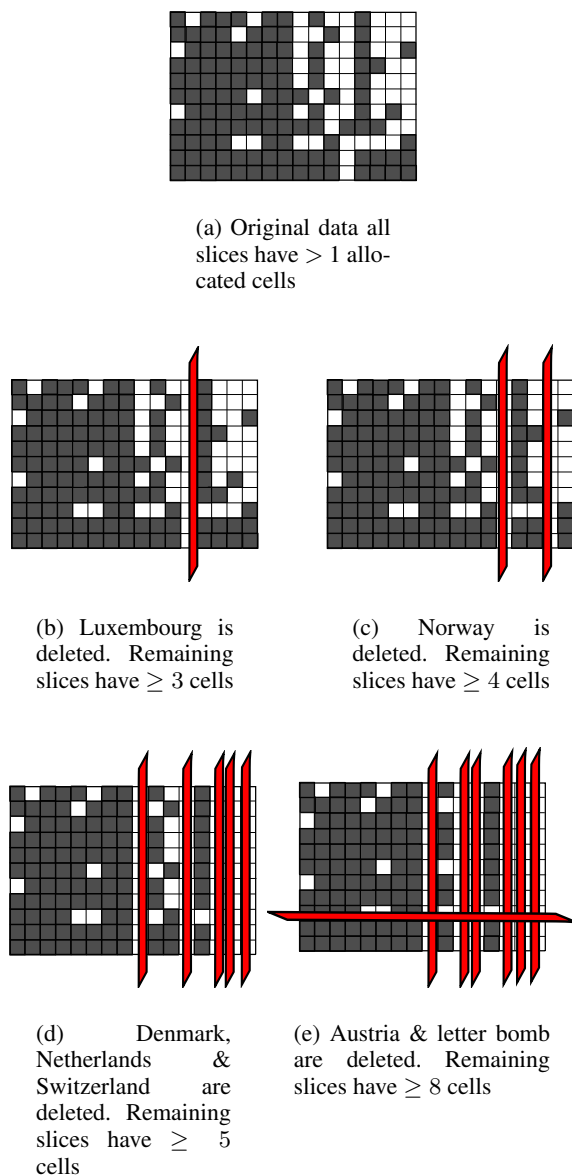


Fig. 2. Cube 5: Country \times Action

2 RELATED WORK

Data cubes are typically large and sparse, so various techniques for efficiently handling storage and access issues have been proposed. Kaser and Lemire observe that by reordering attribute values, one might discover dense areas within the cube [12]. Appropriate disparate methods could then be applied to the dense and sparse regions to reduce overall storage requirements. Iceberg cubes [9, ?,16] provide a summary of the data by extracting only those measure values that

meet or exceed some threshold. They can make it possible to store relevant parts of very large cubes in internal memory.

Other research has focussed on reducing storage requirements when computing specialized functions over the data cube. Chun et al. propose a method that finds a set of dense sub cubes in order to compute range-sum queries [5].

Investigating the relationship between attribute values in two dimensions (biclustering) is of interest to researchers in many different fields: gene expression analysis, information retrieval and text mining, collaborative filtering, etc. Madeira and Oliveira give an overview of biclustering algorithms in their survey paper [15]. Our work generalizes the concept of biclustering to higher dimensions.

We are mostly interested in user-driven tools; one must be aware of the potential pitfalls when relying solely on automated mining. Korukonda reminds us that knowledge, extracted by data mining tools alone, can be misleading [13]. He cites an article in Time Magazine [1] which stated:

4.5% Increase in sales at Red Lobster restaurants during March, in spite (or because) of the war in Iraq

It can be argued that the increase in sales and the war in Iraq are unrelated coincidental events. It is prudent to combine automatic discovery of knowledge with human expert guidance. Ben Messaoud et al. propose a method to mine Association Rules “guided by a meta-rule context, driven by analysis objectives” [3]. Their approach emphasizes the data mining of rules, coupled with human guidance, whereas ours uses automatic discovery of the largest cube of related values, together with opportunity for a user to constrain the cube to include or exclude specific attribute values.

3 PROBLEM DEFINITION

Given a large, potentially sparse, data cube can we find a large dense subcube (diamond cube)? What are the mathematical and semantic properties of such a cube? What are the interrelationships between diamond cubes generated at different levels of roll-up?

Other researchers have sought to find multiple dense regions in data cubes, e.g. to facilitate computation of range-sums [5] or mine Association Rules [14]. We believe that our approach, which includes a mathematical foundation, can provide hitherto undiscovered insights into the data set.

4 CURRENT STAGE OF RESEARCH

Although early in our investigations, we have established some properties of diamond cubes. An algorithm has been developed and implemented, and our preliminary results are very encouraging. What follows is an outline of the properties established thus far and a description of the algorithm. Our experimental results are discussed in Section 7.

4.1 Diamond Cubes

Definition 1. Given a relational database table, a dimension D is the set of values associated with a single attribute.

Definition 2. Given d dimensions D_1, \dots, D_d , a cube C is the set of dimensions together with a map from some tuples in $D_1 \times \dots \times D_d$ to a measure value.

Without losing generality, we shall assume that $|D_1| \leq |D_2| \leq \dots \leq |D_d|$.

Definition 3. A slice of a data cube is the set of cells corresponding to a single value in one dimension.

Figure 3 illustrates the slice corresponding to *arson* which comprises all the measure values for $\{(1987, \text{France}, \text{arson}), (1987, \text{Spain}, \text{arson}), (1987, \text{Belgium}, \text{arson}) \dots\}$. The SQL query that results in this slice would be:

```
SELECT SUM(measure), country, year
FROM incident
WHERE action = 'Arson'
GROUP BY country, year;
```

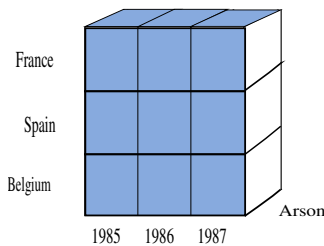


Fig. 3. slice: Arson

We begin by defining a novel measure of density for data cubes, the carat¹

¹ In actual diamonds, one carat is equal to 200 milligrams. The price of a diamond rises exponentially with the number of carats [7].

Definition 4. Given $k \geq 0$, a cube has k carats if all its slices have at least k allocated cells. Similarly, a cube has k carats over dimension i if all its slices along dimension i have at least k allocated cells.

For simplicity, we fix the required number of carats to be uniform across all dimensions. The shaded sections of Figure 4 illustrates a diamond cube of 4 carats; the unshaded area is discarded.

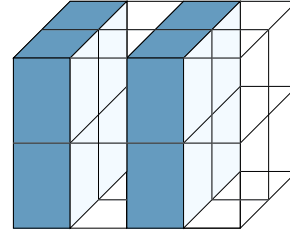


Fig. 4. 4-carat diamond cube

Definition 5. Given $k \geq 0$, a diamond cube is the maximal subcube of k carats.

We only consider this simple definition of a diamond cube in the current paper; our future work will involve more general definitions.

Proposition 1. Given that $|D_1| \leq |D_2| \leq \dots \leq |D_{d-1}| \leq |D_d|$, an upper bound for K is $\prod_{i=1}^{d-1} |D_i|$.

Proof. Consider a perfect diamond cube, i.e. all cells are allocated. We can extract a slice from this diamond by holding a value in some dimension, D_x , constant. The number of allocated cells in this slice will be the product of cardinalities of the remaining dimensions. If we choose D_x to be the dimension of largest cardinality, then the number of allocated cells in this slice is the carat (K) count for this diamond. Since all cells are allocated, there is no diamond cube with more than K carats.

Definition 6. The size of a diamond cube is the number of allocated cells.

Proposition 2. Given a d -dimensional cube, subcubes of k carats have at least size

- $k \max_{i \in \{1, 2, \dots, d\}} n_i$ where n_i is the number of attribute values in dimension D_i in the subcube;
- $k^{d/(d-1)}$.

Proof. Pick dimension D_i ; the subcube has n_i slices along this dimension, each with k allocated cells, proving the first item.

We have that $k(\sum_i n_i)/d \leq k \max_{i \in \{1,2,\dots,d\}} n_i$ so that the size of the subcube is at least $k(\sum_i n_i)/d$.

If we prove that $\sum_i n_i \geq dk^{1/(d-1)}$ then we will have that $k(\sum_i n_i)/d \geq k^{d/(d-1)}$ proving the second item. But this result can be shown using Lagrange multipliers. Consider the problem of minimizing $\sum_i n_i$ given the constraints $\prod_{i=1,2,\dots,j-1,j+1,\dots,d} n_i \geq k$ for $j = 1, 2, \dots, d$. These constraints are necessary since all slices must contain at least k cells. The corresponding Lagrangian is $L = \sum_i n_i + \sum_j \lambda_j (\prod_{i=1,2,\dots,j-1,j+1,\dots,d} n_i - k)$. By inspection, the derivatives of L with respect to n_1, n_2, \dots, n_d are zero and all constraints are satisfied when $n_1 = n_2 = \dots = n_d = k^{1/(d-1)}$. For these values, $\sum_i n_i = dk^{1/(d-1)}$ and this must be a minimum, proving the result.

Definition 7. *The union, of two cubes A and B , $A \cup B$, is the set of attributes, on each dimension, that appear in A , or B or both. The union of A and B is B if and only if A is contained in B : A is a subcube of B .*

Definition 8. *The intersection, of two cubes A and B , $A \cap B$, is the set of attributes, on each dimension, that appear in A and B . A and B are disjoint if and only if $A \cap B = \phi$.*

Definition 9. *Given a cube, A (one member of the cube lattice with a fixed number of dimensions and attributes) the complement of a diamond cube — \bar{d} — is the set of attributes that are in A but not present in the diamond cube, d .*

$$d \cap \bar{d} = \phi \text{ and } d \cup \bar{d} = A$$

Definition 10. *The volume of a cube is computed as the product of the number of attribute values in each dimension.*

Definition 11. *The density of a cube is the number of allocated cells divided by the volume of the cube.*

A diamond cube with density of 1 (all cells are allocated) is a *perfect* diamond cube. In the two-dimensional case, a perfect diamond cube is to a bi-clique what the two-dimensional cube is to an adjacency matrix.

Diamond cubes are unique, as the following proposition shows. If there is otherwise no diamond cube, we use the convention that the empty subcube is the diamond cube.

Lemma 1. *The union of two subcubes having at least k carats has at least k carats.*

Also, any slice of a subcube having k carats has k carats.

Proposition 3. *Given $K \geq 0$, there is a **unique** diamond cube of K carats.*

Proof. Observe that the union of two diamond cubes is a diamond cube, from which uniqueness follows.

Diamond cubes are not only unique, they are also nested as the following proposition shows.

Proposition 4. *The diamond cube having k carats is contained in the diamond cube having $k' < k$ carats.*

Proof. Let A be the diamond cube having k carats and B be the diamond cube having k' carats. By Lemma 1, the union of A and B has at least k' carats, and because B is maximal, the union of A and B must be B , thus A is contained in B .

Algorithm 1 computes the diamond cube in a straight-forward manner; it visits each dimension in sequence until it stabilizes. It is easy to see that the algorithm will always terminate, though it might sometimes return an empty cube. Given X , the total number of attribute values in the data cube, and n , the number of allocated cells, then Algorithm 1 runs in time $O(Xn)$.

Algorithm 1 Algorithm to compute the diamond cube of any given cube.

INPUT: a d -dimensional data cube C and $k > 0$
OUTPUT: the diamond data cube A
`stable` \leftarrow `false`
while \neg `stable` **do**
 `stable` \leftarrow `true`
 for `dim` \in $\{1, \dots, d\}$ **do**
 for `i` in all attribute values of dimension `dim` **do**
 $C_{\text{dim},i}$ \leftarrow number of allocated cells in corresponding slice
 if $C_{\text{dim},i} < k$ **then**
 delete attribute value `i`
 `stable` \leftarrow `false`
 end if
 end for
 end for
end while
return cube without deleted attribute values

Theorem 1. *Algorithm 1 is correct, that is, it always returns the diamond cube.*

Proof. Because the diamond cube is unique, we only have to show that the result of the algorithm, the cube A , is a diamond cube.

Clearly, if the result is not the empty cube, then it has at least k allocated cells per slice hence it has k

carats. We only need to show that the result of Algorithm 1 is maximal: there does not exist a larger cube of k carats.

Suppose A' is such a larger cube having k carats. Because Algorithm 1 begins with the whole cube C , there must be a time when, for the first time, one of the attribute values of C belonging to A' but not A is deleted because its corresponding slice has less than k allocated cells. Let C' be the cube at this instant, with all attribute values deleted so far. We see that C' is larger than or equal to A' and therefore, slices in C' corresponding to attribute values of A' must have more than k carats. Therefore, we have a contradiction and must conclude that A' does not exist and that A is maximal.

5 OBJECTIVES

The overall goals of this research are to show that diamond dicing is a valuable new OLAP operation and to incorporate it into a comprehensive OLAP application. A brief outline of identified subgoals follows; several of these items are examined further.

1. Diamond cubes will be computed efficiently from large data sets.
2. The semantics of diamond cubes will be investigated.
3. We will (dis)prove

Conjecture 1. The size of a diamond cube is related to the independence of the dimensions

4. Storage and indexing options will be investigated.
5. Means to handle updates to the data will be found.
6. We intend to establish a theoretical result that predicts the difference between the upper bound on K as described in Proposition 1, and its actual value.
7. The relationship between diamond cubes and other data mining algorithms and structures will be investigated, e.g. A Priori [2] and iceberg cubes [9].
8. We will establish the relationship, if it exists, between the diamond cube operation and biclustering algorithms [15] and concept lattices [10].

It is expected that this set of objectives will change and grow over time, as intermediate results are realized and produce further questions.

5.1 Semantics

Is there a general statement we can make about the attribute values in a diamond cube? As stated earlier, one can find correlation between data without a causal relationship being present. Ultimately, we believe the diamond dice operation will be useful in a decision support scenario; towards this end we intend to explain the semantics of the diamond cube. For instance, the diamond cube generated from year \times country \times action in our terrorism data indicates the years (not necessarily consecutive) and the countries in which terrorists were active, using different attack methods.

5.2 Storage

One of the inspirations for this research came from discussions about reducing storage requirements for data cubes. Our goal is to find a single large dense subcube at each level of the data cube hierarchy. It is likely that compression techniques can be applied [11] to the sparse diamond cube complement and perhaps a novel storage method will be found for the diamond cube itself.

5.3 Updates

Typically, data cubes are generated from transactional databases. Whenever data is inserted, deleted or updated in the underlying database, a data cube must be adjusted. Often this is done at off-peak times in a bulk operation [17]. Since our structure is based on relationships between values in different dimensions we will need to consider how best to handle:

- purely incremental updates
- purely decremental updates and
- a mixed sequence.

6 Methodology

Algorithm 1 was implemented in Java using multi-linked lists. An option was provided to explicitly define a value(s) that should not be pruned from the cube. This extra functionality did not impede our primary focus, finding diamond cubes, and added only a little complexity to the application logic.

The data set “Terrorism in Western Europe: Events data, TWEED” [8] contains over 11,000 records of events related to internal terrorism in 18 countries in Western Europe between 1950 and 2004. Facts (52 dimensions in total) include:

- the country in which the action took place;

- the type of action, e.g. bomb, kidnapping, ...;
- the target group, e.g. public, military, clergy, ...;
- the acting group.

The 52 dimensions of the TWEED data were rolled up to two, three and four dimensions as described in Table 1 and then used as the ‘real world’ data set for our application.

Algorithm 1 assumes that K is a known value. In practice, this will not be the case, and there are two approaches to discovering it:

1. Guess that the value of K is 1, and try to build a diamond cube. If successful, increment our guess and repeat the process until an empty cube results. We refer to this as the sequential version.
2. Apply the observations already made about a diamond cube’s properties:
 - (a) Implement a binary search over possible k values.
 - (b) Pipeline intermediate results when testing a larger k value. It should be noted, however, that it is not possible to reconstruct the diamond cube for, say $k=9$ using only the information from the diamond cube for $k=10$.
 - (c) Terminate the application when the lower bound is breached.

7 Empirical Results

The TWEED [8] data set was fed into a preprocessing step that rolled up dimensions to levels described in Table 1. Many of the 52 dimensions had obvious dependency relationships, e.g. *acting group* and *regional context of the agent*; one would expect pro-Catalonian activists to be related only with Catalonia. For this reason dimensions were rolled up to levels where we could realistically expect some degree of independence between the measures.

Table 1. Dimensions retained for experiments.

No.	Dimensions	Density
1	year \times country \times action \times target	0.019
2	year \times country \times action	0.121
3	country \times action \times target	0.240
4	year \times action \times target	0.170
5	country \times action	0.676
6	year \times action	0.617
7	year \times country	0.480

Table 2 presents some timing results and illustrates the improvement gained by the binary search

strategy, despite the cost of building a new multi-linked list if K was overestimated. Each of the times represents the average of 10 runs and indicates how long it took to find K and build, by extension, the diamond cube. All experiments were carried out on an AMD Turion64 2GHz processor with 2Gib RAM.

Table 2. Comparison between execution times, in seconds, without binary search (seq) and with (bin).

No	dimension sizes	carats	seq	bin
1	$53 \times 16 \times 11 \times 11$	38	3.00	2.74
2	$53 \times 16 \times 11$	23	2.21	1.99
3	$16 \times 11 \times 11$	24	1.56	1.36
4	$53 \times 11 \times 11$	25	2.03	1.56
5	16×11	8	1.20	0.18
6	53×11	9	1.50	0.40
7	53×16	9	1.69	0.84

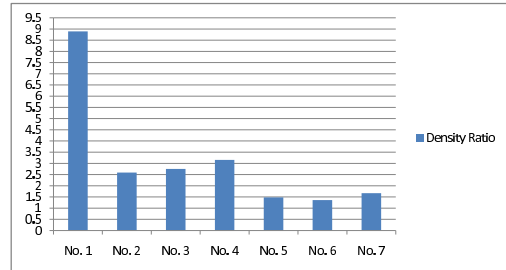


Fig. 5. Comparative Densities

The diamond dicing operation increased the density of the cubes by a factor of between 1.3 and 8.89; thus providing empirical support for our conjecture that this will be an effective method to extract dense

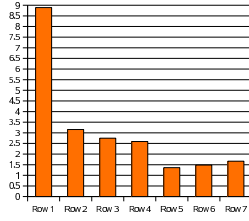


Fig. 6. Density ratios

subcubes from a data cube. Table 3 illustrates the degree to which density was increased (density ratio) by applying our diamond dicing operation. Figure 5 and Figure 6 provide the same information in graphical format. Notice that the 4-dimensional cube experienced the best increase in density, which makes us optimistic that this operator will be effective in high dimension data sets.

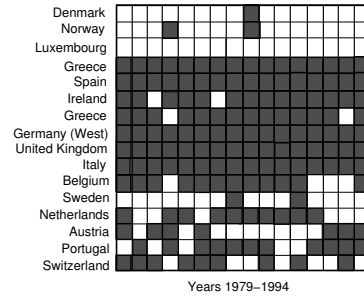
Table 3. Diamond cube size, density and ratio with original data cube.

No.	density ratio	diamond density	diamond cube dimension sizes
1	8.895	0.169	$15 \times 7 \times 5 \times 8$
2	3.157	0.382	$19 \times 9 \times 8$
3	2.750	0.660	$7 \times 7 \times 8$
4	2.588	0.440	$18 \times 7 \times 9$
5	1.360	0.920	10×10
6	1.483	0.915	13×10
7	1.667	0.800	23×12

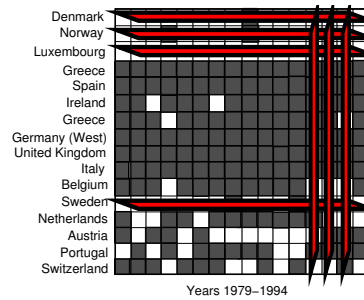
Figure 7(a) illustrates a subset of the original country \times year data; all 53 years would not fit well in the space available. The nine countries shown in 7(b) remained at the termination of the diamond dice operation. Note that although attributes Austria, Portugal and Switzerland are shown with less than nine allocated cells, they were related with enough other years not to be deleted.

7.1 Anticipated Further Work

There is much work still to do and, based on the objectives previously stated, we have identified six components to our future work.



(a) Subset of years, all countries



(b) Attribute values remaining in K-9 diamond cube

Fig. 7. Country \times Year: cube 7

Large data

We will conduct experiments on large real world data sets, where *large* is too big to fit into main memory. We acknowledge that the TWEED data set is small by data warehousing standards, but it has been useful for our initial experiments. It is anticipated that working with very large data sets will at some point require the development of an external memory algorithm to compute the diamond cube. Work in this area has been carried out by Dehne et al. who developed efficient external memory algorithms to compute the iceberg cube [4]; and Vitter et al., who developed an I/O efficient technique to approximate a data cube [19].

Complement

Our theoretical and experimental activities so far have concentrated on the diamond cube itself. We believe that there may also be interesting properties to investigate with respect to the complement, especially in the areas of compression and efficient storage.

Independence

We will build diamond cubes from data with known distributions, perhaps using DBGen [18], in order to test the conjecture that large sparse diamonds have statistically dependent dimensions .

Lattice

The diamond cube described in this paper is computed for a single level of the data cube hierarchy. We plan to compute the lattice of diamond cubes and investigate what, if any, relationships exist between them.

Updates

Handling updates to the underlying data is expected to generate its own set of questions; the decrement-only updates should be straightforward within the current implementation. How to incorporate increment-only or mixed updates is a more difficult question to answer. One potential solution is to build an auxiliary data structure to support increment-only updates.

Operator Interaction

The interaction of the diamond cube operation with other typical OLAP operators (slice, roll-up, drill-down) will need to be investigated.

8 EXPECTED OUTCOME

This research will reveal the usefulness of diamond dicing as a new OLAP operation. We intend to produce better storage and indexing strategies, which will benefit decision support systems. We will apply our operator to scientific data with the expectation that it will be useful in this area too.

REFERENCES

1. Numbers. *Time Magazine*, April 28 2003.
2. R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In J. B. Bocca, M. Jarke, and C. Zaniolo, editors, *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, pages 487–499. Morgan Kaufmann, 12–15 1994.
3. R. Ben Messaoud, S. Loudcher Rabaséda, O. Boussaid, and R. Missaoui. Enhanced mining of association rules from data cubes. In *DOLAP '06*, pages 11–18, New York, NY, USA, 2006. ACM Press.
4. Y. Chen, T. Eavis, F. Dehne, and A. Rau-Chaplin. PnP: Parallel and external memory iceberg cube computation. In *ICDE'05*, pages 576–577, 2005.
5. S.-J. Chun, C.-W. Chung, and S.-L. Lee. Space-efficient cubes for OLAP range-sum queries. *Decision Support Systems*, 37(1):83–102, 2004.
6. E. F. Codd. Providing OLAP (on-line analytical processing) to user-analysis: an IT mandate. Technical report, E. F. Codd and Associates, 1993.
7. Encyclopædia Britannica online. *diamond*. 2007. last checked December 4, 2007.
8. J. O. Engene. Five decades of terrorism in Europe: The TWEED dataset. *Journal of Peace Research*, 44(1):109–121, 2007.
9. M. Fang, N. Shivakumar, H. Garcia-Molina, R. Motwani, and J. D. Ullman. Computing iceberg queries efficiently. In *VLDB'98*, pages 299–310, 1998.
10. R. Godin, R. Missaoui, and H. Alaoui. Incremental concept formation algorithms based on Galois (concept) lattices. *Computational Intelligence*, 11:246–267, 1995.
11. O. Kaser. Compressing arrays by ordering attribute values. *Information Processing Letters*, 92(5):253–256, December 2004.
12. O. Kaser and D. Lemire. Attribute value reordering for efficient hybrid OLAP. *Information Sciences*, 176(16):2279–2438, 2006.
13. A. R. Korukonda. Technique without theory or theory from technique? an examination of practical, philosophical and foundational issues in data mining. *AI and Society The Journal of Human-Centred Systems*, August 2006.
14. W. Lian, D. W. Cheung, and S. M. Yiu. An efficient algorithm for finding dense regions for mining quantitative association rules. *Computers & Mathematics with Applications*, 50(3-4):471–490, August 2005.
15. S. C. Madeira and A. L. Oliveira. Biclustering algorithms for biological data analysis: A survey. *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, 1(1):24–45, 2004.
16. J. Pei, M. Cho, and D. Cheung. Cross table cubing: Mining iceberg cubes from data warehouses. In *SDM'05*, 2005.
17. N. Roussopoulos, Y. Kotidis, and M. Roussopoulos. Cubetree: organization of and bulk incremental updates on the data cube. pages 89–99, 1997.
18. TPC. DBGEN 2.4.0. online: <http://www.tpc.org/tpch/>, 2006. last checked December 4, 2007.
19. J. S. Vitter, M. Wang, and B. Iyer. Data cube approximation and histograms via wavelets. In *Proceedings of the 7th ACM International Conferences on Information and Knowledge Management*, pages 96–104, New York, U.S.A., 1998.