Integrating Clustering Techniques and OLAP Methodologies: The ClustCube Approach*

Alfredo Cuzzocrea[1] and Paolo Serafino[2]

[1] ICAR-CNR and University of Calabria
cuzzocrea@si.deis.unical.it
[2] DIES Dept., University of Calabria
pserafino@deis.unical.it

Abstract. In this paper, we introduce ClustCube, *an innovative OLAP-based framework for clustering and mining complex database objects* extracted from *distributed database settings*. To this end, ClustCube puts together conventional *clustering techniques* and well-consolidated *OLAP methodologies* in order to achieve higher expressive power and mining effectiveness over traditional methodologies for mining tuple-oriented information.

1 Introduction

While lot of proposals on *mining traditional datasets* exist (e.g., [12,3,6,8,2,13,9,7,4]), Data Mining researchers have devoted poor attention to the problem of *effectively and efficiently mining complex objects*, for instance extracted from *distributed database settings* [12]. Contrary to this actual trend, mining complex objects is indeed relevant in practical application scenarios, as *modern database systems are more and more immersed in object-oriented scenarios rather than tuple-oriented scenarios*. Among the wide family of Data Mining techniques available in the active literature, since objects essentially aggregate *low-level fields* (which, in turn, are extracted from attribute values of the underlying distributed database) into *complex classes*, it is natural to think of *clustering* (e.g., [6]) as the most suitable technique to mine such so-derived structures. Also, the combined action of clustering techniques and well-consolidated methodologies developed in the context of *OnLine Analytical Processing* (OLAP) [3] clearly offers powerful tools to mine *clustered objects* according to a multidimensional and multi-resolution vision of the underlying *object domain* [5].

Inspired by these motivations, in this paper we propose *an innovative OLAP-based framework for clustering and mining complex database objects* extracted from *distributed database settings*, called ClustCube, which encompasses a number of research innovations beyond the capabilities of actual Data Mining methodologies over large and complex-in-nature databases (e.g., [12]). To this end, ClustCube combines the power of clustering techniques over complex database objects and the power of OLAP in supporting multidimensional analysis and knowledge fruition of (clustered) complex database objects, with mining opportunities and expressive power infeasible for traditional methodologies. So-obtained ClustCube cubes store clustered complex database objects within cube cells,

* Extended Abstract

Nicola Ferro and Letizia Tanca (Eds.): SEBD 2012, Edizioni Libreria Progetto, Padova, Italia
ISBN: 978-88-96477-23-6,

rather than conventional SQL-based aggregations like in standard *Business-Intelligence*-oriented OLAP data cubes.

Figure 1 shows the "big picture" of the research we propose, i.e. the ClustCube overview. Basically, ClustCube defines a multiple-layer reference architecture that encompasses the following well-separated layers: (*i*) *Distributed DataBase Layer* (DDBL), where the target distributed database from which complex objects are extracted is located; (*ii*) *Complex Object Definition Layer* (CODL), which supports primitives and functionalities for building and managing complex objects extracted from the DDBL layer; (*iii*) the *Object Layer* (OL), where complex objects are located, along with a suitable *object schema*; (*iv*) the ClusterCube *Definition and Management Layer* (CCDML), which supports primitives and functionalities for defining and managing ClustCube cubes; (*v*) the ClusterCube *Layer* (CCL), which stores the final ClustCube cube.

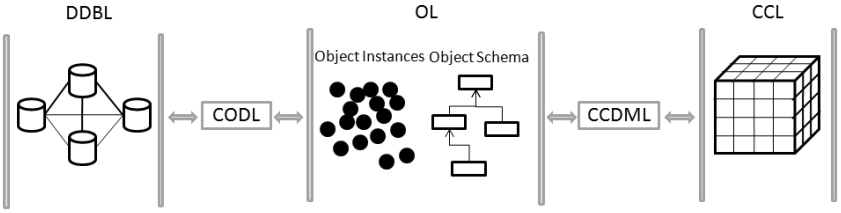

Figure 1. ClustCube Overview

While the mining capabilities exposed by the ClustCube framework are clear and evident enough, a major research challenge in the context of the ClustCube proposal is represented by the issue of efficiently computing ClustCube cubes, enriched by the respective *cuboid lattices* [3], which, similarly to conventional OLAP data cube computation axioms (e.g., [1]) can be extremely resource-consuming on wide collections of objects extracted from large distributed databases. In order to face-off this drawback, we propose algorithms that meaningfully exploit the *structured nature* of complex database objects within cuboids and the *distributive nature* of clustering across hierarchical domains, like those defined by conventional OLAP schemas.

2 Modeling ClustCube Data Cubes

In this Section we provide details on the *ClustCube data cube model*, which is directly inspired from the traditional OLAP data cube model [3], and we describe the solution for computing ClustCube cubes. As mentioned in Section 1, ClustCube cubes store clustered complex objects within cube cells. Complex objects at the OL layer (see Figure 1) are clustered by the CCDML layer on the basis of analysis/mining tasks defined by the administrator and implemented by a suitable class OS_{CODL}, following an innovative clustering approach we describe next. The result of this phase is represented by the materialized ClustCube cube C, which is finally stored in the CCL layer (see Figure 1).

To cluster complex objects and store them within the target ClustCube cube, the CCDML layer makes use of an input clustering algorithm $\mathcal{A}$, which is *orthogonal* to the ClustCube framework itself. This means that any arbitrary clustering algorithm can be exploited within the core of ClustCube, depending on particular characteristics of original data stored in the DDBL layer and the specific clustering goals. In the ClustCube

Nicola Ferro and Letizia Tanca (Eds.): SEBD 2012, Edizioni Libreria Progetto, Padova, Italia
ISBN: 978-88-96477-23-6,

framework, we name $\mathcal{A}$ as *abstract core clustering algorithm* (hereinafter referred as *core algorithm*). At a pure conceptual level, $\mathcal{A}$ captures general concepts that are applicable to every clustering algorithm. In particular, clustering algorithms typically employ a *distance function* [6] in order to determine how objects are partitioned into clusters, and the general goal consists in obtaining clusters having minimal *intra-cluster distance*, i.e. the distance between two objects belonging to the same cluster, and maximal *inter-cluster distance*, i.e. the distance between two objects belonging to different clusters. In our framework, we specialize this classical construct to the case of *distance function over objects*, which relies on the structured nature of objects that, in turn, is based on low-level fields (see Section 1). Obviously, this approach implicitly assumes that *object space is a metric space*, as we demonstrate next.

Given two objects o_i and o_j of class OS_{CODL}, we introduce the following distance function between o_i and o_j $d_{CODL}(o_i, o_j) : OS_{CODL} \times OS_{CODL} \rightarrow \mathbb{R}_0^+$, which is defined as follows:

$$d_{CODL}(o_i, o_j) = \delta_{CODL}(\langle o_i.a_0, \ldots, o_i.a_{|OS_{CODL}|-1}\rangle, \langle o_j.a_0, \ldots, o_j.a_{|OS_{CODL}|-1}\rangle) \quad (2)$$

such that: (*i*) $o_i.a_h$ denotes the h-th field of the object o_i (which, in turn, has been originally extracted from the corresponding attribute value A_h stored in a certain database of the DDBL layer – see Section 1); (*ii*) $\delta_{CODL}(o_i, o_j) : \mathcal{F}(OS_{CODL}) \times \mathcal{F}(OS_{CODL}) \rightarrow \mathbb{R}_0^+$ is a distance function over the metric space induced by the set of fields of the class OS_{CODL}, denoted by $\mathcal{F}(OS_{CODL})$. This metric space is the one taken as reference for the domain of objects of class OS_{CODL}, as mentioned above. It is important to notice here that *the set of* $|OS_{CODL}|$ *fields of class* OS_{CODL} *are used to simultaneously cluster objects of set* OI_{CODL} by means of algorithm $\mathcal{A}$, based on the introduced distance function d_{CODL}. Therefore, OS_{CODL} fields properly play the role of *clustering features* [6]. We formally denote as $\mathcal{F}(OS_{CODL})$ the set of such features. As regards practical issues, it should be noted that clustering algorithm $\mathcal{A}$ analyzes just a sub-set of L features in $\mathcal{F}(OS_{CODL})$, such that $L < |OS_{CODL}|$, in order to cluster objects of class OS_{CODL}, according to well-understood clustering principles [6].

Now, focus the attention on the structure of ClustCube cubes in a greater detail. Given a ClustCube cube C characterized by the *set of dimensions* $\mathcal{D} = \{d_0, d_1, \ldots, d_{N-1}\}$, such that $N = |OS_{CODL}|$, being dimensions in $\mathcal{D}$ corresponding to features in $\mathcal{F}(OS_{CODL})$, each ClustCube cube cell $C[i_0][i_1]\ldots[i_{N-1}] \equiv C[\mathbb{I}]$ in C, such that $0 \leq i_0 \leq |d_0|, 0 \leq i_1 \leq |d_1|, \ldots, 0 \leq i_{N-1} \leq |d_{N-1}|$, denoting $\mathbb{I} = \langle i_0, i_1, \ldots, i_{N-1}\rangle$ an *N-dimensional entry* in the *N-dimensional space* of C, $C[\mathbb{I}]$ stores a set of clustered objects, denoted by $OI_{CODL}(C[\mathbb{I}])$, that *are obtained by simultaneously clustering objects in* OI_{CODL} *with respect to the dimensions/features* $d_0, d_1, \ldots, d_{N-1}$ *in* $\mathcal{D}/\mathcal{F}(OS_{CODL})$. Hence, it is trivial to observe that, for each ClustCube cube cell $C[\mathbb{I}]$ in C *multidimensional boundaries* of $C[\mathbb{I}]$ along the dimension d_i, denoted by $B_{d_i}^{low}$ and $B_{d_i}^{up}$, respectively, with $B_{d_i}^{low} < B_{d_i}^{up}$, are determined by the $\mathcal{A}$-based clustering of objects in OI_{CODL} with respect to feature d_i in $\mathcal{D}$. In other words, while in traditional OLAP data cubes [3] the multidimensional boundaries of data cube cells along dimensions are determined by the input *OLAP aggregation scheme* (e.g., [1]), in ClustCube cubes multidimensional boundaries of ClustCube cube cells along dimensions are the *result* of the clustering process itself.

Nicola Ferro and Letizia Tanca (Eds.): SEBD 2012, Edizioni Libreria Progetto, Padova, Italia
ISBN: 978-88-96477-23-6, Copyright (c) 2012 - Edizioni Libreria Progetto and the authors

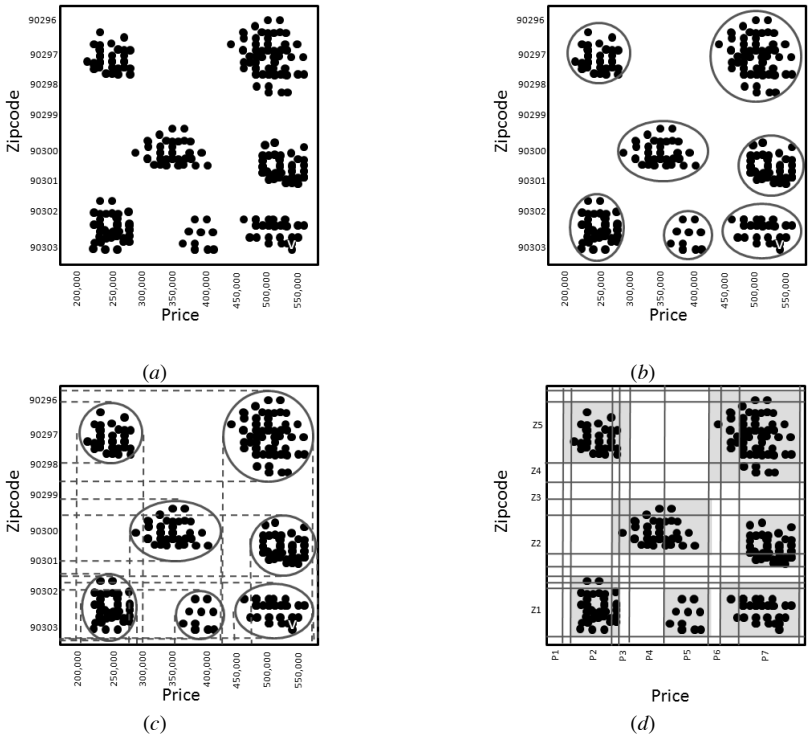

Figure 2. Traditional OLAP Scheme-Driven Aggregation (*a*) and ClustCube Clustering-Guided Aggregation (*b*) for an Example Two-Dimensional House Sale Cube

This novelty has deep consequences even in the way cube aggregations are computed. To become convinced of this, consider a simple case study focused on an house sale dataset that logically defines a two-dimensional space characterized by the following dimensions (features, respectively – see Figure 2 (*a*)): (*i*) *Price*, which represents the price at which a certain house is sold; (*ii*) *Zipcode*, which represents the zipcode of the city where the house is located. Figure 2 (*b*) shows a possible clustering of such dataset with respect to the features (dimensions, respectively) *Price* and *Zipcode*. Figure 2 (*c*) shows instead the *projection* of so-obtained clusters along both the dimensions, and, finally, Figure 2 (*d*) shows the final (logical-representation) of the two-dimensional ClustCube cube, where blue lines denote cube cells boundaries. It should be noted that, contrary to traditional OLAP data cubes where tuples are aggregated according to *regular* and *somewhat natural* groups along dimensional hierarchies defined by the input OLAP aggregation scheme, thus determining *regular partitions* of the target data domain, in novel ClustCube data cubes groups of objects stored in (ClustCube) cube cells correspond to clusters computed by input clustering algorithm $\mathcal{A}$, thus determining irregular partitions of the target object domain following sort of a *clustering-guided ClustCube aggregation scheme*. As shown in Figure 2 (*d*), given a ClustCube data cube C, every cell $C[i_0][i_1]\ldots[i_{N-1}]$ in C may alternatively contain either a whole cluster generated from $\mathcal{A}$ or a sub-cluster of it.

Nicola Ferro and Letizia Tanca (Eds.): SEBD 2012, Edizioni Libreria Progetto, Padova, Italia
ISBN: 978-88-96477-23-6,

3 Computing ClustCube Data Cubes

In this Section we provide details on how to *compute ClustCube data cubes*. One of the most relevant contribution of our research consists in equipping the final ClustCube cube generated by the CCDML layer (see Figure 1) with the canonical cubod lattice [3]. In traditional OLAP, given an N-dimensional data cube C having $\mathcal{D} = \{d_0, d_1, \ldots, d_{N-1}\}$ as dimension set, the cuboid lattice associated to C, denoted by $\mathcal{L}$, is a *hierarchical structure* composed by $2^N - 1$ cuboids, denoted by C_i, i.e. data (sub-)cubes that aggregate original relational data according to arbitrary combinations of dimensions in $\mathcal{D}$, each one at different cardinality. In other words, an n-dimensional cuboid C_i of a data cube C represents a particular n-dimensional view of C, such that $0 \leq n \leq N$. For $n = N$, the base cuboid is defined, which corresponds to the original data cube C. Cuboids of a certain cuboid lattice $\mathcal{L}$ are naturally ordered by means of the *precedence relation* $\prec$, such that, for each pair of cuboids C_i and C_j in $\mathcal{L}$, $C_i \prec C_j$ holds iff $\mathcal{D}_i \subset \mathcal{D}_j$, such that $\mathcal{D}_i$ denotes the set of dimensions of C_i and $\mathcal{D}_j$ denotes the set of dimensions of C_j, respectively. This finally determines a *cuboid hierarchy*, denoted by $\mathcal{H}(\mathcal{L})$. For instance, Figure 3 shows in the left side the cuboid lattice $\mathcal{L}$ for a four-dimensional ClustCube cube C having $\mathcal{D} = \{A, B, C, D\}$ as dimension set. Here, for instance, the property $CD \prec BCD$ holds in $\mathcal{H}(\mathcal{L})$. Also, from Figure 3 (left side), it should be clear enough that in the cuboid lattice $\mathcal{L}$ of an N-dimensional data cube C, cuboids are structurally organized according to $N + 1$ levels such that l-dimensional cuboids are located at level l of $\mathcal{L}$ are hierarchically linked to cuboids at level $l - 1$ and $l + 1$ of $\mathcal{L}$, respectively.

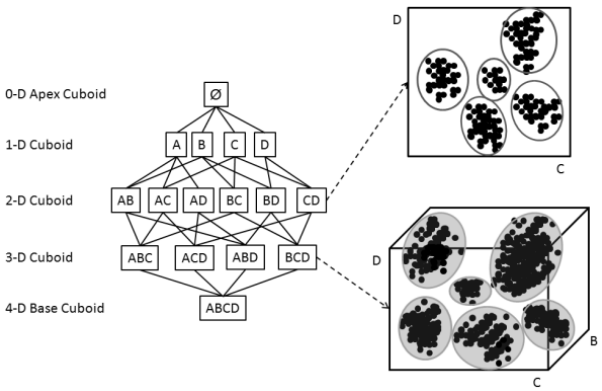

Figure 3. Cuboid Lattice of an Example Four-Dimensional ClustCube Cube, with Details on Cuboids *CD* and *BCD*

Now, focus the attention on some important properties of the ClustCube data cube model with respect to the proper clustering. Figure 3 shows in the right side clustered objects stored by the cuboids *CD* and *BCD*, respectively. Here, it should be clear enough that cuboid *BCD* stores clusters (of objects) that are *conceptually* obtained from clusters of cuboid *CD* (which *precedes BCD* in $\mathcal{H}(\mathcal{L})$, i.e. $CD \prec BCD$) *by distributing objects in CD with respect to the newly-added dimension/feature B*. In the ClustCube framework, we

Nicola Ferro and Letizia Tanca (Eds.): SEBD 2012, Edizioni Libreria Progetto, Padova, Italia
ISBN: 978-88-96477-23-6, Copyright (c) 2012 - Edizioni Libreria Progetto and the authors

fully take advantages from such a distributed nature of clustering across hierarchical cuboids, being this nice amenity the key property that allows us to significantly reduce computational needs due to compute ClustCube cube cuboid lattices. In fact, thanks to this amenity, *we do not need to compute cuboids from the scratch* but any cuboid C_j at level l of $\mathcal{L}$, such that $2 \leq l \leq N$, can be obtained from the cuboids at level $l = 1$, denoted by $\{C_0, C_1, \ldots, C_{N-1}\}$, simply by *simultaneously distributing* objects in C_i with respect to the features $\{\mathcal{D}_0, \mathcal{D}_1, \ldots, \mathcal{D}_{N-1}\}$ of $\{C_0, C_1, \ldots, C_{N-1}\}$, respectively.

Given the input class OS_{CODL}, the collection of complex objects OI_{CODL} and the input clustering algorithm $\mathcal{A}$, the CCDML layer computes the final ClustCube cube to be stored at the CCL layer (see Figure 1). To this end, ClustCube framework comprises several kinds of techniques for efficiently building the ClustCube cube C plus its cuboid lattice $\mathcal{L}$ (each one codified by a respective ClustCube cube building algorithm), which differ with respect to two *orthogonal* strategies, namely: (*i*) *materialization strategy* and (*ii*) *building strategy*. Materialization strategies specify which cuboids, among the $2^N - 1$ cuboids of $\mathcal{L}$, must be *materialized*, i.e. *computed* and *stored* (in secondary memory). On the other hand, building strategies specify how cuboids are *computed* actually. With respect to the materialization strategy, the following two alternatives are introduced in the ClustCube framework: (*i*) *full*, denoted by FUL, according to which *all* cuboids of $\mathcal{L}$ are materialized; (*ii*) *partial*, denoted by PAR, according to which a *sub-set* of the $2^N - 1$ cuboids of $\mathcal{L}$ is materialized. As regards the building strategy, ClustCube framework exposes the following two different approaches: (*i*) *baseline*, denoted by BAS, according to which, for each cuboid C_i in $\mathcal{L}$, clusters are *re-computed from the scratch* (i.e., directly from input objects in OI_{CODL}); (*ii*) *drill-down*, denoted by DRI, according to which cuboids at level l of $\mathcal{L}$ are computed from cuboids at level $l = 1$ of $\mathcal{L}$ by means of a meaningfully distributive method. It is critical to notice here that, in the ClustCube framework, the target ClustCube cube C is obtained via computing the whole cuboid lattice $\mathcal{L}$ in terms of the base cuboid [3].

4 Conclusions and Future Work

A complete OLAP-based framework for clustering and mining complex objects extracted from distributed database settings, called ClustCube, has been presented in this paper. ClustCube encompasses a spectrum of research innovations towards the seamless integration of consolidated clustering techniques over large databases and well-understood OLAP methodologies for accessing and mining (complex) objects according to a multidimensional and multi-resolution vision of the underlying object domain. Future work is mainly oriented towards extending the proposed framework in order to make it able of dealing with *classification issues over complex database objects*, beyond clustering issues like those investigated in this research.

References

1. Agarwal, S., Agrawal, R., Deshpande, P., Gupta, A., Naughton, J.F., Ramakrishnan, R., Sarawagi, S.: On the Computation of Multidimensional Aggregates. In: *Proc. of VLDB 1996*, pp. 506—521 (1996)

Nicola Ferro and Letizia Tanca (Eds.): SEBD 2012, Edizioni Libreria Progetto, Padova, Italia
ISBN: 978-88-96477-23-6,

2. Ester, M., Kriegel, H.-P., Sander, J., Xu, X.: A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In: *Proc. of KDD 1996*, pp. 226—231 (1996)
3. Gray, J., Chaudhuri, S., Bosworth, A., Layman, A., Reichart, D., Venkatrao, M., Pellow, F., Pirahesh, H.: Data Cube: A Relational Aggregation Operator Generalizing Group-by, Cross-Tab, and Sub Totals. *Data Mining and Knowledge Discovery 1(1)*, pp. 29—53 (1997)
4. Han J.: Towards On-line Analytical Mining in Large Databases. *ACM SIGMOD Record 27(1)*, pp. 97—107 (1998)
5. Han, J., Kamber, M.: *Data Mining: Concepts and Techniques, second ed.* Morgan Kauffmann Publishers, San Francisco, CA, USA (2006)
6. Hinneburg, A., Keim, D.A.: Clustering Methods for Large Databases: From the Past to the Future. In: *Proc. of ACM SIGMOD 1999*, p. 509 (1999)
7. Kriegel, H.-P., Kröger, P., Zimek, A.: Clustering High-Dimensional Data: A Survey on Subspace Clustering, Pattern-Based Clustering, and Correlation Clustering. *ACM Transactions on Knowledge Discovery from Data 3(1)*, pp. 1—58 (2009)
8. Ng, R.T., Han, J.: CLARANS: A Method for Clustering Objects for Spatial Data Mining. *IEEE Transactions on Knowledge and Data Engineering 14(5)*, pp. 1003—1016 (2002)
9. Sheikholeslami, G., Chatterjee, S., Zhang, A.: WaveCluster: A Wavelet Based Clustering Approach for Spatial Data in Very Large Databases. *VLDB Journal 8(3-4)*, pp. 289—304 (2000)
10. Transaction Processing Council, *TPC Benchmark H*, available at http://www.tpc.org/tpch/
11. Xin, D., Han, J., Xiaolei, L., Shao, Z., and Wah, B.W.: Computing Iceberg Cubes by Top-Down and Bottom-Up Integration: The StarCubing Approach. *IEEE Transactions on Knowledge and Data Engineering 19(1)*, pp. 111—126 (2007)
12. Yin, X., Han, J., Yu, P.S.: CrossClus: User-Guided Multi-Relational Clustering. *Data Mining and Knowledge Discovery 15(3)*, pp. 321—348 (2007)
13. Zhang, T., Ramakrishnan, R., Livny, M.: BIRCH: A New Data Clustering Algorithm and Its Applications. *Data Mining and Knowledge Discovery 1(2)*, pp. 141—182 (1997)

Nicola Ferro and Letizia Tanca (Eds.): SEBD 2012, Edizioni Libreria Progetto, Padova, Italia
ISBN: 978-88-96477-23-6,

Nicola Ferro and Letizia Tanca (Eds.): SEBD 2012, Edizioni Libreria Progetto, Padova, Italia
ISBN: 978-88-96477-23-6,