

Exploring OLAP Aggregates with Hierarchical Visualization Techniques

Svetlana Mansmann
University of Konstanz
Box D188, 78457 Konstanz, Germany
Svetlana.Mansmann@uni-konstanz.de

Marc H. Scholl
University of Konstanz
P.O.Box D188, 78457 Konstanz, Germany
Marc.Scholl@uni-konstanz.de

ABSTRACT

This paper presents an approach to exploring multidimensional data cubes with hierarchical visualization techniques. Analysts interact with data in a predominantly “drill-down” fashion, i.e. from coarse grained aggregates towards the desired level of detail. We suggest that visual hierarchies are adequate for mapping the multiscale nature of decomposition as they preserve the results of the entire interaction.

We introduce a class of visual structures called *Enhanced Decomposition Tree*. Every tree level is created by a disaggregation step along a chosen dimension, the nodes contain the corresponding sub-aggregates arranged into a chart and the edges are labeled with their dimensional values. Various layouts are proposed to account for different analysis tasks.

Data cubes are queried using a schema-based browser which presents dimensions by the hierarchies of their granularity levels, thus offering an efficient way of generating hierarchical visualizations. Multiple data cubes may be explored in parallel along their shared dimensions. The power of our approach is exemplified using a real-world study from the domain of academic administration.

Categories and Subject Descriptors

H.5.2 [Information Interfaces and Presentation]: User Interfaces—*Graphical user interfaces (GUI)*

Keywords

Multidimensional Data, OLAP, Hierarchical Visualization

1. INTRODUCTION

On-Line Analytical Processing (OLAP) has evolved into a core technology for comprehensive data analysis in business and, more recently, in various non-business environments. Growing complexity and volumes of the data to be analyzed impose new requirements on OLAP systems. The goal of integrating data from heterogeneous sources into a data warehouse has shifted from pure storage and report generation

towards exploratory analysis and knowledge discovery.

To efficiently analyze huge data volumes and uncover the “hidden gems” therein, OLAP tools increasingly provide functionalities for visual exploration of the data allowing the user to navigate to the desired view of aggregated data and to further explore it using various visual layouts (pivot tables, charts, plots, curves etc.) and interaction techniques. If the visual interface fits the analytical tasks and data model, as well as user’s needs and skills, the end user benefits from it by navigating through a rich, complex data set, with faster access and utilization of the data available and a higher degree of confidence in the analysis [1].

OLAP systems employ multidimensional data model to structure “raw” data into cubes in which analytical values, referred to as *measures*, are uniquely determined by descriptive values drawn from a set of *dimensions*. The values within a dimension can be further organized in a containment type hierarchy to support multiple granularities. Desired data view is retrieved by applying OLAP operations, such as *slice-and-dice* to define a sub-cube, *drill-down* and *roll-up* to perform aggregation and disaggregation, respectively, along a hierarchical dimension, *drill-across* to combine multiple cubes, *ranking* to find the outlier values, and *rotating* to see the data grouped by other dimensions.

While analytical queries aggregate over detailed data, visual analysis evolves in the inverse direction, from coarsely grained aggregates to more detailed views via stepwise decomposition along selected dimensions. Queries are specified interactively via a data browser which presents each data cube as a container of its dimensional hierarchies. Users proceed by selecting the measure(s) and the aggregation function to invoke, the dimensions to use as decomposition axes, by filtering the selected subset and manipulating the visual representation of the result. Therefore, navigation is a core component of an advanced OLAP interface.

Considering the prevailing disaggregation pattern in data exploration and the fact that aggregates obtained at earlier stages often remain useful for comparing them to their sub-aggregates or other values, hierarchical visualization techniques suggest themselves as perfect candidates for exploring OLAP data. A *decomposition tree* places each aggregate into a node and recursively obtains the constituent sub-aggregates for the child nodes by drilling down into a specified dimension. Figure 1 shows an example of decomposing the total expenditures of a university along a hierarchical dimension section \rightarrow faculty \rightarrow department. The measure value is mapped to the node’s area and the subtotals of the same parent value are arranged into space-filling bars.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SAC’07 March 11-15, 2007, Seoul, Korea
Copyright 2007 ACM 1-59593-480-4 /07/0003 ...\$5.00.

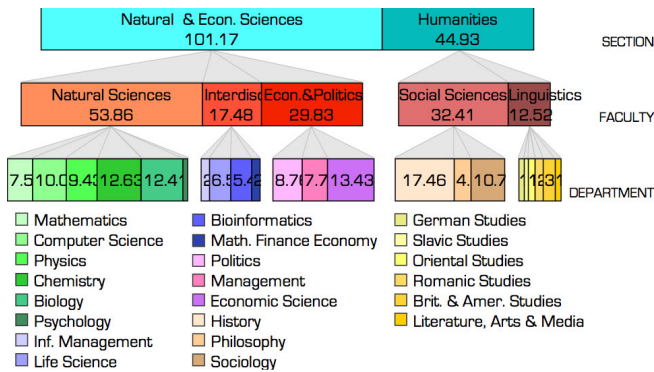


Figure 1: Decomposition tree with space-filling bars

The remainder of the paper is structured as follows: Section 2 outlines our motivation by describing related work, followed by a motivating real-world case study from a university data warehouse in Section 3. We proceed by presenting our visual framework in Section 4. Sections 5 and 6 present the design of the navigational hierarchy and the overview of supported visualization options, respectively. The summary of our contribution is given in Section 7.

2. RELATED WORK

Besides standard visual metaphors, such as pivot tables, scatterplots, and charts familiar to any data analyst, a variety of more comprehensive techniques for incremental exploration and navigation in large multidimensional data volumes have emerged. Applicability of any particular technique or their combination depends on the analysis needs and the level of user expertise. An overview of visualization techniques with respect to OLAP can be found in [7].

Eick et al. [2] introduced the problem of visual scalability and presented a survey of common visual metaphors for exploring large data sets. The presented techniques were implemented in the ADVIZOR system which uses multiple visual metaphors, each with a single zoom path. Rather than displaying multiple granularities within the same visualization, multiple *perspectives*, i.e. linked views displayed on the same screen, are used to present various projections of the same data. Another advanced system called Polaris [9] extends the Pivot Table interface by offering a combination of displays and tools for visual specification of analysis tasks. Polaris provides multiscale visualization of data cubes in form of zoom graphs for various design patterns [10].

Techapichetvanich et al. proposed a visual OLAP framework based on the Hierarchical Dimensional Visualization (HDDV) technique [11] which uses colored stacked bars. Each bar shows the results of decomposing the root aggregate along a chosen dimension, whereas the color is used to mark the portion of values satisfying the specified range condition. Bars are not explicitly linked to each other, allowing to split the same aggregate along multiple dimensions.

ProClarity introduced a hierarchical drill-down visualization called *Decomposition Tree* [6] which is the predecessor of our technique. Figure 2 shows the results of decomposing total sales by drilling down into category, sub category, region, and name. Child nodes display the next level of detail from left to right by value and percentage of the total. Unlike our example from Figure 1, ProClarity’s technique expands

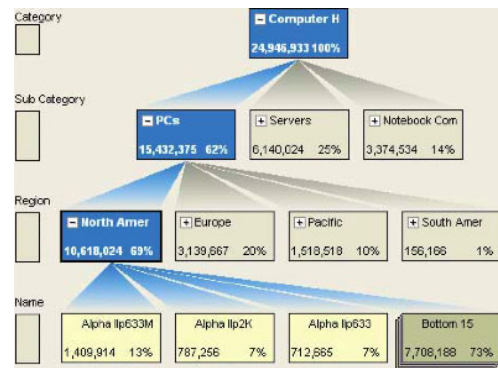


Figure 2: ProClarity’s Decomposition Tree

node-by-node, is limited to analyzing a single measure and does no visual formatting for the node values. Besides, it is rather wasteful in terms of display utilization and is thus infeasible for exploring large data volumes.

In 2005 XMLA Consulting released Report Portal 2.1, an OLAP web client that offered graphical decomposition tree techniques, such as BarChart Tree and PieChart Tree [8]. Rather than displaying plain numbers in nodes, these techniques arrange the entire drill-down view into a chart. Bar chart nodes even support multiple measures and negative numbers. Interaction approach is similar to that of ProClarity, i.e. user can expand a single aggregate value at a time. However, chart trees in Report Portal do not align drill-down steps into dimensional levels, thus allowing to expand each of the “sibling” values along a different dimension.

An overall framework for generating visual hierarchies from OLAP cubes using schema-based navigation and support for complex dimensional hierarchies is proposed in [12].

3. CASE STUDY

Throughout the remainder of the paper we will refer to the following simplified fragment of a university data warehouse consisting of two OLAP cubes:

1. **Expenditures** with facts about the amounts of money spent by the university’s institutions. An entry in the fact table corresponds to a single order.
Measure: amount (€). Dimensions: period, category, institution, project, funding.
2. **Students** with facts about the number of enrollments. An entry in the fact table corresponds to a cohort that shares the same values in all dimensional attributes.
Measures: cases and heads¹. Dimensions: period, term, origin, institution, degree, gender.

With the exception of funding and gender, the values in each dimension are further arranged into hierarchies by imposing additional granularity levels. Figure 3 shows the dimensional fact schema of both cubes. Dimensions are presented as directed graphs with aggregation levels as nodes and “rolls-up-to” relationships between them as edges. A

¹“Heads” are physical persons, assigned to the supervising department of his/her major. A person pursuing a composite degree is considered multiple “cases”, one for each major/minor, registered at the respective department: one student with 1 major and 2 minors constitutes 3 cases.

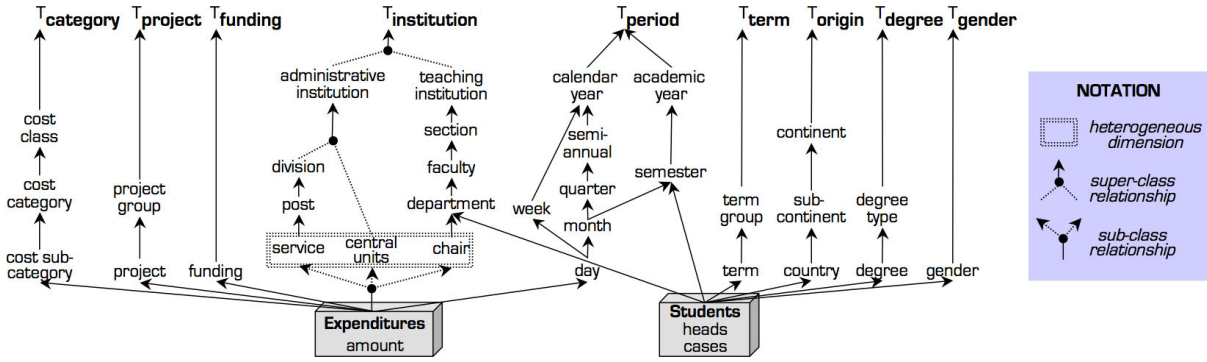


Figure 3: Dimensional fact schema of cubes **Expenditures** and **Students**

dimension’s graph is rooted at an abstract node with a single value \top , also referred to as *ALL*. The bottom-level node of each dimension is linked directly to the fact table.

Notice that *period* is a *multiple hierarchy* since it has multiple aggregation paths. The paths $\text{day} \rightarrow \text{month} \rightarrow \text{quarter} \rightarrow \text{semi-annual}$ and $\text{day} \rightarrow \text{week}$ aggregate to the same top level *calendar year* and are thus called *alternative paths*.

Dimension *institution* deserves special attention due to its heterogeneity. All institutional instances are subdivided into administrative and teaching institutions, each with its own hierarchy. *Administrative institution* is again decomposed into two sub-classes, resulting in three disjoint classes at *institution*’s bottom level. An approach to modeling and navigating in heterogeneous dimensions can be found in [5].

Note that the fact tables do not directly share any dimension, however there are common aggregation paths at upper aggregation levels, namely at *department* and *semester* in *institution* and *period*, respectively. Therefore, both cubes qualify for a multi-cube join (OLAP join) for a subsequent combined decomposition along the entire upward aggregation paths starting from the finest shared granularity.

4. THE OVERALL VISUAL FRAMEWORK

Classical presentation technique for a series of successive disaggregation steps is a pivot table which nests the multiple granularities along its two axes. However, due to the traditional textual list-based representation of the data, pivot tables are not effective for visual analysis. Standard visual metaphors, such as charts and scatterplots, do not provide an adequate alternative since they have limited dimensionality and multi-scalability potential. Decomposition tree overcomes the limitations of standard layouts by preserving all sub-aggregates and hierarchical relationships between them, similarly to pivot tables, however, using visual metaphors for presenting the subsets of the same granule in a compact and expressive way. Generation of a decomposition tree is similar to that of a pivot table: each level of detail emerges by specifying the dimension to disaggregate along².

The entire explorative framework is composed of an input and an output area for specifying queries and presenting query results, respectively. The input is an OLAP navigation which displays the structure and the contents of the data cubes in form of a visual browser. The main area of the graphical interface serves for outputting the results of user

²In terms of a SQL statement, decomposition adds the selected dimension’s attribute to **SELECT** and **GROUP BY** clause.

interaction in a selected visual format. Both above elements of the framework are described in the following sections.

4.1 A Hierarchical Cube Presentation Model

A presentation model is necessary for formalizing the concept and the elements for mapping data cubes to visual hierarchies. Our model borrows the components of the *Cube Presentation Model (CPM)* of Maniatis et al. [4], adjusting the latter to the specifics of hierarchical disaggregation.

- (a) dimensions \mathcal{D}_i defined as lattices of granularity levels, or categories \mathcal{C}_j , denoted $\mathcal{C}_j \in \mathcal{D}_i$,
- (b) extension, or data set, \mathcal{C}_j of the dimensional level \mathcal{C}_j ,
- (c) dimensional values e_k populating \mathcal{C}_j , denoted $e_k \in \mathcal{C}_j$,
- (d) predicates \sqsubset and \sqsubset^* for child and descendant relationship, respectively, between the dimensional levels,
- (e) predicates \sqsubseteq and \sqsubseteq^* for child and transitive closure relationship, respectively, between the values of related levels of a dimension,
- (f) fact tables as detailed data sets at the finest granule of information for all their dimensions,
- (g) cubes, defined as aggregations over fact tables.

Consider an example of applying the above formalism to the dimension *degree* from our cases study:

$$\begin{aligned} \mathcal{D}_1 &= \top \text{degree}, \mathcal{C}_1 = \text{degree}, \mathcal{C}_2 = \text{degree type}, \\ \mathcal{C}_1 &\in \mathcal{D}, \mathcal{C}_2 \in \mathcal{D}, \mathcal{C}_1 \sqsubset \mathcal{C}_2, \\ e_1 &= \text{M.Sc., Information Engineering}, e_1 \in \mathcal{C}_1, \\ e_{12} &= \text{Master of Science}, e_{12} \in \mathcal{C}_2, e_1 \sqsubseteq e_{12}. \end{aligned}$$

We proceed by introducing the presentation layer which defines the elements of the visual layout.

Dimensional axis corresponds to a dimension’s category. Every distinct value of its extension is interpreted as a **coordinate** (e.g. axis *month*, coordinate *March*). Each node of a decomposition tree as well as each level of the tree has its own dimensional axis. **Decomposition axis** is the dimensional axis chosen for the disaggregation step.

Decomposition schema is the lattice of all decomposition axes of the visual hierarchy. The schema of the tree in Figure 1 is composed of *section*, *faculty*, and *department*.

Multicube. A query accessing multiple data cubes implies joining the latter along all their common dimension(s). To qualify for a multi-cube join, the fact tables need to be pre-aggregated to the granularity of those common dimensions. Back to our case study, *Expenditures* is transformed by retrieving the slice belonging to *teaching institution*. Then, both data cubes are “rolled up” to *department* and *semester* and linked to each other via a natural join. The result of the

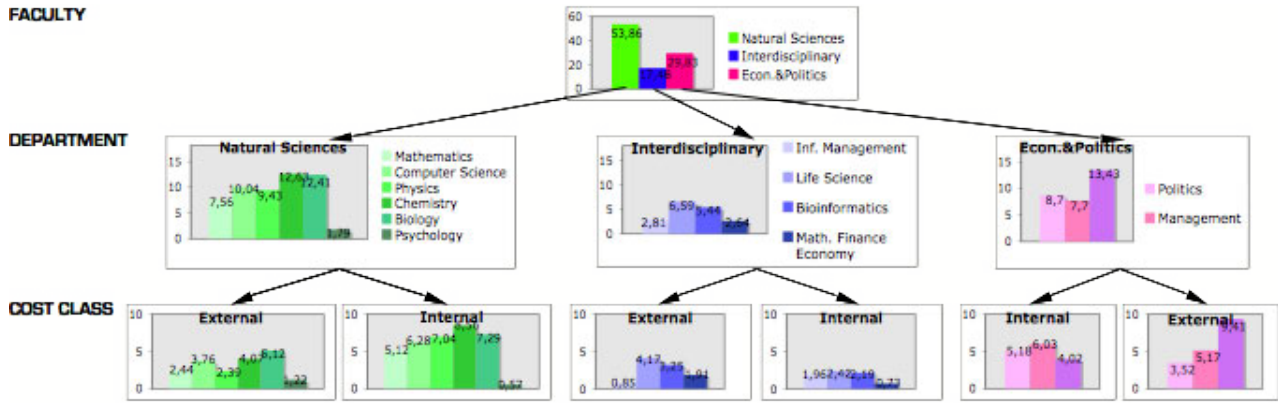


Figure 4: A decomposition tree produced by an inner (faculty → department) and an outer (department → cost class) split

join, called **multicube**, contains only the shared dimensions and all the measures of the joined cubes.

Tree node contains the results of decomposing a value along a dimensional axis. Sub-aggregates in the node are arranged into a chart or another 2-dimensional view.

Tree split operation generates a new bottom level of the decomposition tree by computing the sub-aggregates for a selected decomposition axis. We distinguish between a primary, an inner, and an outer split.

Primary split is the first decomposition step which generates the subset of aggregates to populate the root node. We introduce a function $primary(C_j)$ which returns TRUE if the dimensional level C_j is the axis of the primary split.

Inner split is a drill-down into a sub-dimension C_k of the primary split, i.e. $\exists C_j : C_k \sqsubset^* C_j \wedge primary(C_j) = \text{TRUE}$. The inner split refines the granularity within the nested chart.

Outer split is a decomposition step along any axis C_i which is not a part of the primary split axis's dimensional hierarchy, i.e. $\nexists C_m : C_i \sqsubset^* C_m \wedge primary(C_m) = \text{TRUE}$.

Figure 4 illustrates the above split types. Primary split along **faculty** generated the values in the root node; decomposition along **department** (department \sqsubset faculty) is an inner split leading to a refined granularity of the charts. Outer split along **cost class** leaves the chart granularity unchanged; each value in a child node's chart is a sub-aggregate of the corresponding value in the parent node's chart.

5. SCHEMA-BASED DATA NAVIGATION

OLAP tools allow users to navigate in data cubes by arranging the data into a browser-like hierarchy. Fact tables are represented by top-level folders (cube icons) containing subfolders of their dimensions and the selection of available measures and applicable aggregation functions.

Existing approaches to navigating in dimensional hierarchies can be reduced to the following ones:

1. *Extension-based* navigation represents the dimension by its extension, i.e. data hierarchy. Each hierarchical entity is a folder that can be recursively expanded to display its descendants, as shown in Figure 5 (left).
2. *Intension-based* approach interfaces the dimension via its intension, i.e. schema hierarchy. Each category of the dimension is a subfolder of its parent category; the extension of any category can be retrieved on demand, as illustrated in Figure 5 (right).

While standard OLAP interfaces favor the simple and straightforward extension-based approach, more sophisticated tools opt for the intension-based navigation in order to provide advanced functionality, such support for multiple and heterogeneous hierarchies. In our scenario, it is indispensable to explicitly display the schema since the very generation of a visual hierarchy is basically about specifying the granularities for the tree levels. Filtering (“slice&dice”) of the data is done by accessing the extension of any given category node or by directly deleting/minimizing nodes and their subtrees in the visualization.

A view of the resulting navigation hierarchy for cube **Expenditures** is presented in Figure 6 (left). The information about the structure of the data cubes and their components, necessary for generating the data browser, is stored in the database as metadata. The behavior of any dimensional node depends on its position in the intension hierarchy and its type (e.g. multiple hierarchy, non-hierarchical, etc.). A node is *abstract* if its extension is empty, i.e. contains no data. Abstract nodes are generally used as a common superclass category whenever multiple child categories occur.

Based on our case study, we distinguish between the following types and roles of dimensional navigation nodes:

Node types:

- *Non-hierarchical* node is a bottom-level category with the values of the finest granularity (e.g., **funding**), represented by a non-expandable page icon.
- *Single hierarchy* is a non-abstract node with a single child category (e.g., **academic year**).

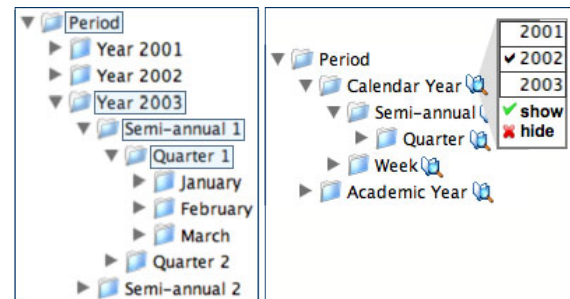


Figure 5: Navigating in hierarchical dimensions: extension-based (left) vs. intension-based (right) approach

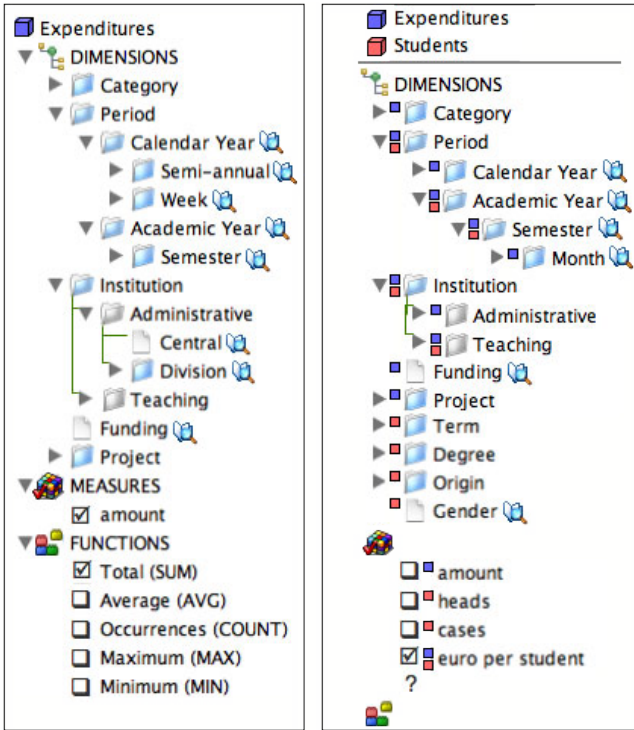


Figure 6: In-cube (*left*) vs. multicube (*right*) navigation

- *Multiple hierarchy* is an abstract node with multiple child categories for the respective aggregation paths (e.g., *calendar year*). Each child node’s extension spans the entire bottom-level data of the dimension.
- *Super-class* is an abstract parent (gray folder icon) of a single or multiple categories. A child node may span the entire bottom-level data (e.g., *teaching institution*) or its subset (e.g., *administrative institution*).

Node roles:

- *Heterogeneous* node is a super-class of multiple sub-class categories. (e.g., *institution*).
- *Sub-class* node is a child in a heterogeneous hierarchy (e.g., *central units* and *division*). Sub-classes of a category are additionally linked to each other to distinguish them from paths in a multiple hierarchy.
- *Root* is a top-level abstract node used purely as a “wrapper” for a single hierarchy (e.g., *project*).

Each node is of exactly one type and may have none, a single, or even a double role. Double role occurs when a heterogeneous node is itself a sub-class of another category. Properties of the above node types and roles are summarized in Tables 1 and 2, respectively, and can be tested using a compatibility graph shown in Figure 7. The type and the role(s) of each node define its query behavior. The navigational framework retrieves this additional metadata to load the corresponding query building routine.

³**span** refers to the coverage of the node’s child subtrees w.r.t. the extension of the dimension’s bottom level.

Table 1: Overview of dimensional node types

type	abstract	children	span ³
non-hierarchical	no	0	100%
single hierarchy	no	1	100%
multiple hierarchy	yes	>1	100%
super-class	yes	≥1	≤100%

Table 2: Overview of dimensional node roles

role	abstract	children	siblings
sub-class	<i>no</i>	≥0	>0
heterogeneous	<i>yes</i>	>1	≥0
root	<i>yes</i>	≥1	N/A

5.1 Multi-cube Join Navigation

In our case study, an analyst may wish to use a multi-cube to compare the evolution of the university’s expenditures versus enrollments over time, or even for deriving the expenditures-per-student ratio.

Some of the existing OLAP tools support multi-cube join by providing wizards for specifying the desired view. Our approach simplifies the creation of a multicube by allowing to join the cubes visually “on-the-fly”. The basic idea is to prompt the user to specify the data cubes for joining and to generate a joined navigation framework for the specified multicube. Our currently implemented system already contains the materialized multicubes for each pair of cubes which share at least one dimension. In the future we plan to enable dynamic computation of user-defined multicubes.

The resulting navigational hierarchy, presented in Figure 6 (right), contains the list of all available dimensions and measures, “un-nested” from their respective cubes. Moreover, redundant navigation paths are eliminated, similarly to the fact-dimensional schema in Figure 3.

Each data cube is assigned a unique color icon. Belonging of any dimension or measure to a cube is then specified by marking it with the respective cube’s color mark. Consequently, any shared dimension node will have multiple color marks. The only valid disaggregation operations in a multicube are those along the shared paths in dimensional hierarchies, whereas the remainder of the navigational hierarchy can be used solely for filtering the queried data subset.

Shared measures are specified using a measure definition wizard. The navigation in Figure 6 (right) shows a user-defined multicube measure *euro per student* (specified as $SUM(amount)/SUM(heads)$ in SQL). Figure 8 shows a simple (plain values in the nodes) decomposition tree, drawn by splitting the above measure along *semester* and *faculty*.

5.2 Context-Aware Navigation Hierarchy

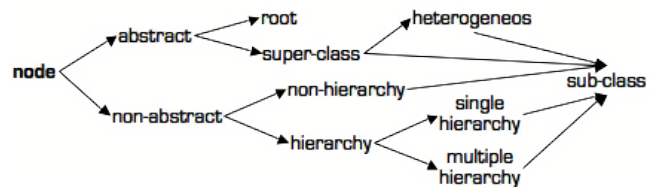


Figure 7: Compatibility of dimensional node types and roles

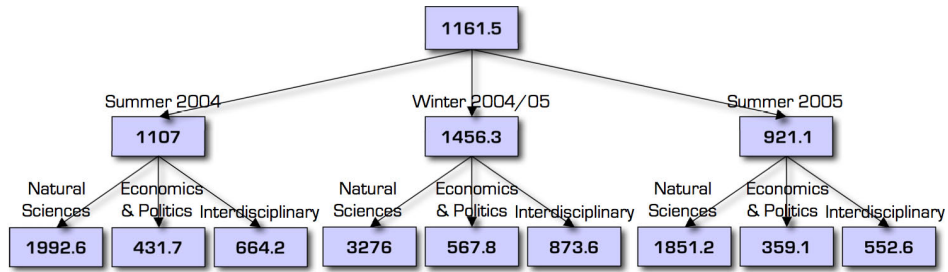


Figure 8: Decomposing a joined measure euro per student

Unlike ProClarity decomposition tree with its node-wise expansion, we pursue level-wise disaggregation where dragging a dimensional node from the navigation into the visualization by default generates the entire tree level. The latter, so called “aggressive”, decomposition is extremely efficient for gaining insight into the entire data set, but may be unfeasible if the task is to investigate a very small subset of interest at each level. Node-by-node expansion can be imitated by filtering the data prior to each decomposition step, but this work-around is obviously too tedious.

Our proposed solution is to explicitly enable bi-directional switching from level-wise to node-wise decomposition at any point of interaction via a corresponding menu option.

In the level-wise mode, each dimensional level can be used as a split axis only once. As the only valid direction of querying is a drill-down, not only the split axis itself, but also its entire upward path becomes invalid for further interaction. To prevent erroneous “roll-up” attempts, the navigation hierarchy simply adapts itself after each split by disabling (making un-draggable) the path down to the split node. Figure 9 shows the state of the navigation after splitting along term and continent. Additionally, those levels which data has been used as a filter, have a changed data-view icon.

In the node-wise mode, the split axis is dropped into the area of the tree node which has to be expanded. To expand several nodes at once, those need to be marked prior to dragging the split axis. In this scenario, a tree level no longer corresponds to a single dimensional axis as “sibling” values are allowed to be decomposed along various paths. Each node in such tree has its own set of still valid decomposition steps. Therefore, the entire navigation remains active and the validity of an attempted split is determined at the moment of dropping the chosen split axis into the visualization.

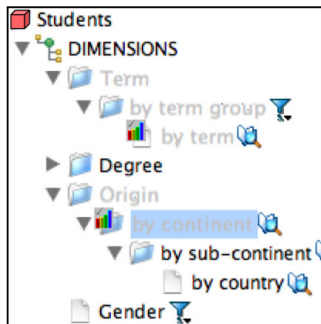


Figure 9: Context-aware navigation.

6. VISUALIZATION AND INTERACTION

Enhanced decomposition tree emerges from combining two classes of visualization techniques: 1) a hierarchical layout for arranging the nodes of each disaggregation step, and 2) a non-hierarchical presentation inside the nodes. In the abundance of existing approaches to visualizing multidimensional data, it is necessary to define a subset of meaningful combinations of visual metaphors for decomposition trees.

The ultimate success of applying any particular visualization technique depends on the type of task to be solved, the kind of data to deal with, as well as on the user’s expertise and preferences. Therefore, our aim is to define a flexible framework with options to select from, rather than to fully pre-configure the available layouts. We proceed by discussing some of the important evaluation criteria.

User acceptance. Node-link trees in a classical hierarchical layout are familiar to any analyst and enjoy high popularity for presenting data hierarchies. Sibling nodes, offering the data of the same granule, appear aligned and can thus be conveniently compared. Descendant subtrees are also aligned w.r.t. their roots, providing a clear view of the hierarchical relationships for any node.

As for non-hierarchical presentation inside the nodes, classical chart views enjoy great user acceptance for their simplicity and ease of interpretation. As the data subset of a single node tends to be rather small and 2-dimensional (measure axis and the inner split dimensional axis) there is no need to employ more complex metaphors at the node level.

Display utilization. The issue of visual scalability, i.e. the capability of a visualization to effectively display large datasets, is crucial in the context of hierarchical exploration of multidimensional data. Standard node-link tree layout is extremely wasteful on the display due to allocating equal area to each tree level. Radial and balloon node-link trees scale somewhat better. Space-filling tree layouts achieve compactness by scaling down the size of nodes and allotting progressively less space for nodes further down on the tree. Unfortunately, such layouts are limited to presenting single values in their nodes and quickly run out of space for displaying any labels or values in the lower levels of the tree.

Our approach takes advantage of space-filling methodology at the node level. We improve the compactness of bar-charts by replacing equal-width bars with equal-height bars, thus transforming the whole chart into a space-filling rectangle, as proposed by Keim et al. in [3]. The tree view shown in Figure 1 is an example of space-filling bars.

Task type. The type of the analytical task at hand may impose additional requirements on the exploration technique. For instance, disaggregation of cumulative aggregations

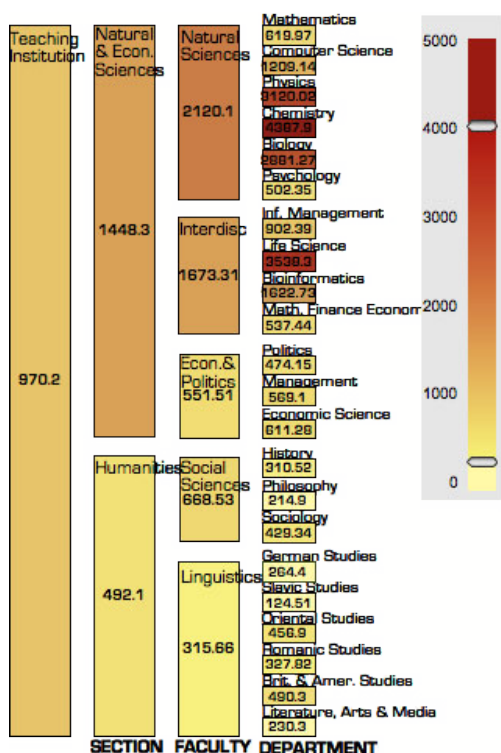


Figure 10: Hierarchical HeatMap with a unipolar colormap

gates, i.e. obtained by applying SUM or COUNT function, can be intuitively performed using space-filling bar-charts, whereas percentages can be better shown as pie-charts. Notice that area-based techniques are not applicable to measures that contain negative values; in such cases length-based bar-charts become indispensable.

Choosing a non-cumulative aggregation function, such as MIN, MAX, or AVG, generates values that remain within comparable ranges across the entire decomposition hierarchy. This observation opens up an option of applying uniform scale at all levels, for instance, to map the values to color and use the HeatMap approach which presents values as colored tiles. Figure 10 shows a hierarchical HeatMap obtained by decomposing the amount's average value successively along three granularity levels of teaching institution. Bottom-level nodes are represented as cells in an array; higher level nodes are shaped as rectangles spanning the width of their subtrees, similarly to the multiscale matrix pattern for pivot table visualization presented in [10], reduced to a single hierarchical axis. A linear unipolar colormap based on the increasing color intensity helps to see the overall behavior of the average and to immediately identify outliers. Sliders at the colormap's poles are useful for dynamically adjusting the visualization's sensitivity to outliers.

7. CONCLUSIONS

We presented a visual framework for decomposing OLAP aggregates along arbitrary dimensional paths using hierarchical visualization techniques. Since explorative analysis is driven by the insights acquired in the course of interaction, hierarchical visualization layouts are especially fitting due to their natural preservation of the entire interaction.

We introduced a class of OLAP-aware hierarchical visual layouts called *Enhanced Decomposition Tree* which can be generated with a few mouse clicks via a schema-based navigation. Presented techniques support “speed-of-thought” analysis by revealing a multiscale view of the data disaggregated along arbitrary dimensional axes. Node-link tree layout is employed to map the disaggregation relationships between the sub-aggregates of various granularity. Non-hierarchical data subsets inside the nodes are presented using easy to interpret chart techniques. By providing various hierarchical layouts, visual metaphors, and interaction options we account for different tasks types.

The directions of our future work include: 1) adding visual approaches for exploring temporal and spatial dimensions, 2) extending the navigation to support symmetric treatment of measures and dimensions, and 3) to further investigate the applicability of space-filling and pixel-based methodologies to improve the visual scalability of our techniques.

8. REFERENCES

- [1] R. Brath and M. Peters. Information visualization for business - past & future. *DM Review*, January 2005.
- [2] S. Eick and A. Karr. Visual scalability. *Journal of Comp. & Graphical Statistics*, 11(1):22–43, 2002.
- [3] D. Keim, M. Hao, U. Dayal, M. Hsu, and J. Ladisch. Pixel bar charts: A new technique for visualizing large multi-attribute data sets without aggregation. In *InfoVis '01: Proc. of the IEEE Symposium on Information Visualization 2001*, page 113, 2001.
- [4] A. S. Maniatis, P. Vassiliadis, S. Skiadopoulos, and Y. Vassiliou. Advanced visualization for OLAP. In *DOLAP '03: Proc. of 6th ACM Int. Workshop on Data Warehousing and OLAP*, pages 9–16, 2003.
- [5] S. Mansmann and M. H. Scholl. Extending visual OLAP for handling irregular dimensional hierarchies. In *DaWaK'06: Proc. of 8th Int. Conf. on Data Warehousing and Knowledge Discovery*, pages 95–105, 2006.
- [6] Proclarity Analytics 6, 2006. Online: <http://www.proclarity.com>.
- [7] V. K. Pang-Ning Tan, Michael Steinbach. *Introduction to Data Mining: Concepts and Techniques*. Addison Wesley, 2006.
- [8] Report Portal: Zero-footprint olap web client solution. *XMLA Consulting*, 2006. Online: <http://www.reportportal.com>.
- [9] C. Stolte, D. Tang, and P. Hanrahan. Query, analysis, and visualization of hierarchically structured data using Polaris. In *ACM SIGKDD '02: Proc. of 8th Int. Conf. on Knowledge Discovery and Data Mining*, pages 112–122, 2002.
- [10] C. Stolte, D. Tang, and P. Hanrahan. Multiscale visualization using data cubes. *IEEE Trans. on Visualization and Comp. Graphics*, 9(2):176–187, 2003.
- [11] K. Techapichetvanich and A. Datta. Interactive visualization for OLAP. In *ICCSA 2005: Proc. of the International Conference on Computational Science and its Applications (Part III)*, pages 206–214, 2005.
- [12] S. Vinnik and F. Mansmann. From analysis to interactive exploration: Building visual hierarchies from OLAP cubes. In *EDBT 2006: Proc. of 10th Int. Conf. on Extending Database Technology*, pages 496–514, 2006.