The road to highlights is paved with good intentions: envisioning a paradigm shift in OLAP modeling

Panos Vassiliadis
Univ. Ioannina
Ioannina, Greece
pvassil@cs.uoi.gr

Patrick Marcel
University of Tours
Tours, France
patrick.marcel@univ-tours.fr

ABSTRACT

In this vision paper we structure a vision for the Business Intelligence of the near future in terms of a model with novel concepts and operators. We envision systems where the end-user requests information at a very high level, expressed as his intention to discover information, and the system transforms this request to the concrete execution of algorithms in order to compute, visualize and comment data and important highlights among them as an answer to the information request made by the end-user.

1 INTRODUCTION

How will BI look like 10 years from now? What foundations should academia build in order to *rigorously* support the building of tools, the optimization of OLAP sessions, and the training of new scientists around a logical Paradigm? In this vision paper, we make a first attempt to revisit the foundations of OLAP and BI in an attempt to address the aforementioned questions.

To start with, it is worth to shortly revisit the evolution of OLAP and BI modeling so far.

- At the beginning of time, people would be working with relational queries and recordsets, returned by these queries. This treatment of BI was very DBMS-oriented, as the focus of attention was on what the DBMS can do for the users [8].
- Then, both the scientific community and the industry understood that it is possible to provide a layer of abstraction on top of the database. So, users would deal (on-line) with cubes, rather than with traditional database data, which gives a very elegant simplification of the data to the user. The operations would be cube-oriented, one level of abstraction above the database operations and would involve roll-ups, drill-downs, etc. (see for example [15]).
- Rapidly, these abstractions became even closer to the way the multidimensional data space was navigated. Research proposed advanced operators permitting *discovery driven analysis* were basically combinations of OLAP primitives [16–18].

What is the vision for the new generation of BI tools? *Broadly speaking, we envision the redefinition of what an OLAP query is, both with respect to what users ask the system and with respect to what the answer entails.*

- Concerning the aspect of what users can ask, we wish to further simplify the operations and choices that the users face and add an extra layer of abstraction. Specifically, we propose to replace cube operators with intentional operators over the data – in other words, with an "algebra" of operators closely to the user's intentions. Instead of the users operating in terms of roll-up and drill-down with the cube, we wish to empower them to state their *intentions* with respect to an operation. For example, instead of saying "drill down this cell one level", the user might ask "explain the drop in the sales of this product", or "is the main drop in sales observed at the best selling city also observed in similar cities?"; as another example, instead of issuing a roll-up operator, the user can state "verify whether a trend that I observe in a certain context still holds in general".
- Concerning the aspect of what the answer to a query is, we believe we can no longer remain with "sets of tuples" as the answers to queries. We can envision BI tools where the answer to a query includes (i) a set of tuples accompanied with appropriate visualizations, (ii) the automatic mining of *models* and patterns, (iii) the extraction of important "jewels" hidden in the result (which we call *highlights*), and naturally, (iv) advanced, intuitive reporting.

Contributions and Benefits. *The main contribution of this paper is that it structures a vision for the Business Intelligence of the near future in terms of a model with novel concepts and operators. We aim our definitions to be broad enough, yet as precise as possible; at the same time, we want to link them as much as possible to the intentional nature of the next generation of BI tools, where the end-user requests information at a very high level and the system transforms these requests to concrete execution of algorithms in order to compute, visualize and comment data and important highlights among them as an answer to the information request made by the end-user.*

Why bother? We are convinced that after 50 years of naive query answering, it is now time to replace it with effortless, automatic insight gaining from the user. Instead of making the end user dig into sets of records, we can increase productivity and the understanding of the essence of the data by using two pillars, one devoted to querying and another devoted to answering, specifically, firstly, by allowing the user to focus on *high-level goals* of information acquisition, rather than details of what data to bring in, and secondly, by providing focus-points in the answers that will move the user effort from manual "jewel mining" to addressing the insights gained.

Several other practical benefits can be envisioned. Our motivation was partially triggered from the problem of generating meaningful, artificial session logs for experimental reasons (e.g., like [14]). Moreover, such efforts also make sense in other domains, e.g., to search Web data sources [3]. Finally, note that this algebra can be seen as a first step towards addressing some of the open challenges proposed by the research community, like e.g., in [9], namely *the lack of declarative exploration languages to present and reason about popular navigational idioms*, or the various challenges raised by [5] around the benchmarking of interactive data exploration by measuring user's gain in terms of insights.

2 THE VISION

An *OLAP session* is a sequence of *dashboards* that the analyst sees, each with its own information, including data, charts and informative summaries of KPI performance. The sequence is produced by the actions of the analyst that changes the contents of the dashboard by requesting more information on the basis of a set of *operations* made available to him by the tool.

2.1 Preliminaries on multidimensional modeling

For lack of space, we do not go to the details of data modeling. We closely follow the model of [7] and constrain ourselves to mention a textual summary. As typically happens with multidimensional models, we assume that *dimensions* provide a *context* for facts [11]. This is especially important if combined with the fact that dimension values come in *hierarchies*; therefore, every single fact can be simultaneously placed in multiple hierarchically structured contexts, providing thus the ability to analyze sets of facts from multiple perspectives. The underlying *data sets* include *measures* that are characterized with respect to these dimensions. *Cube queries* involve measure aggregations at specific levels of granularity per dimension, along with filtering of data for specific values of interest.

We model an OLAP session as a directed graph $G(V,E)$. The set of vertices of the graph represent states of the user's session –practically the dashboard screens the user is receiving by the OLAP tool– and the arcs represent transitions among states.

2.2 Intentional Queries: The Transitions of a Session

Before describing in detail our vision on dashboards and their internals, we start with the deeper essence of the *intentional* nature of our proposal: user operations, or, equivalently transitions between the states of an OLAP session. The main idea behind transitions is that we move from a concrete model of *logical* operators like roll-up's and drill down's, to an *intentional* model where the user expresses in terms of operators for high-level requirements like "explain a certain phenomenon", "predict the future values" and this high-level requirement has to be *automatically* translated to specific OLAP and Data Mining operators/algorithms that will carry the answer. This can also facilitate greatly the extraction of highlights, as the user's goal is explicitly stated to the system.

We organize the transition operators that can express the users' requirements for extra information, in *families*, i.e., a taxonomy of types for our transitions, which we call *transition types. In contrast to other models where a transition would practically be a query (relational or multidimensional), in our model, a transition characterizes the intention of the user with respect to her information need.* Each family is a collection of transition operators with the same high-level intention, but of course, different semantics at the details. This set of families intends to cover (a) traditional OLAP operators, (b) various contributions of the literature, in particular discovery driven analysis operators [16–18], Cinecubes acts [7], and operators for exploratory search in the web [3], as well as (c) novel operators that can be developed by the research community in the future. Before detailing these families, we mention Figure 1 that intuitively describes an analysis phrased with some transitions, using the Star Schema Benchmark data cube [13]. The annotation and highlight generation of the data is explained in Section 2.3.

Compare. A transition of type *compare* is used to retrieve more data to be compared with what the current dashboard displays. Intuitively, transitions in this family ask for bringing in more data to the ongoing analysis. The family includes OLAP operators pivot, drill-across, the augment operator of [3], the put-in-context act of Cinecubes [7]. For instance, on the example of Figure 1, the put-in-context [7] is used twice (top left vertical path in Figure 1), first, to compare the sales for a category of products (*MFGR*#5) in a given supplier country (Argentina) to that of past years (indicated as ❷), and second to compare these past results to the results for sibling countries in the dimension Supplier (indicated as ❺).

FocusOn. A transition of type *FocusOn* is used to exclude data from the current analysis. Transitions in this family focus on a particular cell, or group of cells. The family includes the OLAP slice/dice operation, the take operator of [3], and also personalization operators like skyline, winnow, etc. [10]. For instance, in Figure 1, a slice/dice operation is done to focus on sales results for part *MFRG*#51 in years 2015 and 2016 (indicated as ❼).

Abstract. The transitions of this family are used to reduce the information load of the dashboard, by abstracting them in a more compact form. The family includes the traditional roll-up operator, as well as data mining operations like clustering, frequent itemset mining, etc. For instance, in Figure 1, sales results for years 2011 (not all shown in previous dashboards) to 2016 in Argentina are abstracted in 2 classes based on a result threshold (indicated as ❹).

Analyze and Explain. A transition of type *Analyze and Explain* is used to understand the cause of a phenomenon displayed in the current dashboard and to explore what is displayed at finer levels of detail. With respect to the analysis part, the family includes the drill-down OLAP operation, and the details act of Cinecubes [7]. With respect to the explain part, the family includes Diff [16] (interesting drill downs at various granularities explaining the content of two cells) and the operators of Cariou & al. [4]. For instance, at the top of Figure 1, a drill-down from countries to cities is used to analyze sale results by cities (indicated as ❶), and, at the left bottom of the figure, a Diff is used to explain an important drop of sales in Argentina in year 2016 compared to year 2015, for the specific category of product *MFGR*#5 (indicated as ❻).

Verify. A transition of type *Verify* is used to check that a potential trend or pattern currently observed can be generalized to broader contexts, e.g., we can check whether it occurs at coarser levels of detail. The family includes Relax operator[18] (interesting rollups at various granularities agreeing or contradicting a trend observed in the content of two cells), the verification of outlier data points in broader context than the current, etc. For instance, in Figure 1, a Relax operation is used to verify whether the drop of sale results for category *MFGR*#5 in Argentina, in 2016, also holds for all products sold (indicated as ❸).

Predict. A transition of type *Predict* is used for predictive analysis of the displayed data. Typically this involves the entire set of timeseries forecasting methods, as well as classification methods used for assessing future events [1]. For instance, in Figure 1, sales results of product *MFGR*#51 for year 2015 and 2016 are used to issue predictions for year 2017 in Argentina cities (indicated as ❽).

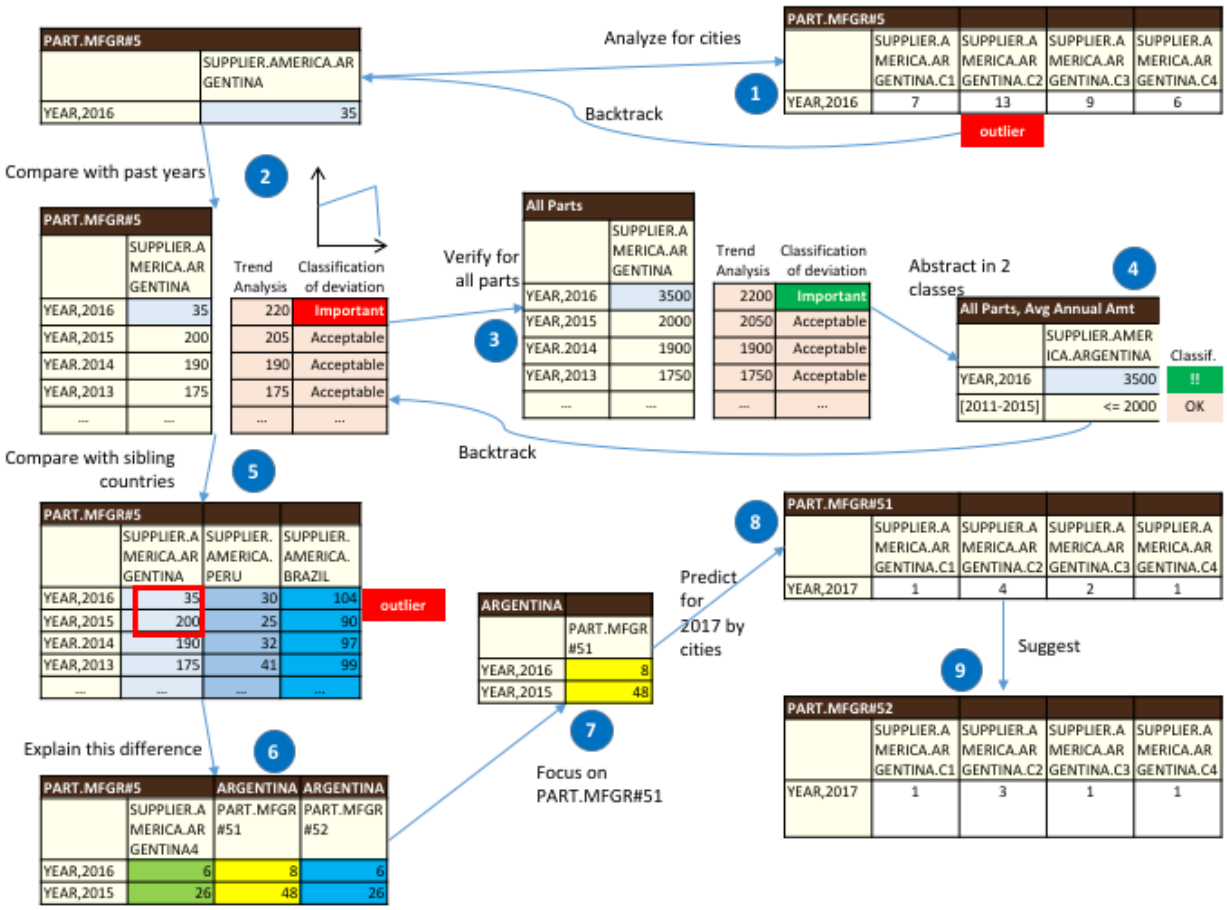

Figure 1: A BI session phrased with some of our transitions. Numbers show the sequence of dashboard generation. Colored cells are results of data analytics operations annotating the dashboard data.

Suggest. A transition of type *Suggest* is used to ask for guidance with respect to where to navigate next in the multidimensional data space. The family includes Sarawagi's Inform [17] (Interesting drill downs at various granularities, deviating from trends that what was already observed), or query recommendation techniques [12]. We note that this kind of transition should incorporate risk control mechanisms to prevent false discoveries [2]. For instance, in Figure 1, a suggest operation is used to automatically find neighboring sale results deviating from the results of the previous operation (indicated as ❾).

Concluding with transitions, we emphasize that an open research challenge is to define an algebraically closed *language of intentional operators*, each of which can be translated automatically to the execution of *concrete* query and data mining operators. The synthesis of these operators can then create entire *stories*. Assume the user, being at state ❷, would like to: "*verify* whether the distribution of sales for mfgr#5 in Argentina from 2011 to 2016 *still holds in general* for all parts, and build a 2 classes model for it, then backtrack to parts and compare with sibling countries, and finally *explain* the highest country-wise difference." This request would correspond to the following statement in the intentional language: $explain_{highlight:MaxDifference}($
$compare_{siblingCountries}($
$backtrack_{showTrend}($
$abstract_{2Classes}($
$verify_{allParts,showTrend}(❷)))))$.
An optimizer would then automatically translate this request into the sequence of operations: roll up, cluster, backtrack, Cinecube's put in context, Diff and optimize it to produce a data story using dashboards ❷ to ❻.

2.3 Dashboards: The states of a session

A state is a dashboard that the user sees. In principle, each dashboard is ultimately based on the generating data provided by as a finite collection of queries, posed to the underlying database. A sharp distinction of our approach compared to previous models is that we do not restrict ourselves to data for each state but accompany them with a set of interesting findings, which come in two flavors, specifically, (a) in terms of *models*, i.e., results of data mining or machine learning algorithms applied over the data of a dashboard's state, and, (b) important findings that accompany the dashboard, to which we refer as the *highlights* of the state.

Take for example, the small scenario of Fig. 2. The dashboard is based on a simple cube, as its generating data, involving *Product*, *Region* and *Year* as dimension levels and *Sales* and *Cost* as (aggregate) measures. Then, the dashboard automatically computes the *Benefit* as the application of a simple arithmetic function, specifically, the difference of the two measures. A second step

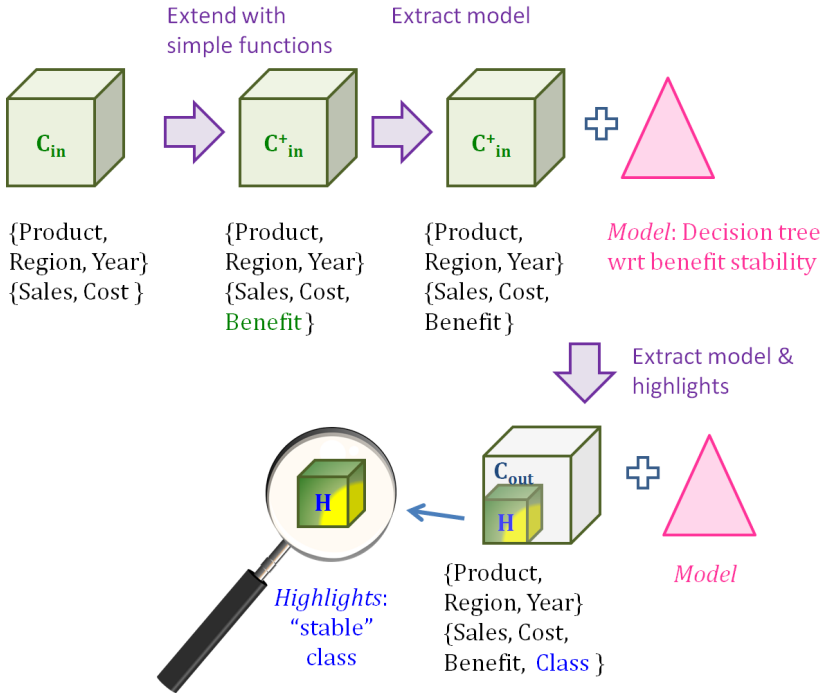

Figure 2: Model extraction, data annotation and highlight production: an example

involves the building of a model, and specifically the classification of sales with respect to their stability of the benefit measure. Finally, highlights are extracted on the basis of this model, and specifically, these highlight are the cube cells pertaining to the class *stable benefits*. Observe how the final data are extended with the *class* attribute linking them to their specific model counterpart.

Steps. In order to construct and visualize a dashboard, we envision several computations taking place. Here is the sequence of the performed actions:

(1) First, the queries of the state's dashboard are issued and their results, the *generating data of the dashboard*, are computed. Any straightforward computations for extra, derived columns of the dashboard (e.g., *gain=price-cost*) are performed too.
(2) Then, the available data are fed to model extraction algorithms for the computation of *models* abstract, summarize and provide patterns and insights for the data.
(3) The potentially large amount of data and models computed has to be ranked and assessed on their interestingness for the analyst; the most important findings are classified as the dashboards *highlights* to be used for providing the main insights and the main directions for future transitions by the analyst.
(4) The above are accompanied by visualization, text construction and reporting tasks that aim the process of understanding and communicating the main findings.

The generating data of the dashboard. The results of the underlying queries are the basis for the subsequent computations that take place for the construction of the dashboard: this is why we refer to them as the *generating data* of the dashboard.

Models. In difference from the state of the art in OLAP modeling, in our approach, we believe that the static results of aggregate queries, and their visualizations with charts and speedometers will simply be not enough in the near future. The *automatic* assessment and critical characterization of the presented data will be part of the BI of the near-future. See some simple cases based on the example of observing sales data of an international company:

- Sales data will be *automatically* characterized with respect to a decision tree that classifies them (e.g., as "successful", "risky", "potentially hazardous" etc)
- Sales per country will be *automatically* clustered to reveal similarities and differences, as a first step towards understanding outliers and non-expected behavior
- Aggregate sales over significant periods will be fed into time series analysis and forecasting methods to *automatically* detect trends, seasonalities and to deduce future values

We consider the plugging of data analysis algorithms in the backstage of a dashboard as an indispensable part of BI. These algorithms can range from very simple ones (e.g., finding the top values of a cuboid, or detecting whether a dimension value is systematically related to top or bottom sales) to very complicated ones (like, for example, outlier detection, dimensionality reduction, etc). Most importantly, as the operation of the algorithms will likely be as transparent as possible to the end user, their execution will require an almost automatic tuning of their parameters. The findings of these algorithms will be **models** of the data that are typically (not always) used to **annotate the existing data** with characterizations and offer *focus points* to the visualization of the dashboard (forecasts, outliers, dimension values that dominate top or bottom measures, . . .). The models themselves give a multitude of results. However, some of these results indicate that a part of a dashboard's data are of important *interestingness* value to the end user. Due to that, we collectively refer to the important results of the execution of these algorithms as **highlights**, in an attempt to show that the aim is to enrich

the current data-intensive dashboards with knowledge that is worth exploring or using for decision making.

We are going to treat model extraction algorithms as "black-box" algorithm without probing into their internals, and, most importantly, without assuming any particular properties for their output. What does a model extraction algorithm do? Basically, the algorithm receives as input (a) a set of input data, and, (b) a set of execution parameters that have to be fixed for the algorithm's execution. Without loss of generality, we can assume that a subset of these parameters will be bound to string or numerical values and the rest will be mapped to attributes of the input data. The output of a model extraction algorithm is a *model* of the input data. Depending on the algorithm, the result differs. For instance, a descriptive model built using unsupervised clustering is basically just a labeling of each cube's cells, while a predictive one allows enriching the cube with predictions and comes with an accuracy score. In summary, the main properties of a model extraction algorithm is outlined as follows:

(1) Input: a set of input data, which is the result of an extended cube query set of the dashboard, along with a fixing of the algorithm's input parameters
(2) Output: a (possibly complex) result composed of (a) a model of the input data, and, (b) several characterizations of it (precision, strength, p-value, ...)

Each model type can be arbitrarily complex and consists of a set of *model components* of versatile nature. Examples:

- A time series splits each of its points to 3 measurements, specifically error, trend and seasonality (practically creating 3 times series in the place of one, whose sum reconstructs the original one).
- A clustering scheme includes a set of clusters, each with a set of tuples that constitute it, along with a centroid.
- A classification decision tree includes a tree structure, best expressed as the composition of a set of paths, leading to characterization classes; again, each class comprises a set of generating tuples that pertain to it.

Typically, each such component, as well as the model on its entirety is accompanied with a set of metrics of its statistical power. Given a model type T, we denote its components with $T_C = \{tmc_1 \ldots tmc_m\}$. We avoid resolving the internals of the composition of the *tmc* parts and treat T_C as a *set* of components. This is realistic, as we can always mask a part-of relation as a set of constituting components, even a recursive one, by allowing an extra composition relation to provide the semantics of the part of-relationship. Apart from this simple modeling, attempts to relationally code mining results already exist [6]. In other words: assuming a model M of type T (e.g., a particular cluster set M of type $ClusterSet$), we can use its components (e.g., the clusters of M, $M = \{cluster_1, \ldots, cluster_m\}$) as fundamental parts of subsequent analyses, and, at the same time, link each of these components to their underlying data.

But, then, how do we handle the heterogeneity of different model types? Clearly, a cluster is inherently different from a decision tree or the formula for a trend. Is there a unifying model to cover them all? The unifying essence of all the plethora of diverse model types is that, at the end of the day, all of them are annotations of the original data. At the end of the day, every component of a complex model type (be it a cluster id, a path in a decision tree and a resulting class, a characterization of the top-k tuples, or a trend formula): (a) refers to a subset of the input data and vice-versa, and, (b) refers to the overall model via a part-of relationship. So, once a model of the underlying data is available, our solution to the problem is to provide a distinct identity (an *id*) to the components of a complex type T, retain the membership/part-of relationships of T separately and *annotate or characterize the data with respect to the part of T that pertains to them.* In fact, this step can be blended within the model extraction itself. Examples of such annotation follow:

- Assuming a time series model that splits a time series to $trend$, $seasonality$ and $noise$, these attributes can be appended to the generating data set.
- Assuming a cluster model, the generating data can be annotated with the *id* of the cluster to which they belong.
- Assuming a classification model, the input data can be labeled via an extra attribute with respect to the class(es) of the model to which they belong.
- Assuming a model of top-k values of a measure, the input data can be annotated with their rank, if they belong in the top-k set and they have been ranked.

A notable property of our modeling is that we require model components to be directly mapped and linked to their generating data in a bidirectional mapping, so that the end-user can navigate back and forth between cube cells and their models.

Highlight Production. As already mentioned, the set of *highlights* of the dashboard is a set of *important findings* that accompany the dashboard. These can be findings of any nature, e.g., important outliers in the contents of the dashboard's data, all the tuples belonging to a certain class of a classification scheme, the top or bottom values of a measure, etc. It would be straightforward (and doable with our modeling) to treat all the contents of an extended query as highlights. However, we would like to stress that we want to restrict our attention to the ones that are really important for the end-user.

Other "local' operations. Once all computations are done, there are several tasks to be taken care of before automatically constructing the dashboard. We envision that the development of principled methods for (a) the visualization of the dashboard data in various ways and (b) the creation of reports, via data storytelling methods will present major challenges for the future.

Visualize. Operations in this family compute the contents of visual representations (bar charts, speedometers, scatterplots, ...) on the basis of the current contents of the dashboard. We consider them to be local operations as they are performed without any further interaction with data external to the ones already retrieved from the dashboard.

Data storytelling. Data storytelling is a novel trend that seems to carry significant weight in the way the future BI will look like. The main idea involves the automatic generation of a textual report from the data of a particular dashboard or of an entire OLAP sessions. There are already some academic efforts, like Cinecubes [7] as well as some tools (see [19] for a nice discussion).

3 PATHS FOR FUTURE RESEARCH

This paper is a vision paper describing, in broad terms, a potential future for OLAP, to strengthen its place as the corner stone of BI. Our call to arms to the research community can be structured along several lines. *Population of the families of the transition operators with concrete operators.* Each new operator (or each new formalization of existing operators) should hopefully carry (a) clear semantics, (b) execution algorithms as well as their optimizations, (c) fine tuning algorithms for the parameter fixing of

the introduced algorithms and, equally importantly, (d) a graceful linkage to the overall model proposed here (in an attempt to be able to gracefully plug it in the respective BI tools). *Automation of transitions and tuning*. We believe this to be the most important piece of the puzzle. How do we fully automate what model and highlight extraction methods should be employed? This includes the possibility of predicting interesting results for the end-user, which in turn, requires appropriate models of *interestingness*. Along with the necessary run-time optimizations, all these tasks provide challenging and open problems of significance practical importance. *Key Performance Indicators* (at least in the way they are used today) are examples of simple model extraction, and by definition, KPI's are highlights (or else they wouldn't be "Key" indicators). Linking them explicitly to a model for OLAP is a necessary add-on for a comprehensive view of what OLAP does. *Benchmarking and tools*. A reference free, open-source tool and a reference benchmark for the future BI (involving data, model and highlight extraction requests and sessions) can be a really handy tool for the research community (otherwise, each new paper will need to improvise on its experimental assessment). A tool will also trigger other research directions, like e.g., the incorporation of research results on natural language processing to accept the user requests, visualizations to depict the models and highlights, etc.

REFERENCES

[1] Charu C. Aggarwal. 2015. *Data Mining - The Textbook*. Springer.

[2] Carsten Binnig, Lorenzo De Stefani, Tim Kraska, Eli Upfal, Emanuel Zgraggen, and Zheguang Zhao. 2017. Toward Sustainable Insights, or Why Polygamy is Bad for You. In *CIDR 2017, 8th Biennial Conference on Innovative Data Systems Research*.

[3] Alessandro Bozzon, Marco Brambilla, Stefano Ceri, and Davide Mazza. 2013. Exploratory search framework for Web data sources. *VLDB J.* 22, 5 (2013), 641–663.

[4] Véronique Cariou, Jérôme Cubillé, Christian Derquenne, Sabine Goutier, Françoise Guisnel, and Henri Klajnmic. 2009. Embedded indicators to facilitate the exploration of a data cube. *IJBIDM* 4, 3/4 (2009), 329–349.

[5] Philipp Eichmann, Emanuel Zgraggen, Zheguang Zhao, Carsten Binnig, and Tim Kraska. 2016. Towards a Benchmark for Interactive Data Exploration. *IEEE Data Eng. Bull.* 39, 4 (2016), 50–61.

[6] Arnaud Giacometti, Patrick Marcel, and Arnaud Soulet. 2011. A Relational View of Pattern Discovery. In *Database Systems for Advanced Applications - 16th International Conference, DASFAA*. 153–167.

[7] Dimitrios Gkesoulis, Panos Vassiliadis, and Petros Manousis. 2015. CineCubes: Aiding data workers gain insights from OLAP queries. *Inf. Syst.* 53 (2015), 60–86.

[8] Jim Gray, Adam Bosworth, Andrew Layman, and Hamid Pirahesh. 1996. Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Total. In *Proceedings of the Twelfth International Conference on Data Engineering*. 152–159.

[9] Stratos Idreos, Olga Papaemmanouil, and Surajit Chaudhuri. 2015. Overview of Data Exploration Techniques. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. 277–281.

[10] Ihab F. Ilyas, George Beskales, and Mohamed A. Soliman. 2008. A survey of top-k query processing techniques in relational database systems. *ACM Comput. Surv.* 40, 4 (2008), 11:1–11:58.

[11] Christian S. Jensen, Torben Bach Pedersen, and Christian Thomsen. 2010. *Multidimensional Databases and Data Warehousing*. Morgan & Claypool Publishers.

[12] Patrick Marcel and Elsa Negre. 2011. A survey of query recommendation techniques for data warehouse exploration. In *Actes des 7èmes journées francophones sur les Entrepôts de Données et l'Analyse en ligne, Clermont-Ferrand, France, EDA 2011, Juin 2011*. 119–134.

[13] Patrick E. O'Neil, Elizabeth J. O'Neil, Xuedong Chen, and Stephen Revilak. 2009. The Star Schema Benchmark and Augmented Fact Table Indexing.. In *TPCTC* (2009-10-28). 237–252.

[14] Stefano Rizzi and Enrico Gallinucci. 2014. CubeLoad: A Parametric Generator of Realistic OLAP Workloads. In *Advanced Information Systems Engineering - 26th International Conference, CAiSE 2014, Thessaloniki, Greece, June 16-20, 2014. Proceedings*. 610–624.

[15] Oscar Romero and Alberto Abelló. 2007. On the Need of a Reference Algebra for OLAP. In *Data Warehousing and Knowledge Discovery, 9th International Conference, DaWaK 2007, Regensburg, Germany, September 3-7, 2007, Proceedings*. 99–110.

[16] Sunita Sarawagi. 1999. Explaining Differences in Multidimensional Aggregates. In *Proceedings of 25th International Conference on Very Large Data Bases (VLDB'99)*. 42–53.

[17] Sunita Sarawagi. 2000. User-Adaptive Exploration of Multidimensional Data. In *Proceedings of 26th International Conference on Very Large Data Bases (VLDB 2000)*. 307–316.

[18] Gayatri Sathe and Sunita Sarawagi. 2001. Intelligent Rollups in Multidimensional OLAP Data. In *Proceedings of 27th International Conference on Very Large Data Bases (VLDB 2001)*. 531–540.

[19] Alex Wright. 2015. Algorithmic authors. *Commun. ACM* 58, 11 (2015), 12–14.