*International Journal of Applied Information Systems (IJAIS) – ISSN : 2249-0868*
*Foundation of Computer Science FCS, New York, USA*
*Volume 6– No. 6, December 2013 – www.ijais.org*

# The Near Greedy Algorithm for Views Selection in Data Warehouses and Its Performance Guarantees

Omar H. Karam
Faculty of Informatics and Computer Science, The British University in Egypt
and
Faculty of Computer and Information Sciences, Ain Shams University
Cairo, Egypt

## ABSTRACT

In data warehouses, views or summaries can be materialized to obtain better performance. In this paper, the near greedy algorithm for views selection is proposed. It is a generalization of the greedy algorithm for views selection and defines a class of solutions existing in the range between the optimal and the greedy solutions. At each of its iterations, the algorithm selects multiple views in a greedy manner instead of just one. The iterations continue until the number of desired views is reached. The algorithm's complexity is presented and the performance guarantee for the greedy algorithm is expanded to obtain a general equation that specifies the minimum performance expected from each near greedy solution.

## General Terms

Data Warehouse, views selection.

## Keywords

Greedy algorithm, near greedy algorithm, performance guarantee.

## 1. INTRODUCTION

With the increasing attention to Big Data research, there has been renewed interest in the fields of data analytics, data warehousing, cloud data storage and distributed storage. These have been awarded intense investigation through the lenses of new applications and new infrastructure technologies [1, 2, 3].

A data warehouse is "a subject-oriented, integrated, nonvolatile and time-variant collection of data in support of management's decisions" [4]. Data Warehousing research is a prime element within the cycle of knowledge discovery along with its various designs and its different costs from storage space to maintenance considerations and response times.

Major parameters for the performance of any very large database naturally include query response times. In order to improve these, summary tables and views can be pre-calculated and made available during on line operation for faster response preparation and presentation [5 - 10].

A data cube lattice is a graph showing all views of a database as vertices and the dependency relations between them as links [5, 11]. Since queries can be based on any combination of the dimensions or attributes, therefore the available views correspond to all possible combinations of dimensions and with their dependencies they produce a graph of $2^d$ vertices representing the views and with links representing the dependencies between these views. This graph or data cube lattice is known mathematically as a binary hypercube of d dimensions. It should be noted that a full lattice is only a special case. For any database or data warehouse, a range of lattices exists depending on the relationships between the different dimensions and the existing hierarchies within each dimension. Resultant final data lattices can be quite complex and may be the mathematical product of multiple sub graphs each representing a dimension or attribute. The raw data in the cube is the "original" view represented by the top vertex. Any queries can be answered directly from the relevant view.

It is possible to build or materialize any or all of the views in a lattice. The option of materializing "all" can lead to excessive unneeded storage requirements and excessive precious off line maintenance and update times. There needs to be a selection or a subset of the available views options that can be materialized without excessive storage or maintenance times and that provide acceptable query response times. These queries can directly correspond to a materialized view and therefore answered in O(1) time. Otherwise, the response will have to be calculated from the view that would provide the answer with the least possible number of calculations. A view p is dependent on another view q if p can be calculated from q. All views are dependent on the raw data view.

Figure 1 shows an example of such a lattice. It represents a simple data warehouse containing the sales values at a travel agent by the attributes passenger, airline and route: p, a and r respectively. The size of a view is shown as the number next to each view and is the number of "rows" in that particular view. The eight possible views are shown and their dependencies are present in the available links, e.g. the view (pr) has a size of 60 M rows and can directly answer queries by passenger and route. The view (p) directly answers queries by passenger. The response for such a query by passenger can be obtained from several views: (p), (pa), (pr) and (par).

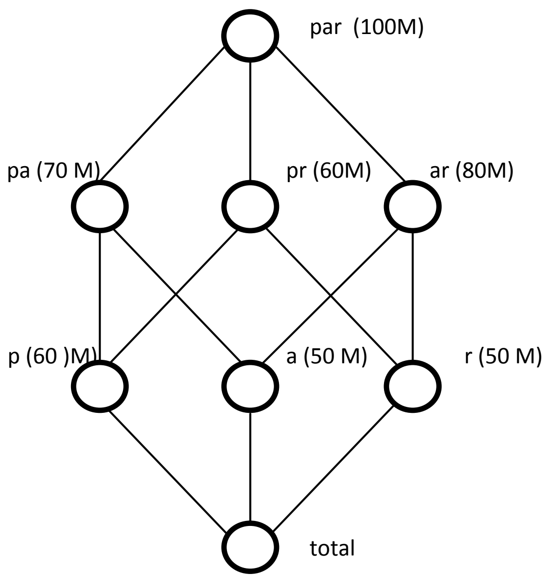

**Figure 1: A datacube lattice with three dimensions**

Data warehouse views selection algorithms therefore need to select a subset of views to materialize that fulfill constraints such as maximum storage space and best possible response times. They need to achieve these targets with as low a time complexity as possible. Finding an optimal solution means finding an optimal set of k views that succeed in realizing all the goals. For a graph of N views, this necessitates examining N.(N-1).(N-2). … . (N-k+1) possible view arrangements or sets of views, i.e. of the order $O(N^k)$.

In [5], the differences between the many researches in the literature with regards to the selection of a set of views to materialize are enumerated as:

1. The way the set of candidate views is determined;
2. The framework used to capture relationships between candidate views;
3. The use of mathematical cost models vs. calls to the system's query optimizer;
4. View selection in the relational or multidimensional context;
5. Multiple or simple query optimization;
6. Theoretical or technical solutions.

The Greedy Algorithm for views selection [5] is an algorithm that achieves a compromise between the complexity of the search for a reasonable set of views to materialize and the amount of reduction in query response times obtained from the solution. The Greedy algorithm for selecting k views to materialize is based on a greedy approach that selects the view achieving the highest "benefit" at each of k iterations. In [5] the Greedy algorithm is shown to guarantee a solution that provides at least 63% of the performance of an optimal solution with a complexity O(kN) . Further variations for the Greedy algorithm and other algorithms [6 - 10] either incorporated further cost details or were attempts at further reductions in calculations without serious loss of performance.

The near greedy algorithm for views selection presented in this paper is a generalization of the greedy algorithm for views selection. It defines a class of solutions existing in the range between the optimal and the greedy solutions. At each of its iterations, the algorithm selects a greedy set of views instead of just one. The iterations continue until the number of desired views is reached or the imposed constraints are reached.

Section 2 explains the Greedy algorithm along with its cost model and its main definitions and calculations. Section 3 presents the proposed Near-Greedy algorithm as a generalized case that covers all the space between the optimal solution and the greedy algorithm. Section 4 is the mathematical derivation of a performance guarantee for the Near Greedy algorithm's performance relative to that of the optimal solution. This guarantee equation encompasses also both the Greedy algorithm solution and the optimal solution. This section also presents the algorithm's complexity. Section 5 is the conclusion.

## 2. THE GREEDY ALGORITHM

A main algorithm for the views selection is the greedy algorithm [2]. In this section a review of the greedy algorithm is presented since the proposed near greedy algorithm and the remainder of the paper are based on that algorithm.
The cost model used is a linear cost model in which "the time to answer a query is taken to be equal to the space occupied by the view from which the query is answered" [2]. The "almost" linear relationship between size and running time of the query is expressed by the formula:

$$T = m * S + c$$

where T is the running time of the query on a view of size S and c is a fixed cost representing the overhead of running this query on a view of negligible size. The size of the view is therefore also a representation of the query time. It is expressed as the number of rows in the data warehouse. The benefit of a view is a measure of how much reduction in processing can be obtained from materializing a certain view. For the purposes of this work, if a view p is dependent on a view q, then this relation is expressed as $p \leftarrow q$.

The benefit of a view *v* relative to *S*, denoted by *B(v; S)*, is obtained as follows:

1. For each view $w \leftarrow v$, i.e. w is a descendant of *v* in the dependency graph, the benefit quantity $B_w$ can be specified by:
   a) Let *u* be the view of least cost in *S* such that $w \leftarrow u$. Note that since the top view is in *S*, there must be at least one such view in *S*.
   b) If $C(v) < C(u)$, then $B_w = C(v) - C(u)$, otherwise, $B_w = 0$.
2. Define

$$B(v;S) = \sum\nolimits_{w \leftarrow v} B_w$$

The Greedy Algorithm is then stated as:

```
S = {top view};
for i=1 to k do begin
select that view v not in S such that B(v,S) is maximized;
S = S union {v};
end;
```

The resulting set *S* is the *greedy* selection. The total benefit, A, obtained from this selection of *k* views out of all possible N views, compared to the benefit B of the optimal solution is given by:

$$\frac{A}{B} \geq 1 - \left(\frac{k-1}{k}\right)^k \qquad (1)$$

At large values of $k$, this ratio tends to 0.63. Hence, "for no lattice whatsoever does the greedy algorithm give a benefit that is less than 63% of the optimal benefit."

## 3. THE NEAR GREEDY ALGORITHM

In this paper, the near greedy algorithm is proposed. It is a generalization of the greedy algorithm with respect to the number of views selected per iteration. Instead of choosing a single view that is greedy, t views are selected simultaneously per iteration. These are chosen to offer the maximum benefit at that particular iteration. An equation is then derived that provides a performance guarantee for the algorithm's choice compared to that of an optimal solution.

A determination of the benefit of a set of views $V = \{v_1, v_2, \ldots, v_t\}$ therefore needs to be defined. The benefit of a single view explained in Section 2 is expanded to include the definition and determination of the benefit from a set of any number of views within the dependency graph or lattice.

The benefit of the t views of $V = \{v_1, v_2, \ldots, v_t\}$ relative to $S$, which is denoted by $B(V; S)$, can be determined as follows:

1. For each view $w \leftarrow V$, i.e. $w$ is a descendant of one or more views of $V$, the benefit quantity $B_w$ can be specified by:
   a) Let u be the view of least cost in $S$ such that $w \leftarrow u$. There must be at least one such view in S since the top view is in $S$,.
   b) If $w$ is a descendant of more than one view from $V$, then the $v_i$ to consider is the one having the least cost of this subset of $V$, $V'=\{v'_1, v'_2, \ldots\}$, i.e. $C_{min}(v'_i)$
   c) If $C_{min}(v'_i) < C(u)$, then $B_w = C_{min}(v'_i) - C(u)$. Otherwise, $B_w = 0$.

2. Define

$$B(V;S) = \sum\nolimits_{w \leftarrow V} B_w$$

The Near Greedy algorithm can now be written as:

```
S = {top view};
for i=1 to (k/t) do
      begin
      Select t views for V_i from the remaining R_i views
      not in S such that B(V_i,S) is maximized;
      S = S union {V_i};
      end;
```

The resulting $S$ is the *near greedy* selection. Note that at any iteration $i$, $R_i = (N-1-(i-1)t)$.

## 4. PERFORMANCE GUARANTEES AND COMPLEXITY

$V_1, V_2, \ldots, V_{k/t}$ are the sets of views selected in order by the near-greedy algorithm. These are k/t sets, each containing "t" views. A set $V_i$ contains the views $\{v_{i1}, v_{i2}, \ldots, v_{it}\}$.

$A_i$ is the benefit of $V_i$, i.e. the benefit obtained from materializing all the elements of $V_i$ with respect to the set consisting of the top view in addition to the sets of views $V_1, V_2, \ldots, V_{i-1}$.

The views $w_1, w_2, \ldots, w_k$ constitute the optimal set of $k$ views. They constitute the set that gives the maximum total benefit. They will be grouped into $(k/t)$ sets $W_1, W_2, \ldots, W(k/t)$. Each set of these is therefore of t views and they provide the maximum total benefit that can be obtained from $k$ views and therefore also from $(k/t)$ sets of views. Each $W_i$ therefore consists of the $t$ views $\{w_{i1}, w_{i2}, \ldots, w_{it}\}$ and for any $w_{ij}$, $1 <= i <= k/t$, and $1 <= j <= t$.

The benefit $a_{ij}$ is the benefit contributed by a view $v_{ij}$ and the benefit $b_{ij}$ is the benefit contributed by a view $w_{ij}$.

Let

$$A = \sum_{i=1}^{k/t} \sum_{j=1}^{t} a_{ij}$$

$$B = \sum_{i=1}^{k/t} \sum_{j=1}^{t} b_{ij}$$

Similar to [2], we put an upper bound on the b's in terms of the a's.
Define $x(w_{cd}, v_{mn})$ to be the sum over *all views* u in the lattice of the amount of the benefit $b_{cd}$ (from a view $w_{cd}$) that is attributed to $v_{mn}$.
For clarity, this analysis is applied to the case of t = 2 first before generalizing to any possible value of t. Note then that the following inequalities hold:

i. For the case of $V_1 = \{v_{11}, v_{12}\}$, therefore
For all $p, q, r, s$; for all $((p <> r)$ OR $(q <> s))$,

$$(b_{pq} + b_{rs}) \leq (a_{11} + a_{12}) \qquad (2)$$

ii. Upon considering $V_2 = \{v_{21}, v_{22}\}$, therefore
For all $p, q, r, s$; for all $((p <> r$ OR $q <> s))$,

$$(b_{pq} + b_{rs}) - \Big(x(w_{pq}, v_{11}) + x(w_{pq}, v_{12}) + x(w_{rs}, v_{11}) + xwrs,v12 \leq a21+a22 \qquad (3)$$

iii. Upon considering $V_3 = \{v_{31}, v_{32}\}$, therefore
For all $p, q, r, s$; for all $(p <> r$ OR $q <> s)$,

$$(b_{pq} + b_{rs}) - \Big(x(w_{pq}, v_{11}) + x(w_{pq}, v_{12}) + x(w_{pq}, v_{21}) + xwpq,v22+xwrs,v11+xwrs,v12+xwrs,v21+xwrs,v22 \leq a31+a32 \qquad (4)$$

Generalizing, it can then be stated that:

$$(b_{pq} + b_{rs}) - \Big(x(w_{pq}, v_{11}) + x(w_{pq}, v_{12}) + x(w_{pq}, v_{21}) + xwpq,v22+\ldots+xwpq+vj-11+xwpq+vj-12+xwrs,v11+xwrs,v12+xwrs,v21+xwrs,v22+\ldots+xwrs+vj-11+xwrs+vj-12 \leq aj1+aj2$$

Therefore,

$$(b_{pq} + b_{rs}) - x(w_{pq}, v_{11}) - x(w_{pq}, v_{12}) - x(w_{pq}, v_{21}) - x(w_{pq}, v_{22}) - \cdots - x(w_{pq} + v_{(j-1)1}) - x(w_{pq} + v_{(j-1)2}) - x(w_{rs}, v_{11}) - x(w_{rs}, v_{12}) - x(w_{rs}, v_{21}) - x(w_{rs}, v_{22}) - \cdots - x(w_{rs} + v_{(j-1)1}) - x(w_{rs} + v_{(j-1)2}) \leq (a_{j1} + a_{j2}) \qquad (5)$$

Summing over all the possible values of *r*, *s*, *p* and *q*, the following can be obtained:

Summing equation 1,

$$\sum_{p,q,r,s} (b_{pq} + b_{rs}) \leq \sum_{p,q,r,s} (a_{11} + a_{12})$$

But the LHS is B, therefore

$$B \leq \tfrac{k}{2}\,(a_{11} + a_{12}) \qquad (6)$$

Summing equation 2, the following equation is similarly obtained

$$B \leq \tfrac{k}{2}\,(a_{21} + a_{22}) + (a_{11} + a_{12}) \qquad (7)$$

Summing equation 3, the following equation is obtained

$$B \leq \tfrac{k}{2}\,(a_{31} + a_{32}) + (a_{21} + a_{22}) + (a_{11} + a_{12}) \qquad (8)$$

A general equation for this case of *t =2* can now be written as

$$B \leq \tfrac{k}{2}\left(a_{\frac{k}{2}1} + a_{\frac{k}{2}2}\right) + \cdots + (a_{21} + a_{22}) + (a_{11} + a_{12}) \qquad (9)$$

As in [5], and for a fixed A, then for the tightest bound on B to occur, the RHSs for the family of equations (5) to (8) are equal. From any two consecutive equations in this group and given their equality, the following relation is obtained:

$$\left(\frac{k}{2}\right)\left(a_{(i+1)1} + a_{(i+1)2}\right) = \left(\frac{k}{2} - 1\right)(a_{i1} + a_{i2})$$

$$a_{i1} + a_{i2} = \frac{\left(\frac{k}{2}\right)\left(a_{(i+1)1} + a_{(i+1)2}\right)}{\frac{k}{2} - 1}$$

Since

$$A = \textstyle\sum_{i=1}^{k/2}(a_{i1} + a_{i2}),$$

Therefore

$$A = \textstyle\sum_{i=0}^{\frac{k}{2}-1}\left(\frac{\frac{k}{2}}{\frac{k}{2}-1}\right)^{i}\left(a_{(\frac{k}{2})1} + a_{(\frac{k}{2})2}\right) \qquad (10)$$

and

$$B \leq \tfrac{k}{2}\left(\frac{\frac{k}{2}}{\frac{k}{2}-1}\right)^{(\frac{k}{2}-1)}\left(a_{(\frac{k}{2})1} + a_{(\frac{k}{2})2}\right) \qquad (11)$$

Therefore

$$\frac{A}{B} \geq \frac{1}{\frac{k}{2}} \sum_{i=0}^{\frac{k}{2}-1} \left(\frac{\frac{k}{2}}{\frac{k}{2}-1}\right)^{i-(\frac{k}{2}-1)}$$

$$\geq \frac{1}{\frac{k}{2}} \sum_{i=0}^{\frac{k}{2}-1} \left(\frac{\frac{k}{2}-1}{\frac{k}{2}}\right)^{(\frac{k}{2}-1)-i}$$

$$\geq \frac{1}{\frac{k}{2}} \sum_{i=0}^{\frac{k}{2}-1} \left(\frac{\frac{k}{2}-1}{\frac{k}{2}}\right)^{(\frac{k}{2}-1)-i}$$

$$\geq \frac{1}{\frac{k}{2}}\left(1 + \left(\frac{\frac{k}{2}-1}{\frac{k}{2}}\right)^{1} + \left(\frac{\frac{k}{2}-1}{\frac{k}{2}}\right)^{2} + \left(\frac{\frac{k}{2}-1}{\frac{k}{2}}\right)^{3} + \cdots + \left(\frac{\frac{k}{2}-1}{\frac{k}{2}}\right)^{\frac{k}{2}-2} + \left(\frac{\frac{k}{2}-1}{\frac{k}{2}}\right)^{\frac{k}{2}-1}\right)$$

$$\geq \frac{1}{\frac{k}{2}}\left(\frac{1 - \left(\frac{\frac{k}{2}-1}{\frac{k}{2}}\right)^{\frac{k}{2}}}{1 - \left(\frac{\frac{k}{2}-1}{\frac{k}{2}}\right)}\right)$$

Therefore,

$$\tfrac{A}{B} \geq 1 - \left(\frac{\frac{k}{2}-1}{\frac{k}{2}}\right)^{\frac{k}{2}} \qquad (12)$$

For the general case of t, equations 10 to 12 become:

$$A = \textstyle\sum_{i=0}^{\frac{k}{t}-1}\left(\frac{\frac{k}{t}}{\frac{k}{t}-1}\right)^{i}\left(a_{(\frac{k}{t})1} + a_{(\frac{k}{t})2} + \ldots\ldots + a_{(\frac{k}{t})(\frac{k}{t})}\right) \qquad (13)$$

$$B \leq \tfrac{k}{t}\left(\frac{\frac{k}{t}}{\frac{k}{t}-1}\right)^{(\frac{k}{t}-1)}\left(a_{(\frac{k}{t})1} + a_{(\frac{k}{t})2} + \cdots \ldots\ldots + a_{(\frac{k}{t})(\frac{k}{t})}\right) \qquad (14)$$

Equation 11, defining the requested ratio *A/B*, becomes

$$\tfrac{A}{B} \geq 1 - \left(\frac{\frac{k}{t}-1}{\frac{k}{t}}\right)^{\frac{k}{t}} \qquad (15)$$

This latter equation represents the ratio of the benefit of the Near Greedy algorithm to that of an optimal solution. At *t=1*, this is the case of the greedy algorithm and equation (1) is immediately obtained by this substitution. At *t=k*, this represents an optimal solution and the ratio *A/B* from the equation is immediately equal to 1. It is to be noted that the performance or the ratio *A/B* depends solely on the ratio *(k/t)*; it is not dependent upon the total number of possible views, *N*.

The complexity, however, of the near greedy algorithm proposed in this work, is $O((k/t).N^t)$; e.g. for $t=2$, the complexity is $O((k/2).N^2)$ since the number of iterations is $((k/2).N.(N-1))$.For any $t$, the number of iterations is $((k/t).N.(N-1).(N-2). \dots . (N-t+1))$. For the optimal solution, $t=k$ and the complexity is $O(N^k)$ and for the case of the greedy algorithm $(t=1)$, it is $O(kN)$.

Table 1 shows a comparison between the near greedy algorithm for several $(k/t)$ ratios, the greedy algorithm, and the optimal solution with respect to their performance guarantee ratios, $(A/B)_{min}$, and their complexities. In this table, $k$ is taken to be 12. The possible values of $t$ are therefore 1, 2, 3, 4, 6 and 12.

**Table 1. A comparison between the greedy, the optimal and several near greedy solutions**

| t | k/t | $(A/B)_{min} = 1-\left(\frac{\frac{k}{t}-1}{\frac{k}{t}}\right)^{\frac{k}{t}}$ | **Complexity** $=O((k/t)\ N^t)$ |
|---|---|---|---|
| 1 | 12 (greedy) | 0.6480 | $O(12N)$ |
| 2 | 6 | 0.6666 | $O(6N^2)$ |
| 3 | 4 | 0.6836 | $O(4N^3)$ |
| 4 | 3 | 0.7037 | $O(3N^4)$ |
| 6 | 2 | 0.7500 | $O(2N^6)$ |
| 12 | 1(optimal) | 1.0000 | $O(N^{12})$ |

## 5. Conclusion

In this work, the near greedy algorithm for the selection of data warehouse views was presented. It is a generalization of the greedy algorithm for views selection. The near greedy algorithm selects a number of views, $t$, at each of its iterations that are the optimal selection at that specific iteration. It thus encompasses the whole range of solutions from the optimal solution at one end, (t=k), to the greedy solution, (t=1), at the other. The complexity of the algorithm is found to be $O((k/t).\ N^t)$. An estimate of the benefit or gain to the query response times resulting from materializing a set of views is used to assess the performance of the algorithm's solution. From this estimate, a performance guarantee equation is derived that specifies the minimum achievable ratio between the gain or benefit obtained from the new algorithm and the benefit of the optimal solution. This ratio has a minimum value of 0.63 (greedy) and a maximum value of 1 (optimal). Future work will address the performance and implementation of the algorithm in multiple processor or parallel environments in addition to the details of warehouse maintenance and update constraints.

## 6. REFERENCES

[1] Omar Boussaid, Jérôme Darmont, Fadila Bentayeb and Sabine Loudcher, Warehousing complex data from the web, *International Journal of Web Engineering and Technology*, vol. 4, no. 4, pp. 408-433.

[2] Laurent d'Orazio, and Sandro Bimonte, Multidimensional arrays for warehousing data on clouds. In *Data Management in Grid and Peer-to-Peer Systems*, pp. 26-37. Springer Berlin Heidelberg, 2010.

[3] Eya Ben Ahmed, Ahlem Nabli and Faïez Gargouri, A survey of user-centric data warehouses: from personalization to recommendation. *arXiv preprint arXiv:1107.1779* (2011).

[4] W. H. Inmon, Building the Data Warehouse. John Wiley & Sons, 2005.

[5] Venky Harinarayan, Anand Rajaraman, and Jeffrey D. Ullman. Implementing data cubes efficiently. In *ACM SIGMOD Record*, vol. 25, no. 2, pp. 205-216. ACM, 1996.

[6] Himanshu Gupta, Selection of views to materialize in a data warehouse. *In Database Theory—ICDT'97*, pp. 98-112. Springer Berlin Heidelberg, 1997.

[7] Himanshu Gupta and Inderpal Singh Mumick. Selection of views to materialize under a maintenance cost constraint. *In Database Theory—ICDT'99*, pp. 453-470. Springer Berlin Heidelberg, 1999.

[8] Kamel Aouiche and Jérôme Darmont, Data mining-based materialized view and index selection in data warehouses, *Journal of Intelligent Information Systems* vol. 33, no. 1. pp. 65-93.

[9] T.V. Vijay Kumar, Mohammad Haider, and Santosh Kumar, A view recommendation greedy algorithm for materialized views selection. In *Information Intelligence, Systems, Technology and Management*, pp. 61-70. Springer Berlin Heidelberg, 2011.

[10] Kamel Aouiche, Pierre-Emmanuel Jouve and Jérôme Darmont, Clustering-based materialized view selection in data warehouses. *In Advances in Databases and Information Systems*, pp. 81-95. Springer Berlin Heidelberg, 2006.

[11] Jiawei Han, Micheline Kamber, and Jian Pei. *Data mining: concepts and techniques*. Morgan Kaufmann, 2006.