CineCubes: aiding data workers gain insights from OLAP queries

Dimitrios Gkesoulis[a,1], Panos Vassiliadis[b], Petros Manousis[b]

[a] *UTC Creative Lab, Ioannina, Hellas*
[b] *University of Ioannina, Ioannina, Hellas*

Abstract

In this paper we demonstrate that it is possible to enrich query answering with a short data movie that gives insights to the original results of an OLAP query. Our method, implemented in an actual system, *CineCubes*, includes the following steps. The user submits a query over an underlying star schema. Taking this query as input, the system comes up with a set of queries complementing the information content of the original query, and executes them. For each of the query results, we execute a set of *highlight* extraction algorithms that identify interesting patterns and values in the data of the results. Then, the system visualizes the query results and accompanies this presentation with a text commenting on the result highlights. Moreover, via a text-to-speech conversion the system automatically produces audio for the constructed text. Each combination of visualization, text and audio practically constitutes a movie, which is wrapped as a PowerPoint presentation and returned to the user.

1. Introduction

Can we answer user queries with data movies? Why should query results be treated simply as sets of tuples returned by the DBMS as if they would be visualized in an orange CRT of the 70's? So far, database systems assume their work is done once results are produced, effectively prohibiting even well-educated end-users to work with them. Can we do something better?

In this paper, we revise the traditional assumptions of *query answering* in order to raise the issue of *insight gaining*. We serve the purpose of insight gaining in two ways, by demonstrating that

- *it is possible to produce query results that are (a) properly visualized, (b) textually exploitable, i.e., enriched with an automatically extracted text that comments on the result, (c) vocally enriched, i.e., enriched with audio that allows the user not only to see, but also hear*, and,

Email addresses: `dimitris.gesoulis@gmail.com` (Dimitrios Gkesoulis), `pvassil@cs.uoi.gr` (Panos Vassiliadis), `pmanousi@cs.uoi.gr` (Petros Manousis)

[1]Work conducted while in the Univ. of Ioannina

- *it is possible to come up with a working, extensible method that accompanies a query result with the results of complementary queries which allow the user to contextualize and analyze the information content of the original query.*

Interestingly, an insightful sequence of related queries that provide context and depth to the original query, "dressed" with the appropriate visualization and sound, ends up to be nothing else but a data movie where cubes star.

Motivation Yet, what does *insight* mean? In a recent approach, Dove and Jones [1] combine the definitions from the communities of Information Visualization and Cognitive Phycology: whereas the InfoVis community defines insight as "*something that is gained*" (after the observation of data by a participant), psychologists define it as an *"Aha!" moment which is experienced.* Interestingly, the two definitions can be combined in a common view, where once the user works with information, starting with an original state of mind on the current state of affairs, there is an "Aha!" moment, where the user suddenly realizes a new way of looking at the data, resulting in a new mental model for the state of affairs, or else, new understanding [1].

In order to facilitate the "Aha!" moment that creates insight, the scientific community is spending more and more effort nowadays in the area of data analysis. In a recent SIGMOD keynote speech in 2012 [2], Pat Hanrahan from Stanford University and Tableau Software makes a case for visual analytics as the best way to support the data analysis process; whereas the former involves the automatic extraction of information accompanied by the appropriate visualization, the latter can be summarized as follows: "get the data, deliver them in a clean usable form, contextualize them, extract relationships and patterns hidden within them, generalize for insight, confirm hypotheses and errors, share with others, decide and act".

Our goal with the CineCubes system is to provide an extensible tool that acts as the platform that supports the insight generation of the data analysis lifecycle (contextualization, pattern extraction, insight) by producing small stories that make an impression "in a memorable way" to the data enthusiast or the data worker who performs data analysis.

Key contributions. Then, the question arises: *And how can we do that?* In a nutshell, our main result is the introduction of a fully automated and extensible method that allows the generation of a data movie, over an OLAP database, with a simple user query as starting point. In detail, our individual assumptions and contributions can be listed as follows:

- We start with a realistic assumption that empowers us with the ability to address the challenge in a clear setting. We assume the existence of a star schema with clean, reconciled hierarchies of reference data; we also assume that the end users are interested in working with OLAP queries over these data.

- We demonstrate how to complement the original query with additional queries that allow the contextualization and analysis of the original result.

To provide contextualization, we exploit the defining values (i.e., selection conditions) of the original query and automatically generate complementary queries that compare its results with the results of queries having similar values. Practically, this exploits OLAP hierarchies and compares sibling values within the same hierarchy. For example, if the user has scoped the data of interest with selection conditions $Continent = \mathsf{North\ America}$ and $Gender = \mathsf{Male}$, we accompany the original query with queries that compare North America to other continents and men to women. To provide further analysis of the results, we drill in the grouping levels of the original result to see the breakdown of its (aggregate) measures and understand its internal structure. So, for example, if the user originally aggregates information by 10 $Year$ intervals and $Continent$, we provide details by drilling-in to 5 $Year$ intervals and $Country$.

- Whereas the above actions produce a first step towards supporting the contextualization and analysis of the data, we have also implemented a fully automated mechanism for producing patterns and trends within each of the above results. To this end, we introduce *highlight* extraction methods, that operate on the result of a query and discover interesting findings (like e.g., the fact that a column contains a large share of the highest or lowest values of a result, or that a row systematically has higher/lower values than another). These highlights serve also the visual presentation of the data via appropriate coloring of important values.

- Automatic highlight generation is a key contribution and not only for visualization purposes. In this paper, we also demonstrate how to automate the generation of text describing the aforementioned highlight findings (by accompanying each type of highlight with a template text) and how to convert this text to audio (via publicly available text-to-speech conversion software).

- Much like movies, we organize our stories in acts, with each act including several episodes all serving the same purpose. We demonstrate that all the above can be packaged with small programming effort in a PowerPoint presentation, practically presenting a small movie to the user. The emphasis on small programmatic effort is intended: an important goal of this paper is to demonstrate that the technical barrier for someone who would be interested to conduct research on this problem is low. Existing API's for the construction of PowerPoint presentations [3] and for text to speech conversion [4] allow us to produce a PowerPoint presentation programmatically: each query can have a slide where its result is neatly visualized; the slide's notes can contain the text explaining the result and the slide's audio can be produced via text-to-speech conversion.

- Quite importantly, and orthogonally to the above, we have intentionally built our system in a way that is both fully automated and extensible. Extensibility has been a cornerstone of our approach and it is best demonstrated by two extensibility mechanisms, (i) one concerning the generation of the complementary queries to the original question, and, (ii) another concerning the automatic identification of interesting highlights within the results of each query. In this paper, we discuss the points of extensibility of CineCubes in detail.

Target audience and added value. *Who can benefit from CineCubes?* There are many kinds of people that currently work with data in order to deliver a report, a live or a self-running presentation, an on-line talk, or a journal article, and who would all benefit from such a system. Business users with particular questions in mind, are a first such case [2]. People creating self-running presentations [5] (i.e., presentations publicly available for mass audiences without the presenter being involved), either in a film-clip or a slide-show can benefit from a system giving both insights and visual representations. Collaborative work in small groups [5] can benefit from CineCube presentations as they provide the basis to broaden the scope of the original search and lead to new questions to be answered. Journalists nowadays are more and more preparing data-driven articles that involve working with data and using infographics to make a case (see [6] for a large list of examples – typically, New York Times, Washington Post and the Guardian are reference news media for infographics).

Overall, we claim that a*ny data worker creating a report summarizing findings and insights based on data can benefit from CineCubes* in many ways: automated highlight extraction, auxiliary query results, automatically generated text, audio and visual highlight do not only work together to generate contextualization, analysis, probes for further exploration, and ultimately, insight, but also provide a reusable means of precanned text and visual graphics that can speed-up the compilation of the desired report. The results of a user study that we have conducted (Section 5.4) reveal that improvements come in two ways, and specifically, (a) better quality and (b) faster creation of the report.

Novelty. We believe that despite the vast amount of work (plz., refer to Section 6) in the areas of data visualization, query recommendation, pattern mining, and, to a lesser extent, text generation from query results, this paper makes a disruptive contribution *by raising the issue of gaining insight from the data via small data movies (as opposed to traditional, simple query answering) and providing an automated solution to it via (a) auxiliary queries and (b) automated highlight extraction.* The idea of a data movie has been a driver for the compilation of individual techniques in a single, fully automated and extensible packaging: a movie requires episodes in its structure, visual effects and audio; these needs have produced the solution of query sequences, automatic highlight extraction, as well as automatic text and audio generation via simple programmatic APIs. Thus, one should not regard the individual parts of the method as the novel contribution of the paper; it is their principled and extensible bundling in a single, extensible tool that creates a research opportunity and practically new research ground to explore.

Roadmap. In Section 2, we give an overview of the method as well as a reference example. In Section 3, we discuss our method's internals. In Section 4 we present the software architecture of CineCubes with special emphasis on the extensibility aspect and explain the low technical barrier of the method, too. In Section 5, we show experimental results in terms of efficiency and usability. In Section 6, we discuss related work. We conclude with a presentation of open issues in Section 7.

2. Method Overview

2.1. Constructing a CineCube Story

A really useful characteristic of cubes is that *dimensions* provide a *context* for facts [7]. 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 cats from multiple perspectives. At the same time, hierarchies allow the comparison of their members with (a) *ancestors*, (b) *descendants* and (c) *siblings* (children of the same parent). Assume a basic, detailed cube C defined (a) over a set of dimensions $D = \{D_1, \ldots, D_n\}$ and (b) over a measure M. A query Q in our context exploits the multidimensionality of the cube space and can be considered as a quintuple Q=$(C, D, \Sigma, \Gamma, \gamma(M))$ where:

- Σ is a conjunction of dimensional restrictions of the form $D_i.L_j = value_i$ – i.e., constraints that focus the context of the query to certain dimensional values.

- Γ is a set of grouper dimensional level (practically comprising the GROUP BY attribute set in a SQL query), over which the information will ultimately be grouped.

- $\gamma(M)$ is an aggregate function applied to the measure of the cube; again, we restrict ourselves to a single measure.

Given a query Q and its result $Q.RS$, we can create a short "data story" by seeking for answers to the following questions:

0. A first assessment of the current state of affairs. Practically, this requirement refers to the execution of the original query.

1. Put the state in Context. Are the results of $\gamma(M)$ good? What does "good" mean in this case? Typically, we would expect to compare the result of the query Q to the results of similar queries over siblings of the values that appear in the filter list Σ.

2. Analysis of why things are this way. Given a certain cuboid that is the result of a query, we would like to provide some more insight on the presented results; one way to achieve this is to show the breakdown of the contributions of the detailed values to the overall, aggregate value. Practically speaking, this involves drilling-down for each of the involved groupers and presenting the analysis of the internal breakdown for each of the groupers.

Clearly, this set of complementary queries that a story comprises is extensible; existing and novel results in query recommendation (see Section 6) can be progressively plugged in our method in order to produce more informative CineCube movies.

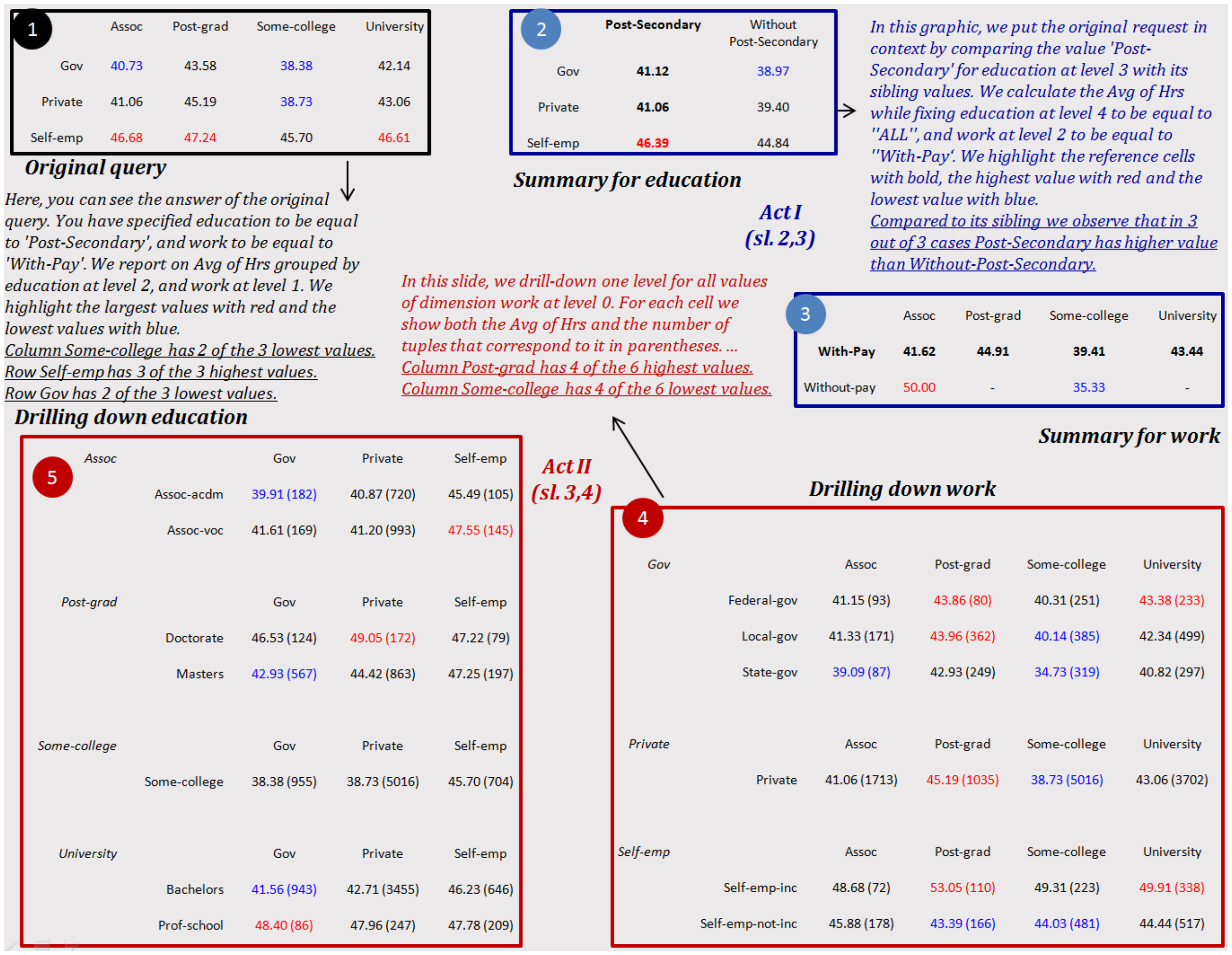
2.2. Running Example

To demonstrate our approach we use an example from the well known Adult (a.k.a census income) dataset referring to data from 1994 USA census. There are 7 dimensions (*Age, Native Country, Education*, *Occupation*, *Marital status*, *Work class*, and *Race*) in the data set and a single measure, *Hours per Week*. We will use a uniform terminology to refer to the dimensions' levels, (L_0, L_1, ..). Also, the ragged dimensions are complemented with values identical to their parent, to make them balanced and fit to the model of [8].

We start with an original query where the user has fixed *Education* to 'Post-Secondary' (at level L_3), and *Work* to 'With-Pay' (at level L_2) and requests the *Avg* of *HrsPerWeek* grouped by *Education* at level 2, and *Work* at level 1. We depict these two dimensions in Fig. 2. We arrange the presentation of the result in columns (*Education*) and rows (*Work*). In Fig. 1, in the slide with the indication ❶, one can also see the actual presentation as a 2D matrix, the visualization interventions (highlighting high and low values with color) and the text accompanying the visual presentation. The text is (a) part of the slide's notes (so that the user can reuse it) and (b) orally voiced as an audio file accompanying the slide. The slide's text is delivered via a set of *highlight extraction* methods that search the 2D matrix for prominent features (high and low values, rows or columns dominating some of these indicatory values, etc).

Once the originally query has been answered, we move on to put it in context. *Act I* of the CineCube movie, including slides ❷ and ❸ (dressed in blue color), performs the following analysis: since there is a selection condition with two atoms (*Education.L3*='Post-Secondary' and *Work.L2*='With-Pay'), we compare each of the defining values with its sibling. So, slide ❷ presents a comparison between the siblings of 'Post-Secondary' at level L_3 of *Education* (specifically, the single value 'W/O post secondary'). The analysis shows that in 3 out of 3 cases people with Post-Secondary education work more (see 1 at top right for the respective text). Similarly, in slide ❸ we relax the constraint on *Work* and compare the value 'With-Pay' with its siblings at level L_2 of *Work* (again the single value 'W/O Pay'). The results are inconclusive; for lack of space we omit the respective text from 1. In both these cases, we did two things: (a) we took a single atomic formula from the selection condition of the original query and replaced it by fixing the defining value to the parent of the original value, and (b) we put the grouping level to the level of the replaced value.

Then, we detail the results of the original query in *Act II* of the CineCube movie. In slides ❹ and ❺ (dressed in red color) we present the results of drilling-down one level per grouper value. Observe slide ❹ as an example (slide ❺ is similar): for each of the values in the rows of the original query (at level L_1 of dimension *Work*) we drill-down one level (at level L_0 that is) and group-by accordingly. For each aggregated cell of the result we also show the number of



1

	Assoc	Post-grad	Some-college	University
Gov	40.73	43.58	38.38	42.14
Private	41.06	45.19	38.73	43.06
Self-emp	46.68	47.24	45.70	46.61

Original query

Here, you can see the answer of the original query. You have specified education to be equal to 'Post-Secondary', and work to be equal to 'With-Pay'. We report on Avg of Hrs grouped by education at level 2, and work at level 1. We highlight the largest values with red and the lowest values with blue.
Column Some-college has 2 of the 3 lowest values.
Row Self-emp has 3 of the 3 highest values.
Row Gov has 2 of the 3 lowest values.

2

	Post-Secondary	Without Post-Secondary
Gov	**41.12**	38.97
Private	**41.06**	39.40
Self-emp	**46.39**	44.84

Summary for education

In this graphic, we put the original request in context by comparing the value 'Post-Secondary' for education at level 3 with its sibling values. We calculate the Avg of Hrs while fixing education at level 4 to be equal to ''ALL'', and work at level 2 to be equal to ''With-Pay'. We highlight the reference cells with bold, the highest value with red and the lowest value with blue.
Compared to its sibling we observe that in 3 out of 3 cases Post-Secondary has higher value than Without-Post-Secondary.

Act I (sl. 2,3)

3

	Assoc	Post-grad	Some-college	University
With-Pay	**41.62**	**44.91**	**39.41**	**43.44**
Without-pay	50.00	-	35.33	-

Summary for work

In this slide, we drill-down one level for all values of dimension work at level 0. For each cell we show both the Avg of Hrs and the number of tuples that correspond to it in parentheses. ...
Column Post-grad has 4 of the 6 highest values.
Column Some-college has 4 of the 6 lowest values.

Act II (sl. 3,4)

Drilling down work

4

		Assoc	Post-grad	Some-college	University
Gov					
	Federal-gov	41.15 (93)	43.86 (80)	40.31 (251)	43.38 (233)
	Local-gov	41.33 (171)	43.96 (362)	40.14 (385)	42.34 (499)
	State-gov	39.09 (87)	42.93 (249)	34.73 (319)	40.82 (297)
Private		Assoc	Post-grad	Some-college	University
	Private	41.06 (1713)	45.19 (1035)	38.73 (5016)	43.06 (3702)
Self-emp		Assoc	Post-grad	Some-college	University
	Self-emp-inc	48.68 (72)	53.05 (110)	49.31 (223)	49.91 (338)
	Self-emp-not-inc	45.88 (178)	43.39 (166)	44.03 (481)	44.44 (517)

Drilling down education

5

		Gov	Private	Self-emp
Assoc				
	Assoc-acdm	39.91 (182)	40.87 (720)	45.49 (105)
	Assoc-voc	41.61 (169)	41.20 (993)	47.55 (145)
Post-grad		Gov	Private	Self-emp
	Doctorate	46.53 (124)	49.05 (172)	47.22 (79)
	Masters	42.93 (567)	44.42 (863)	47.25 (197)
Some-college		Gov	Private	Self-emp
	Some-college	38.38 (955)	38.73 (5016)	45.70 (704)
University		Gov	Private	Self-emp
	Bachelors	41.56 (943)	42.71 (3455)	46.23 (646)
	Prof-school	48.40 (86)	47.96 (247)	47.78 (209)

Figure 1: An excerpt of a CineCubes story over the Adult data set

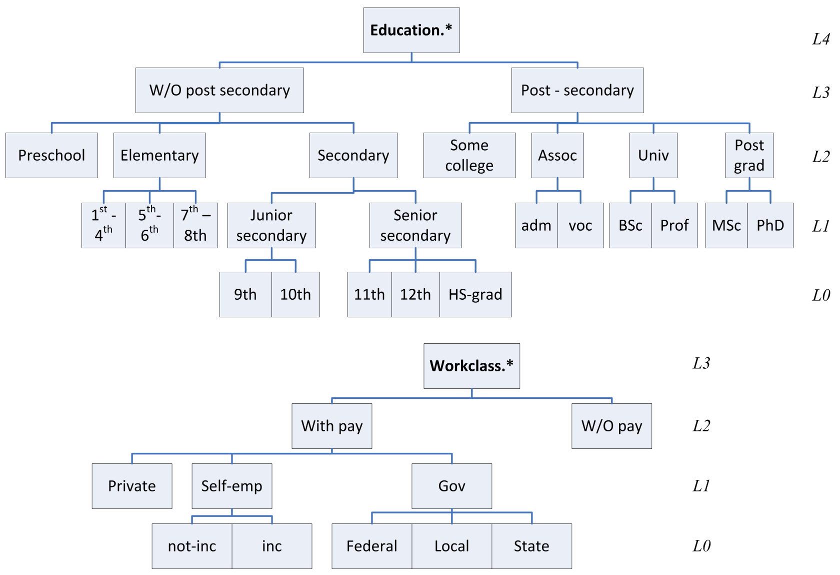


Figure 2: Dimensions Workclass and Education

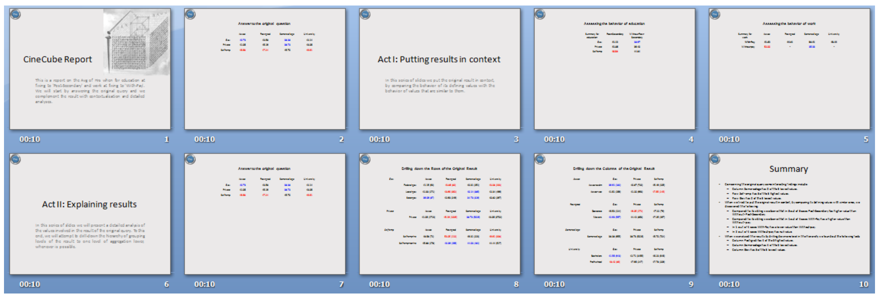


Figure 3: A snapshot of the internal structure of the CineCube movie

detailed tuples that correspond to it, in parentheses. The text is constructed similarly with the previous act and includes a discussion of trends for high and low values along columns and rows.

In the actual presentation that we generate, the set of information-carrying slides is also enriched with transition slides among the acts, explaining the intuition behind them as well as with a summary of the key highlights in the end (see Fig. 3).

One can find information about CineCubes at its web page (http://www.cs.uoi.gr/~pvassil/projects/cinecubes/) and test its functionality by posing queries at a demo site (http://snf-56304.vm.okeanos.grnet.gr/).

2.3. Internal Structure of the CineCube Movie

A typical movie story is structured in approximately 3 acts [9]: the first providing contextualization for the characters as well as the incident that sets the story on the move, the second where the protagonists and the rest of the roles build up their actions and reactions and the third where the resolution of the film is taking place. Each act is composed of sequences of scenes: each scene involves a change in the status of the plot (typically oscillating this status in order to keep viewers interested) and a sequence drives a subset of the plot to a major status update [9].

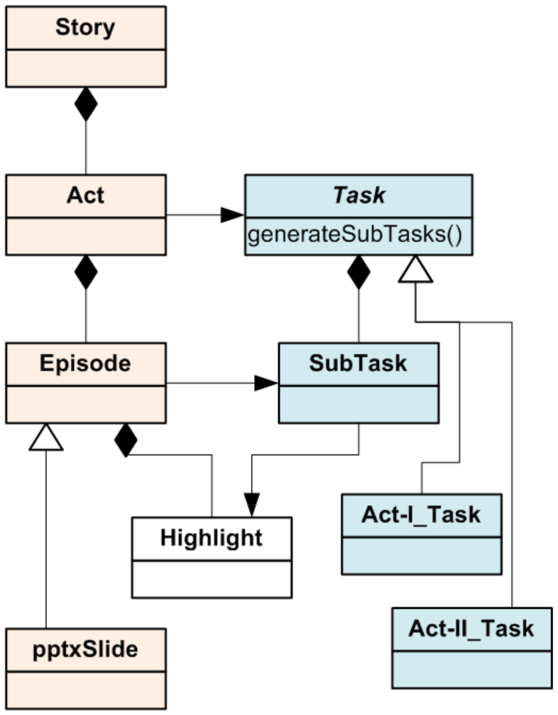


Figure 4: Extensibility mechanism for CineCubes

We follow this traditional structure of a movie in our effort. We are clearly avoiding the temptation to automate a 90' movie; on the contrary, we wish to keep the story short and limited, as we anticipate users will explore several CineCube stories before gathering their results and discoveries from exploring the data. We organize *Acts* in *Episodes*: each episode practically corresponds to a pptx slide (although, we can envision extensions to other formats – e.g., it could be a section in a document).

This *result-based structure* of the CineCube movie (left-hand side of Fig. 4) is accompanied by a *procedural-based structure*, with a set of classes that actually get the job done (right-hand side of Fig. 4). Specifically, the generation of queries (and slides) within each *Act* is delegated to the abstract class *Task*. For reasons of extensibility, *Task* is an abstract class: therefore, we materialize it differently for each kind of *Act* (in Fig. 4 we depict two such materializations, for Act I and Act II). The crux of the approach is that each episode comes with (typically one, but sometimes more) queries in its background; therefore, each Act generates *SubTask*s, with each *Subtask* carrying and being responsible for the execution of a query that gathers the data (that are ultimately visualized in the main part of the slide). An *Episode* can have several *SubTask*s to compute its contents. Since each *SubTask* carries its own query depending on the *Act*/*Task*, the above mechanism is extensible by appropriately constructing the method *generateSubTasks()* for each materialization of *Act*.

Moreover, the determination of key findings, or *Highlight*s within each *Episode* is performed by the homonymous class. We fundamentally consider the presentation of results as a 2D matrix on the screen[2]; to this end, we have structured several methods that scan a 2D matrix and isolate interesting cells (top-k max or top-k min values, domination of a class of values by a column or row, etc). The class *Highlight* is a point of extensibility where methods for result extraction can be added to search for more results within the answer of a query.

For more information on the internal structuring of CineCubes, we refer the interested reader to Section 4, where we discuss the software architecture as well as the two aforementioned *extensibility mechanisms* in more detail. Before that, however, our next step is to present the essence of our method along with its formal foundations.

3. Foundations and Method Internals

In this section, we start with a short description of the model for cubes and cube queries and then we move on to describe (a) acts, as the means for collecting data via complementary queries and (b) highlights as the means for automatically detecting some important findings within query results and the means for text construction. We also provide the basic steps of our method for the creation of CineCube movies.

3.1. Formal Background

We base our approach on an OLAP model that involves (a) *dimensions* defined as lattices of dimension *levels*, (b) *ancestor functions*, mapping values between related levels of a dimension, (c) *detailed data sets*, practically modeling fact tables at the lowest granule of information for all their dimensions, and (d) *cubes*, defined as aggregations over detailed data sets. We follow the logical cube model of [8], accurately summarized in [11], which we provide here too for completeness.

Domains. We assume four countable pairwise disjoint infinite sets exist: a set of *level names* (or simply *levels*) $\mathcal{U}_\mathcal{L}$, a set of *measure* names (or simply

[2]Of course, other forms of visualization can *accompany* the result; however, it is our conviction that *the actual data should definitely be part of the answer* [10].

measures) $\mathcal{U_M}$, a set of *dimension* names (or simply *dimensions*) $\mathcal{U_D}$ and a set of *cube* names (or simply *cubes*) $\mathcal{U_C}$. The set of *attributes* $\mathcal{U}$ is defined as $\mathcal{U} = \mathcal{U_L} \cup \mathcal{U_M}$. For each $A \in \mathcal{U_L}$, we define a countable totally ordered set $dom(A)$, the domain of A, which is isomorphic to the integers. Similarly, for each $A \in \mathcal{U_M}$, we define an infinite set $dom(A)$, the domain of A, which is isomorphic to the real numbers. We can impose the usual comparison operators to all the values participating to totally ordered domains $\{ <, >, \leq, \geq \}$.

Dimensions and levels.A *dimension* D is a lattice ($\mathbf{L}$,$\prec$) such that:

- $\mathbf{L} = \{L_1,\ldots,L_n\}$, is a finite subset of $\mathcal{U_L}$.
- $dom(L_i) \cap dom(L_j)= \emptyset$ for every $i \neq j$.
- $\prec$ is a partial order defined among the levels of $\mathbf{L}$.
- The highest level of the hierarchy is the level $D.ALL$ with a domain of a single value, namely '$D.all$'.

Each path in the dimension lattice, beginning from its upper bound and ending in its lower bound is called a *dimension path*.

A family of functions $anc_{L_1}^{L_2}$ is defined, satisfying the following conditions:

1. For each pair of levels L_1 and L_2 such that $L_1 \prec L_2$, the function $anc_{L_1}^{L_2}$ maps each element of $dom(L_1)$ to an element of $dom(L_2)$.
2. Given levels L_1, L_2 and L_3 such that $L_1 \prec L_2 \prec L_3$, the function $anc_{L_1}^{L_3}$ equals to the composition $anc_{L_1}^{L_2} \circ anc_{L_2}^{L_3}$. This implies that:
   - $anc_{L_1}^{L_1}(x) = x$.
   - if $y = anc_{L_1}^{L_2}(x)$ and $z = anc_{L_2}^{L_3}(y)$, then $z = anc_{L_1}^{L_3}(x)$.
   - for each pair of levels L_1 and L_2 such that $L_1 \prec L_2$, the function $anc_{L_1}^{L_2}$ is monotone (preserves the ordering of values). In other words: $\forall\ x,y \in dom(L_1)$: $x < y \Rightarrow anc_{L_1}^{L_2}(x) \leq anc_{L_1}^{L_2}(y)$, $L_1 \prec L_2$

Schemata and data sets. A *schema* $\mathbf{S}$ is a finite subset of $\mathcal{U}$. Normally, we will represent a schema as divided in two parts: $\mathbf{S} = [D_1.L_1, \ldots, D_n.L_n, A_1, \ldots, A_m]$, where:

- $\{L_1,\ldots,L_n\}$ are levels from a dimension set $\mathbf{D} = \{D_1,\ldots, D_n\}$ and level L_i comes from dimension D_i, for $1 \leq i \leq n$.
- $\{A_1,\ldots, A_m\}$ are attributes, i.e. measures and levels.

A *detailed schema* $\mathbf{S}^0$ is a schema whose levels are the lowest in the respective dimensions. When we refer to a level L as the *lowest* in the dimension, it means that there does not exist any other level L', such that $L' \prec L$.

A *tuple* t over a schema $\mathbf{S} = [D_1.L_1, \ldots, D_n.L_n, A_1, \ldots, A_m]$ is a total and injective mapping from $\mathbf{S}$ to $dom(L_1) \times \ldots \times dom(L_n) \times dom(A_1) \times \ldots \times dom(A_m)$, such that $t[X] \in dom(X)$ for each $X \in \mathbf{S}$.

A *data set* **DS** over a schema $\mathbf{S} = [D_1.L_1, \ldots, D_n.L_n, A_1, \ldots, A_m]$ is a finite set of tuples over S such that:

- $\forall\ t_1,\ t_2 \in \mathbf{DS},\ t_1[L_1, \ldots, L_n] = t_2[L_1,\ \ldots,\ L_n] \Rightarrow t_1 = t_2$.
- for no strict subset $X \subset \{L_1, \ldots, L_n\}$, the previous also holds.

In other words, $A_1,\ \ldots,\ A_m$ are functionally dependent (in the relational sense) on levels $\{L_1, \ldots ,L_n\}$ of schema $\mathbf{S}$. A *detailed data set* $\mathbf{DS}^0$ is a data set over a detailed schema $\mathbf{S}^0$.

A *star schema* ($\mathbf{D}$,$\mathbf{S}^0$) is a couple comprising a finite set of dimensions $\mathbf{D}$ and a detailed schema $\mathbf{S}^0$) defined over (a subset of) these dimensions.

Selection filters. An *atom* is *true*, $false$, (with obvious semantics) or an expression of the form $x\ \theta\ y$, where x and y can be one of the following: (a) a level L_1 (i.e., not a measure); (b) a value l; (c) an expression of the form $anc_{L_1}^{L_2}(L_1)$, where $L_1 \prec L_2$; (d) an expression of the form $anc_{L_1}^{L_2}(l)$, where $L_1 \prec L_2$ and $l \in dom(L_1)$. θ is an operator from the set $\{>, <, =, \geq, \leq, \neq\}$.

A *selection condition* ϕ is a formula involving atoms and the logical connectives $\wedge$, $\vee$ and $\neg$.A selection condition is always applied to a data set such that all the level names occurring in the selection condition – either in the form (a) or (c) – belong to the schema of the data set. Let $\mathbf{DS}$ be a data set over schema $\mathbf{S}$. The expression $\phi(\mathbf{DS})$ is a set of tuples $\mathbf{X}$ belonging to $\mathbf{DS}$ such that when, for all the occurrences of level names in ϕ, we substitute the respective level values of every $x \in \mathbf{X}$, the formula ϕ becomes true. A *detailed selection condition* ϕ^0 is a selection condition where all participating levels are the detailed levels of their dimensions.

Cube queries. The user can submit *cube queries* to the system. A cube query specifies (a) the (basic) cube over which it is imposed, (b) the selection condition that isolates the records that qualify for further processing, (c) the aggregator levels, that determine the level of coarseness for the result, and (d) a list of aggregations over the measures of the underlying cube that accompany the aggregator levels in the final result. More formally, a *primary cube* c (over the schema $[L_1,\ \ldots,\ L_n,\ M_1,\ \ldots,\ M_m]$), is an expression of the form:

$$c = (\mathbf{DS}^0,\ \phi,\ [L_1,\ \ldots,\ L_n,\ M_1,\ \ldots,\ M_m],\ [agg_1(M_1^0),\ \ldots,\ agg_m(M_m^0)]),$$

where:

- $\mathbf{DS}^0$ is a detailed data set over the schema $\mathbf{S} = [L_1^0,\ \ldots,\ L_n^0,\ M_1^0,\ \ldots ,M_k^0]$, $m \leq k$.
- ϕ is a detailed selection condition.
- $M_1, \ldots ,M_m$ are measures.
- L_i^0 and L_i are levels such that $L_i^0 \prec L_i$, $1 \leq i \leq n$.
- $agg_i \in \{sum, min, max, count, avg\}$, $1 \leq i \leq m$.

The semantics of a primary cube in terms of SQL over a star schema are:

SELECT $L_1, \ldots, L_n$, $agg_1(M_1^0), \ldots,$ $agg_1(M_m^0)$

```
FROM DS^0 INNER JOIN D_1 ...INNER JOIN D_n
WHERE $\phi$
GROUP BY $L_1$,...,$L_n$
```

We also make the following assumptions for the query class of the supported cube queries:

- We work with cube queries that involve a single measure.
- We assume strictly two aggregator levels for the result; this allows a straightforward tabular representation of the result in a 2D screen.
- We assume that the selection condition is defined as the conjunction of a set of atomic formulae, *one per dimension*, each of which is of the form $L = v$, with L being a dimension level and v being a valid value for this level.

In the rest of our deliberations, we will assume that the users submit to the system cube queries that we denote as:

$$q = (\mathbf{DS}^0,\, \phi_1 \wedge \ldots \wedge \phi_k,\, [L_\alpha,\, L_\beta],\, agg(M))$$

A note on data presentation is due at this point. Although there are several ways that we can employ to visualize results, like for example scatter plots on a 2D space or bar charts with multiple data series, we would like to stress once again that any such visualization methods are *complementary* to the actual data [10]. So, in the rest of our deliberations, we treat the results of a cube query of the above form as being visualized in tabular format with the values of L_α as rows and the values of L_β as columns. Expanding the method for more than two dimensions (via the typical nesting of dimensions in rows and columns) is part of future work.

3.2. Act I: Putting Things in Context – or "How good is the original cube compared to its siblings?"

In this subsection, we deal with the first of the acts. The main purpose of the first act is to provide a context for the original query. So, we compare the marginal aggregate results of the original query to the results of "sibling" queries that use "similar" values in their selection conditions (to be explained right next).

Method. We assume an original query and we want to compare its results with similar queries. We define a sibling query as a query with a single difference to the original: instead of an atomic selection formula $L_i = v_i$, the sibling query contains a formula of the form $L_i \in children(parent(v_i))$.

Formally, given an original query

$$q = (\mathbf{DS}^0,\, \phi_1 \wedge \ldots \phi_x \wedge \ldots \wedge \phi_k,\, [L_\alpha,\, L_\beta],\, agg(M)),\, \phi_i\text{: } L_i = v_i,\, i = 1, \ldots, k$$

a new query q^s is a *sibling query* if it is of the form

$$q^s = (\mathbf{DS}^0,\ \phi_1 \wedge \ldots\ \phi^*_{\mathbf{x}} \wedge \ldots \wedge \phi_k,\ [L_\alpha,\ L_\beta],\ agg(M)),\ \phi_i\text{: } L_i = v_i,\ i\text{=}1,\ldots,\ x\text{-}1,\ x\text{+}1,\ \ldots,\ k,\ \phi^*_x\text{: } L_{x+1} = anc^{L_{x+1}}_{L_x}(v)$$

Naturally, if q originally has k atomic selections, it also has k sibling queries.

To compare the results of the original query to the ones of its siblings, one would need to lay out all the k sibling queries on the same screen and visually inspect their differences. This becomes too hard to exploit as k increases – in fact, even with a very small k (e.g., k=2) it can be too hard to be able to visually compare the results. So we, need to resort to auxiliary comparisons that provide the context needed. To this end, we introduce two *marginal sibling queries*, one for each aggregator. Each time, we keep one of the two aggregators, and the other becomes L_x. If we combine this with the fact that the new selection condition ϕ^*_x restricts L_x to the siblings of the original value v, then the resulting 2D matrix has one of the original aggregators in one of its two dimensions and the siblings of v on the other. This way, the marginal values of the original query on one of the two aggregators are compared to the respective marginal values of the siblings.

Formally, given an original query

$$q = (\mathbf{DS}^0,\ \phi_1 \wedge \ldots\ \phi_x \wedge \ldots \wedge \phi_k,\ [L_\alpha,\ L_\beta],\ agg(M)),\ \phi_i\text{: } L_i = v_i,\ i = 1,\ \ldots,\ k$$

its two *marginal sibling queries* are

$$q^s_\alpha = (\mathbf{DS}^0,\ \phi_1 \wedge \ldots\ \phi^*_{\mathbf{x}} \wedge \ldots \wedge \phi_k,\ [L_\alpha,\ L_x],\ agg(M)),\ \phi_i\text{: } L_i = v_i,\ i = 1,\ \ldots,\ x\text{-}1,\ x\text{+}1,\ \ldots,\ k,\ \phi^*_x\text{: } L_{x+1} = anc^{L_{x+1}}_{L_x}(v)$$

$$q^s_\beta = (\mathbf{DS}^0,\ \phi_1 \wedge \ldots\ \phi^*_{\mathbf{x}} \wedge \ldots \wedge \phi_k,\ [L_x,\ L_\beta],\ agg(M)),\ \phi_i\text{: } L_i = v_i,\ i = 1,\ \ldots,\ x\text{-}1,\ x\text{+}1,\ \ldots,\ k,\ \phi^*_x\text{: } L_{x+1} = anc^{L_{x+1}}_{L_x}(v)$$

Example. The original query is expressed as:

$$q = (\mathbf{DS}^0, W.L_2 = \text{'With-Pay'} \wedge E.L_3 = \text{'Post-Sec'},\ [W.L_1,\ E.L_2],\ avg(Hrs))$$

In the reference example, slides ❷ and ❸ involve the two marginal subqueries – see for example the former with the selection set to parent('With-Pay') and the grouping to the level of 'With-Pay'(i.e., L_3):

$$q^2 = (\mathbf{DS}^0,\ W.L_2 = \text{'With-Pay'} \wedge E.L_4 = \text{'ALL'},\ [W.L_1,\ E.L_3],\ avg(Hrs))$$

3.3. *Act II: Explaining Variation – or "Drilling into the breakdown of the original result"*

The purpose of Act II is to help the user understand why the situation is as observed in the original query. In order to shed some more light to what is happening, we drill in the details of the cells of the original result in order to inspect the internals of the aggregated measures of the original query.

Assume a cube query

$$q = (\mathbf{DS}^0,\ \phi_1 \wedge \ldots \wedge \phi_k,\ [L_\alpha,\ L_\beta],\ agg(M)),\ \phi_i\colon\ L_i = v_i,\ i = 1, \ldots, k$$

and its result, visualized as a 2D matrix. Then, each cell c of this result is characterized by the following cube query:

$$q^c = (\mathbf{DS}^0,\ \phi_1 \wedge \ldots \wedge \phi_k \wedge \phi_{\mathbf{c}},\ [L_\alpha,\ L_\beta],\ agg(M)),\ \phi_i : L_i = v_i,\ i = 1, \ldots,\text{k},$$
$$\phi_c\colon\ \phi^c_\alpha \wedge \phi^c_\beta = (L_\alpha = v^c_\alpha \wedge L_\beta = v^c_\beta)$$

For each of the aggregator dimensions, we can generate a set of *explanatory drill in queries*, one per value in the original result:

$$q^{\alpha_i} = (\mathbf{DS}^0,\ \phi_1 \wedge \ldots \wedge \phi_k \wedge \phi^{\alpha_i},\ [L_{\alpha-1},\ L_\beta],\ agg(M)),$$
$$q^{\beta_i} = (\mathbf{DS}^0,\ \phi_1 \wedge \ldots \wedge \phi_k \wedge \phi^{\beta_i},\ [L_\alpha,\ L_{\beta-1}],\ agg(M)),$$

with α_i and β_i being all the values of the original result for the grouper levels. So, each of the two slides has a set of such queries.

Example. Observe slide ❹ where we drill-down for values Gov, Private and Self-emp via the explanatory drill in queries for dimension *Work*.
$q^{gov} = (\mathbf{DS}^0,\ W.L_2 =$ 'With-Pay' $\wedge\ W.L_1 =$ 'Gov' $\wedge\ E.L_3 =$ 'Post-Sec', $[W.L_0,\ E.L_2],\ avg(Hrs))$

$q^{prv} = (\mathbf{DS}^0,\ W.L_2 =$ 'With-Pay' $\wedge\ W.L_1 =$ 'Private' $\wedge\ E.L_3 =$ 'Post-Sec', $[W.L_0,\ E.L_2],\ avg(Hrs))$

$q^{s-e} = (\mathbf{DS}^0,\ W.L_2 =$ 'With-Pay' $\wedge\ W.L_1 =$'s-e' $\wedge\ E.L_3 =$ 'Post-Sec', $[W.L_0,\ E.L_2],\ avg(Hrs))$

Observe that due to the fact that this is the special case where selection conditions involve grouper values at finer levels of detail, we have completely removed the atomic formula of the dimension that we drill-down ($W.L_2 =$ 'With-Pay').

3.4. Highlights and Text

As already mentioned, the extraction of highlights is orthogonal to the query that creates the results of a slide. Once the results of the query are computed and organized in a 2D matrix, we utilize a palette of highlight extraction methods that take a 2D matrix as input and produce important findings as output. This way, (a) we can reuse highlight extraction methods to all the query results, independently of the Act or the query that has been executed, and, (b) we can gracefully extend the palette of highlight extraction methods with more results. We have implemented a small number of highlight extraction methods for the moment that include the highlighting of the top and bottom quartile of values in a matrix, the absence of values from a row or column, the domination of a quartile by a row or a column (i.e., when all the values of a quartile appear in a certain row or column), the identification of min and max values, etc. Clearly, there is a vast area of enriching this palette (trend analysis, correlations, relative relationships of rows and columns, to name just a few); however, implementing the full spectrum of such techniques can be done with diligence as part of future work.

Text is constructed by a Text Manager that customizes the text per Act, by plugging values to a template that comes with each act. Compare the following excerpt with the text of slide ❹ in Fig. 1.

In this slide, we drill-down one level for all values of dimension `<dim>` *at level* `<l>`. *For each cell we show both the* `<agg>` *of* `<measure>` *and the number of tuples that correspond to it ...*

3.5. Creation of CineCubes

Having explained all the individual steps, we now move on to discuss the overall process for creating a CineCube movie. In its current configuration, a CineCube movie includes three kinds of acts: the *Introductory Act* (including the introductory slide), three *Operational Acts* including the act involving the original query and the two acts for the management of complementary queries, and a *Summary Act* with a summary slide with all the important highlights of the previous three acts.

Overall the method includes the following steps:

1. Construct Introductory Act
2. For all the Operational Acts, execute the *Construct Operational Act* algorithm that calculates the Act's contents (result visualization, highlights, text and audio)
3. Construct Summary Act in the end
4. Wrap-up the Acts in a PowerPoint movie

```
Algorithm Construct Operational Act
Input: the original query over the appropriate database
Output: a set of an Act's episodes fully computed

  1. Create the necessary objects (act, episodes, tasks, subtasks) appropriately
     linked to each other
  2. Construct the necessary queries for all the subtasks of the Act, execute
     them, and organize the result as a set of aggregated cells (each including its
     coordinates, its measure and the number of its generating detailed tuples)
  3. For each episode
     (a) Calculate the visual presentation of cells
     (b) Calculate the cells' highlights
     (c) Produce the text based on the highlights
     (d) Produce the audio based on the text
```

Figure 5: Constructing an Operational Act

The first step of the method, the construction of the *Introductory Act*, is trivial. The second step, the computation of the contents and presentation of the *Operational Acts* is outlined in the Algorithm of Fig. 5. The third step, which involves the construction of the *Summary Act*, is simply a slide with the text of the highlights copied to it, organized per act. Finally, the fourth step, *wrapping-up* the individual acts in a single report, introduces a few programmatic tasks worth mentioning here. In a nutshell, for every episode, we create a slide, with its title and contents (i.e., the 2D tables or the text, depending on the type of slide). This can be done straightforwardly with the programming facilities provided by the Apache POI. Unfortunately, though, POI does not support the management of notes, where we actually store the text of each slide, and audio. This is done by exploiting the open nature of pptx documents (see section 4.1).

4. Software Architecture and the Pledge to Extensibility

In this Section, we start with a description of the employed technologies that allow the programmatic construction of CineCube movies; we believe this description can help the interested reader to probe further in the employed API's that come with a quite low technological barrier for the new-comer. Second, in this Section, we describe the software architecture of CineCubes with the main goal of highlighting the different ways in which CineCubes is extensible. Extensibility is a key feature that should not be underestimated by no means, as it can allow a system to expand in diverse ways. In fact, in this case, the extensibility-oriented architecture of CineCubes is the backbone for all the diverse ways in which future research can be pursued. Readers who are not interested in the software-related aspects of our method may prefer to jump directly to Section 5 for the experimental assessment of our method; this can be done without a gap in the understandability of the text.

4.1. Employed Technologies

One of the major goals of this paper is to highlight how we can automatically construct a CineCube presentation that includes result visualization, text and audio. So, before elaborating further in the software architecture of CineCubes, it is worth discussing the programmatic simplicity[3] of our method. In this subsection, we explain the main technologies via which our PowerPoint presentations are programmatically constructed.

Apache POI [3] is a Java API that provides several libraries to create and modify Microsoft Word, PowerPoint and Excel files. MS Office files obey the Office Open XML standards (OOXML) and Microsoft's OLE 2 Compound Document format (OLE2). More specifically, XSLF is the Java implementation of the PowerPoint 2007 OOXML (.pptx) file format in POI.

The automatic manipulation of .pptx files is relatively simple for simple tasks. See the following excerpt for creating a file and a slide:

[3]We would insist that simplicity is a strong feature of any method, making it scalable, extensible and maintainable and fundamentally different to naiveness or superficiality.

```
XMLSlideShow ss = new XMLSlideShow();
XSLFSlideMaster sm = ss.getSlideMasters()[0];
XSLFSlide sl= ss.createSlide
  (sm.getLayout(SlideLayout.TITLE_AND_CONTENT));
XSLFTable t = sl.createTable();
t.addRow().addCell().setText("added a cell"); ...
```

Also, we automate the construction of text that characterizes each slide. We add the text for each slide that we create as a slide's note. At the same time, the existence of text can help us create a narrative as audio. We use the API provided by MARY [4], which is an open-source, multilingual Text-to-Speech Synthesis (TTS) platform written in Java and allows to generate one audio file per slide, simply by providing the notes of the slide as input to a method call.

```
MaryInterface m = new LocalMaryInterface();
m.setVoice(‘‘cmu-slt-hsmm’’);
AudioInputStream audio = m.generateAudio("Hello’’);
AudioSystem.write(audio, audioFileFormat.Type.WAVE, new
    File("myWav.wav")); ...
```

It is worth mentioning here, that unfortunately, the current version of POI does not support the management of notes, where we actually store the text of each slide, and audio. Fortunately, the open nature of MS Offuce allows us to exploit its internal structure. It is noteworthy that each MS Office file is actually a zipped folder with a rigid structure, within which, XML and media files are located in a principled fashion. So, to deliver a presentation in the form that we wish to have it, we proceed as follows (via the appropriate *WrapUp Manager*): (i) we unzip the pptx in a temporary folder, (ii) create appropriate files for the notes in the ppt/notes/ folder, along with the necessary links that link them to their slide, (iii) do the same for audio at the ppt/media folder, and (iv) zip the folder again to a .pptx file.

Overall, despite the existence of several nuts and bolts that need fine tuning, *the main lesson learned here is that the packaging of the results of our method, one by one as slides in a presentation is attainable with neat programming facilities, already available in the Web.*

4.2. Architecture under the prism of extensibility

The architecture of CineCubes is organized in packages, each dedicated in a specific task. In the rest of this section, we present the functionality and the points of extensibility of these packages. For the moment, we start by giving a short overview of the package structure of CineCubes architecture (see Fig. 6 for a diagrammatical representation that also includes the dependencies between packages).

- The package *CubeMgr*, consists of two subpackages as shown in Fig. 6, which are:

  - *CubeBase*, which includes the classes that we use to represent the cube model.
  - *StarSchema*, which includes the classes that we use to capture the star schema of an underlying relational database that hosts all the data used to answer our queries.

- The package *TaskMgr* includes the classes which we use in our algorithm *Construct Operational Act* for constructing the necessary *Task* and *Subtask* objects.
- The package *StoryMgr* contains the classes which we use to construct the main objects of a *Story.*
- The package *HighlightMgr* contains the classes needed to construct the different highlights for each episode of a *Story.*
- The package *TextMgr* includes the classes which construct the text for each episode of a *Story.*
- The package *AudioMgr* includes the classes which convert the text to audio.
- The package *WrapUpMgr* include the classes which create the final result for the user.

In the following, we provide more information for the classes of the above packages.

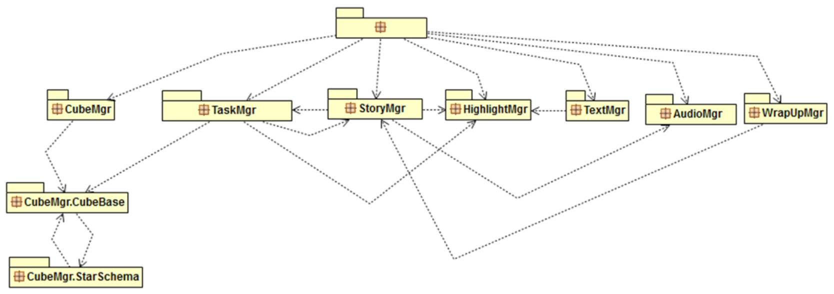


Figure 6: Structure of CineCubes' Packages

The package CubeMgr. As already mentioned, the package *CubeMgr* contains two subpackages, *CubeBase* and *StarSchema* that are responsible for representing the cube model, as an object model in main-memory and the star-schema model which is the mapping of the cube model in relational terms (and provides the metadata for translating cube queries to SQL queries). All the classes for the cube model (*Dimension*, *Cube*, *Hierarchy*, *Level*, *Measure* and *CubeQuery* among others) are part of the former subpackage, whereas classes like *FactTable*, *DimensionTable*, *Table*, *Attribute* and *SQLQuery* are part of the latter. Notably, our implementation has conveniently been based on cube queries that can be automatically translated to SQL queries due to the simplicity of the star schema. These SQL queries are the ones that are ultimately executed in order to collect the results.

The package TaskMgr. The package *TaskMgr* contains the necessary classes which help us to create a new kind of *Act*. Here we have a *TaskMgr* class to manage the tasks. *A first point of extensibility of our method involves the creation of tasks*: the *Task* class is abstract to facilitate the creation of a different type of task for each new kind of *Act v*ia the appropriate materialization – e.g., in our implementation we have created two subclasses for *Act I* and *Act II* and one subclass to implement the original request. Another point of extensibility, with an eye to future work, is located in the abstract class *ExtractionMethod* which is used in order to provide us with results. Currently, we materialize this class via *SqlQuery* to get the result from a relational database; however, in future we can materialize it differently to get data from different sources e.g., XML or $SPARQL$ files. Class *Result* keeps the query result in a 2D matrix and implements a set of functions to manipulate this matrix.

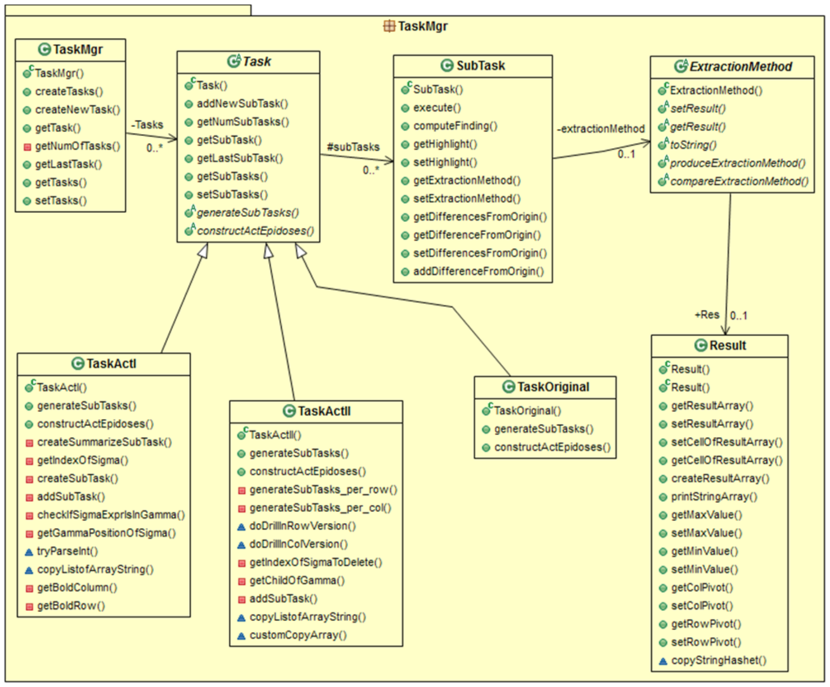


Figure 7: Class Diagram for Package TaskMgr

The package StoryMgr. In the package *StoryMgr*, we host the main classes needed to create a *Story*. This package has the class *StoryMgr* to manage the story and the *Story* class. Also, it has the classes which implement the acts, the episodes of each act and the visualization of an episode.

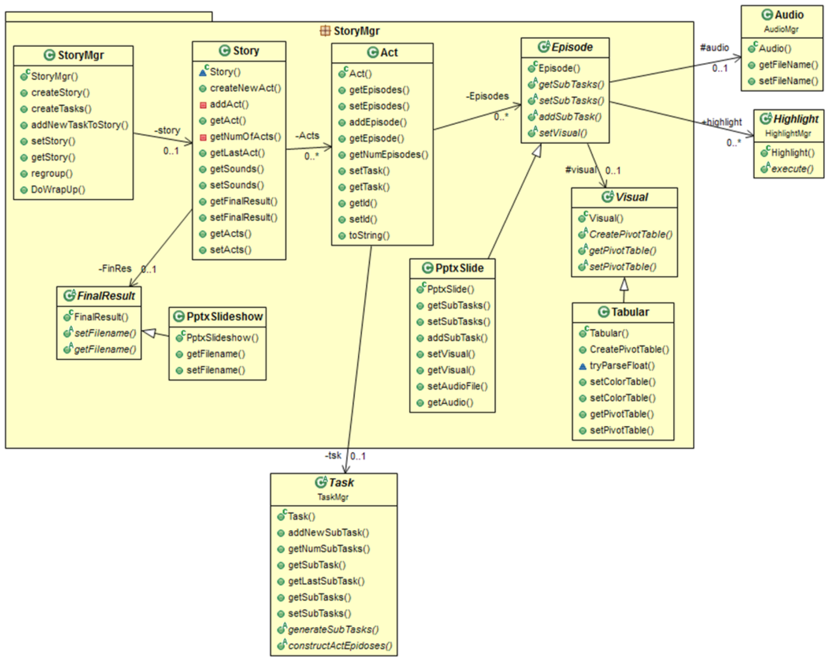


Figure 8: Class Diagram for Package StoryMgr

There are several points of extensibility here:

- The implementation of the episodes of *Act* is performed via an abstract class *Episode,* which in our approach is materialized as *PptxSlide*, although in the future one can materialize it in other ways too, to create different types of episodes (e.g., frames in *.wmv* file).
- In a similar fashion, the *Story* class relates to the abstract class *FinalResult*, currently materialized as *PptxSlideshow* in our method and also open to extensions (such as *.wmv* file), in the future.
- The *Visual* class is an abstract class, currently materialized by the *Tabular* class, which visualizes the result as a pivot table; again, in the future we can create new kinds of visualization (such as a charts of all kinds).

Episodes are key for managing results: observe how the *Episode* class is associated with the *Highlight*, the *Audio*, the *Visual* class and the *Subtask* class (of package *TaskMrg*, not depicted here) and glues all these together.

The package HighlightMgr. One of the strongest points of extensibility and key to the operation of CineCubes is the enrichment of each slide with the possibility of extracting highlights from the data that it hosts. The package *HighlightMgr* hosts an abstract class *Highlight*, which is open to extension for finding highlight in episodes. In our current implementation, we have created the

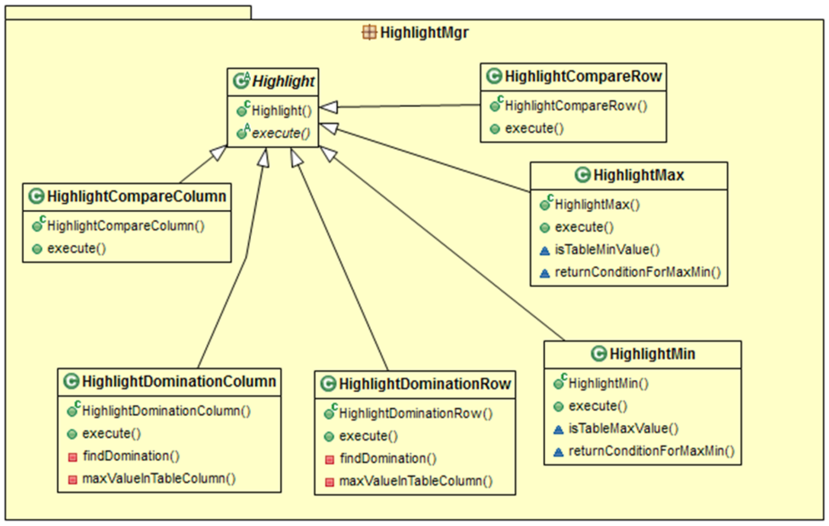


Figure 9: Class Diagram for Package HighlightMgr

six following subclasses, all of which implement a method *execute()* that takes a 2D matrix of values as input and creates lists of values where the findings are stored:

- *HighlightCompareRow*, to compare one row with the other rows
- *HighlightCompareColumn*, to compare one column with the other columns
- *HighlightMax*, to find the top quartile of the values in a matrix
- *HighlightMin*, to find the bottom quartile of the values in a matrix
- *HighlightDominationRow*, to test the domination of a quartile (top or bottom) by a row
- *HighlightDominationColumn*, to test the domination of a quartile (top or bottom) by a column

We utilize a dedicated *Highlight Manager* class to extract Highlights.

Text and Audio packages. Each of these two packages implements its homonymous functionality. Note that for the case of audio, we have experimented with more than one TTS systems (MaryTTS and FreeTTS) and thus, it is already implemented in an extensible way via the materialization of an abstract class by a concrete class, one per TTS system.

The package WrapUpMgr. The package *WrapUpMgr* serves the purpose of packaging all the different parts computed by the previous packages in a single, final result delivered to the user. Again, this package is constructed in an extensible way as it contains the abstract class *WrapUpMgr* which, then, has to be materialized by a subclass in order to construct the proper format for a story. Currently, we create the subclass *PPTXWrapUpMgr* which returns to the user a Microsoft PowerPoint presentation, although one can think of several ways of representing stories (as Word documents, as *.wmv* files, and so on) in the future.

4.3. Extending the set of Acts

In this subsection, we present the sequence of steps needed in order to extend the system with a new *Act*, along with its constituents. We will use the existing acts that we have already implemented as reference cases for this discussion. To create a new act for our current method we must implemented **one** new class which materializes the class *Task*. Moreover, the new class must implement the **two** abstract methods of class *Task*: (a) the *generateSubTask()* and (b) *conctructActEpisodes()*. Also, we must add a new method in class *TextExtractionPPTX* such that to extract the proper contextual description added at each slide of new act. For example, for Act I of our approach we materialized the class *TaskActI* which implements the two aforementioned abstract methods along with the method *createTextForAct1()*. Similarly, for Act II, we materialized the class *TaskActII* which implements the two abstract methods and the method *createTextForAct2()*.

4.4. Extending the set of Highlight Extraction Methods

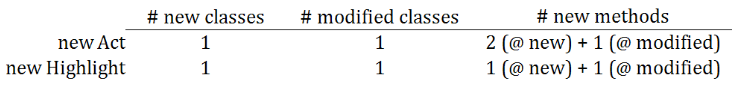
To have the ability to create different highlights we create an abstract class *Highlight* which has an abstract method with name *execute()*. In our current implementation, we have created six subclasses which help us to create the different highlights for our episodes. In Fig. 9, we can observe that all the subclasses of *Hightlight* implement the abstract function *execute()*. In addition, every time we want to add a new kind of *Highlight* we must add a new method in class *TextExtractionPPTX* such that to extract the proper text for new highlight. We conclude that in order to enter a new highlight we must create a new class (which materializes the *Highlight* class), to implement the abstract method *execute()*, and to add a new method to class *TextExtractionPPTX.*

4.5. Assessing the Extensibility of our framework

In Fig. 10, we present the programming effort which was needed in order to extend the current approach of our method for adding Act II. As already mentioned, adding a new kind of Act requires to create **one** new class and to implement **three** methods. Moreover, in order to create a new kind of highlight, we must create **one** new class and implement **two** methods. We believe that in summary, the programming effort to extend our method in each flavor of extensibility is too low.

5. Experiments

In this section, we describe our experimental assessment of the CinceCubes tool.



	# new classes	# modified classes	# new methods
new Act	1	1	2 (@ new) + 1 (@ modified)
new Highlight	1	1	1 (@ new) + 1 (@ modified)

Figure 10: Assessment of the Extensibility Effort for CineCubes

5.1. Experimental Setup

For all our experiments, we have experimented with the Adult (a.k.a census income) dataset referring to data from 1994 USA census. The dataset in its cleansed version (after uncertain and NULL values are removed) comprises 30162 tuples of the 1994 USA census. There are 8 dimensions (Age, $Native\,Country$, $Education$, $Occupation$, $Marital\ status$, $Work\ class$, $Gender$, and $Race$) in the data set and a single measure, $Hours\ per\ Week$. The hierarchies for the dimensions $Education$ and $Work\ class$ are depicted in Fig. 2. The hierarchies for the dimensions $Occupation$, $Marital\ status$, $Gender$, and $Race$ are depicted in Fig. 11 and the hierarchy of dimension $Native\ Country$, except the level 0 (which includes too many values), is depicted in Fig. 12. The dimension Age is organized in years, 5-year intervals, 10-years intervals, 20-year intervals and *.

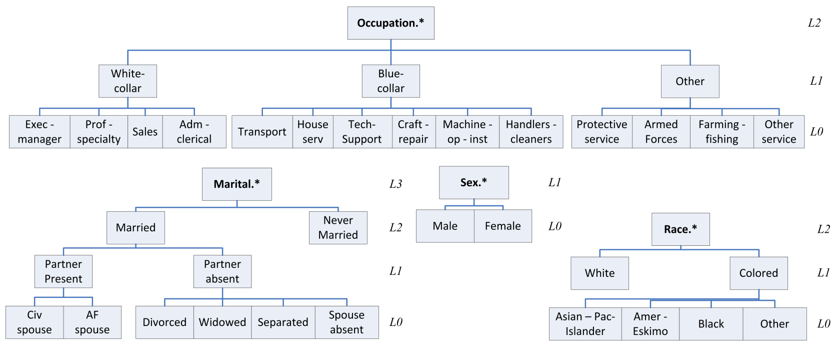


Figure 11: Dimensions Occupation, Marital Status and Race

In terms of efficiency, what we are interested to discover is where we spend more time during the generation of a CineCubes movie. As the generation of the pptx file advanced, we have carefully monitored all the individual steps of the method. Therefore, we are able to discuss the time costs of the method from two points of view: (i) concerning the individual parts of the method (results / highlight / text and audio generation etc) and (ii) concerning the different acts of the method. To provide a thorough evaluation, we have worked with a variant number of two, three, four or five atomic selection conditions in the WHERE clause of the original query. This results in an increase in the number of slides in Act I as the number of atomic selections in the WHERE clause increases: since Act I compares with sibling values of the selection values, each selection condition adds two extra slides, where the siblings of the involved value

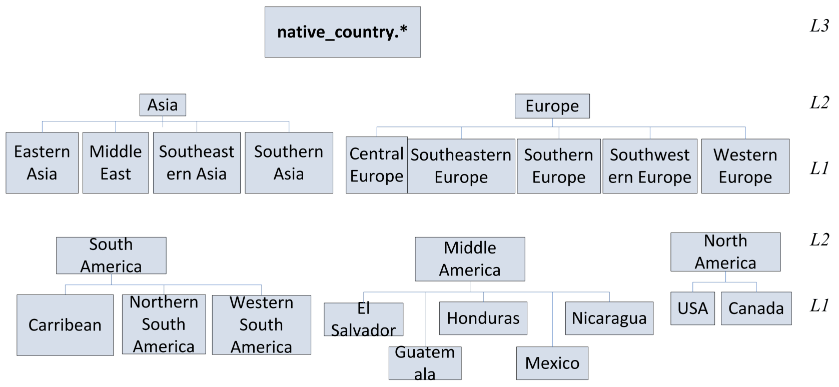
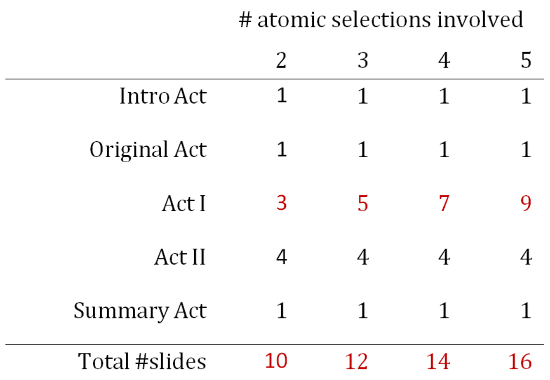


Figure 12: Dimension Native Country (the most detailed level L_0 is not depicted due to its large number of instances



	# atomic selections involved			
	2	3	4	5
Intro Act	1	1	1	1
Original Act	1	1	1	1
Act I	3	5	7	9
Act II	4	4	4	4
Summary Act	1	1	1	1
Total #slides	10	12	14	16

Figure 13: Number of slides for different numbers of atomic selection conditions

are grouped for the two groupers (Fig. 13). Instead, the slides of Act II remain in all cases constant, which is consistent to the essence of the method, which drills down the grouping levels of the original query. All experiments have taken place in a conventional PC running Windows 7 over an Intel Core Duo CPU at 2.50GHz, and with 3GB main memory.

5.2. Analysis of Results per Part of the Method

Our first experimental goal has been to assess the amount of time taken by each of the parts of our method. We have measured the time needed to perform each part of the method in milliseconds. The individual steps of the method have been grouped in 5 parts that are executed for each slide as follows:

I. Result Generation. Result generation involves the construction and execution of queries to the underlying database. Specifically, this part involves the following individual tasks:

- *Produce Cube Query*: in this step, we create a Cube Query from the original query.
- *Produce SQL Query*: in this step, we convert a Cube Query to SQL query.
- *Execute SQL Query*: in this step, we perform the query to the database and take the result back.

II. Highlight Generation & Visualization. This part involves the shaping of the presentation of the results, as well as the identification of important highlights.

- *Tabular Creation*: in this step, we format the result of query execution as a pivot table.
- *Highlight Creation*: in this step, we calculate the highlights over the pivot table (such as row domination, largest values etc)
- *Color Table Creation*: in this step, we add color to each cell of pivot table.
- *Combine Pivots in the SameSlide*: this action is performed only on Act II, as the slides of this act contain more than one subqueries (and therefore, pivot tables) which have to be combined in a single slide.

III. Text Creation: in this part, we produce the slide's text from the calculated highlights.

IV. Audio Creation: in this part, we pass the produced text to the text-to-speech conversion API, in order to create the audio file.

V. Put all in pptx: in this part, which is the only one that takes place for the entire presentation, we wrap up all the above to a slideshow presentation.

As already mentioned, our experimental method involves varying the number of atomic selection conditions within the WHERE clause. Remember that as the number of selection conditions rises, each time we have two extra slides at

Act I for every extra atomic selection condition. We depict all the results in Fig. 14 (the number of slides of each try is depicted in parentheses at the header of the table of values in Fig. 14).

Clearly, the audio generation dominates the entire process, being several orders of magnitude larger than anything else and presenting a clear case for improvement. As the number of slides slowly increases, the time needed to generate text slowly increases too. Concerning the rest of the parts of the process, we see that query generation and execution takes up two orders of magnitude more than the other two tasks; therefore, being prudent with the number of slides (and thus, executed queries) is also necessary – esp., if someone would decide to exclude audio generation from the process.

A very interesting observation is also that, so far, both text creation and highlight extraction are extremely fast, and thus, provide the potential for enrichment with more algorithms that try to find interesting highlights and create representative textual descriptions for them.

5.3. Analysis of Results per Act

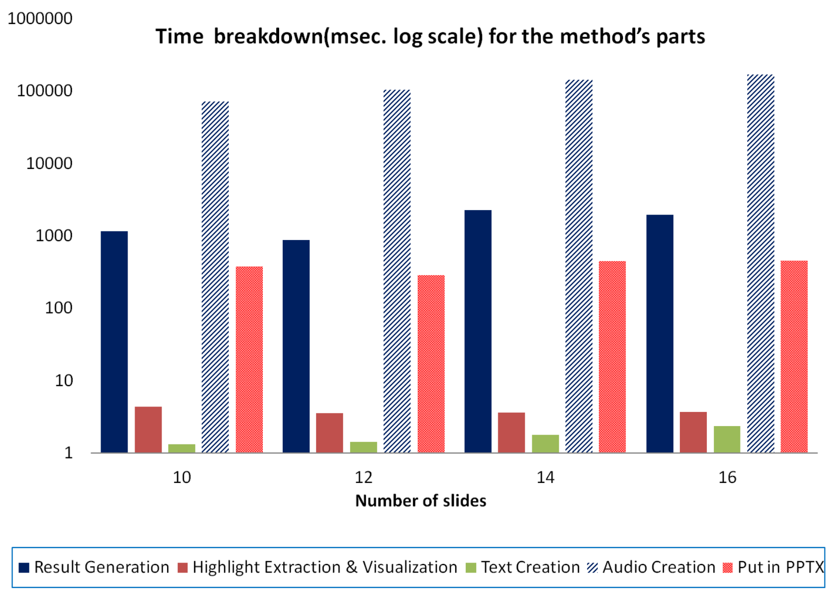
The second goal of our study was to find out how time is divided within the acts of the story. Again, we have measured the time needed to produce each Act of the story (measured in milliseconds). We depict our findings in Fig. 15.

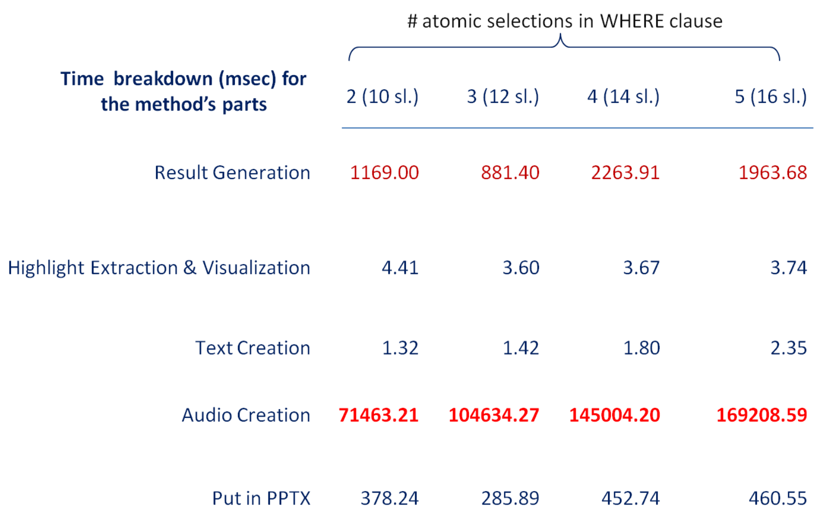
As the number of selection conditions rises, each time we have two extra slides at Act I (the number of slides of each try is depicted in parentheses at the header of the table in Fig. 15). Clearly, we can observe that the time of each Act is increasing as the number of atomic selection conditions increases. Moreover, the construction of Act I in three of the four cases takes more time than the construction of the others – only in the case when we have two atomic selection conditions, the construction of Act II takes about 90 msec more. In addition, the time needed to create Act II is practically stable, independently of the number of atomic selection conditions in the WHERE clause.

In Fig. 15, observe that as the number of slides increases (2 extra slides each time) Act I increases with significant rate; the Summary Act behaves similarly, albeit with a lower increase. Both these effects are mainly due to the text and audio generation (as already mentioned). The linearity of the increase for Act I can safely be attributed to the cost of the extra slides that are added each time to the Act (in fact, this is also corroborated by the detailed measurements per individual slide that are not included here).

Also in Fig. 15, we can observe that the Summary Act needed more time than the Act II in three of the four cases. This happens because the Summary Act, as described in Chapter 2, has all the highlights of the story (i.e., all the text for these highlights) which must be also converted to sound. Once again the text to speech API dominates the time of our result.

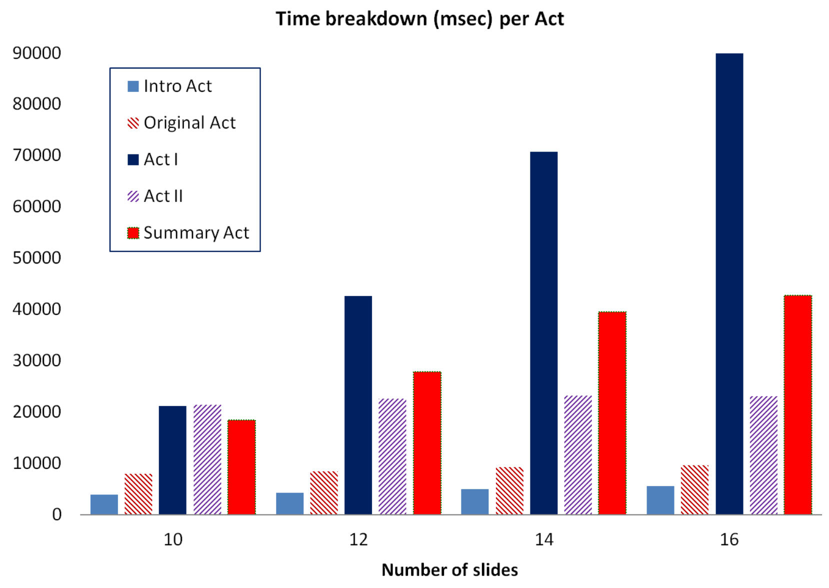
Why does the Summary Act increase with a smaller rate than Act I? This happens because the Summary Act has only the highlights of all episodes (and not the entire text). Due to the text that dresses and contextualizes the highlights in each slide, Act I has always more words to be converted to sound from





Time breakdown (msec) for the method's parts	# atomic selections in WHERE clause			
	2 (10 sl.)	3 (12 sl.)	4 (14 sl.)	5 (16 sl.)
Result Generation	1169.00	881.40	2263.91	1963.68
Highlight Extraction & Visualization	4.41	3.60	3.67	3.74
Text Creation	1.32	1.42	1.80	2.35
Audio Creation	**71463.21**	**104634.27**	**145004.20**	**169208.59**
Put in PPTX	378.24	285.89	452.74	460.55

Figure 14: Time breakdown (msec) for the method's parts



	# atomic selections in WHERE clause			
Time breakdown (msec) per Act	2 (10 sl.)	3 (12 sl.)	4 (14 sl.)	5 (16 sl.)
Intro Act	3746.99	4240.22	4919.71	5572.97
Original Act	7955.17	8425.59	9234.47	9577.76
Act I	**21160.78**	**42562.10**	**70653.22**	**90359.89**
Act II	21250.44	22419.34	22819.94	22738.88
Summary Act	18393.10	27806.35	39456.52	42750.78

Figure 15: Time breakdown (msec) for the method's acts

Summary Act (Table 1). It is noteworthy (observe Fig. 16) that the difference in number words between Act I and Summary Act, in each case, is linearly related to the extra time needed each time; the bond is extremely strong with a Pearson correlation of 0.999.

	# atomic selections in WHERE clause			
	2 (10 sl.)	3 (12 sl.)	4 (14 sl.)	5 (16 sl.)
Act I	244	499	764	1069
Summary Act	200	298	357	424

Table 1: Count of words for Act I and Summary Act

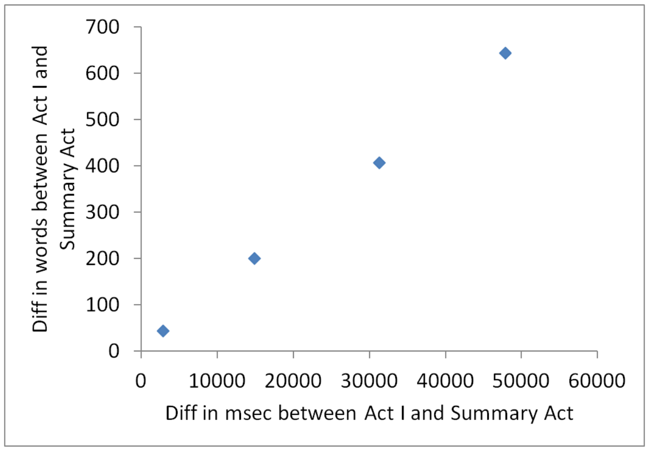


Figure 16: Difference in words and execution time for Act I and Summary Act

5.4. *Effectiveness assessment via a user study*

We have conducted a user study to verify the effectiveness of our approach and assess its benefits and shortcomings. In the sequel, we present the experimental setting and method, and then we move on to present our findings.

5.4.1. *Experimental method*

The epicenter of the effort was to assess the effectiveness of CinecCubes compared to simple querying in the presence of dimension hierarchies. To this end, we constructed a simple system answering aggregate queries in OLAP style to compare it against CineCubes. Both systems had the same user interface that allows users to construct queries by point-n-click without having to actually write them in SQL. The users that participated in the study were 12 PhD students from our Department, all of which were experienced in data management and statistics.

The user study consisted of four phases.

Phase 0 (warm-up). In the first phase, the users were familiarized with the data set and the tools. To this end, we presented the data set, its dimensions and levels. We also gave a demo of how to pose queries to the systems. We explained to the users that they could use any combination of (a) slideshow, (b) browsing through the slides, and (c) reading a printout of a query result or a CineCubes report. All users were given a pamphlet reminding the basics of the above. Then, the users returned to their offices, where they all had ample networking, computing and printing facilities to work with the next of the phases.

Phase 1 (simple OLAP functionality). The first part of the evaluation was to ask the users to prepare a report on a specified topic. The report should contain (a) a bullet list of key, highlight findings, (b) a text presenting the overall situation and, (c) optionally, any supporting statistical charts and figures to elucidate the case better.

Phase 2 (CineCubes functionality). We assigned the same task to the users, but now, they had CineCubes available. Both the simple querying system and CineCubes were at the disposal of the users in order to pose auxiliary requests for simple queries or CineCubes reports. To speed up the process, we also provided a link with a version of Cinecubes without audio.

Phase 3 (Questionnaire completion). Once the users had used the two systems, they were asked to complete a questionnaire, where they would comment on the usage of the two systems. The questionnaire prompted the users to complete information for the time needed to complete their reports, an assessment in a scale of 1 to 5 of the usefulness of the different acts of the CineCubes report, as well as of the textual parts and the voice features of CineCubes and an overall assessment of the two reports after having produced both of them.

5.4.2. Evaluation of Cinecube's parts

In this part of the questionnaire, the users were asked to provide an assessment of the usefulness of the parts of CineCubes in a scale of 1 to 5, with 1 being the worst value and 5 being the best. In Fig. 17 we depict the frequencies of the scores assigned by the users.

The figures reveal that the users appreciated differently the different acts and parts of the system – in fact, some of the findings have been surprising. All features scored an average higher than 3. The most popular feature was Act II, with the detailed, drill-down analysis of the groupers. The users attributed this to the fact that it provided them with information they thought interesting to include in the report, as it enlarged the picture of the situation that was presented to them. The second most popular feature was the treatment of the original query (that includes coloring, and highlight extraction compared to the simple query results given to them by the simple querying system).

It has been quite interesting that the two less appreciated parts were Act I (which contextualizes the result by comparing it to similar values) and the summary act (presenting all the highlights in a single slide). Although this originally came as a surprise to us, with the benefit of the hindsight we reckon that it should not have been surprising in the first place: users love concise

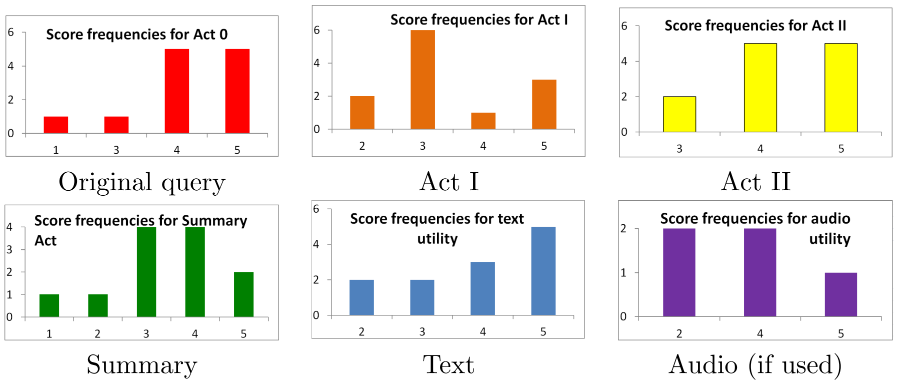


Figure 17: Evaluation of the usefulness of Cinecubes' parts, in a scale of 1 (worst) to 5 (best); x-axis depicts each score and y-axis the number of users that assigned it

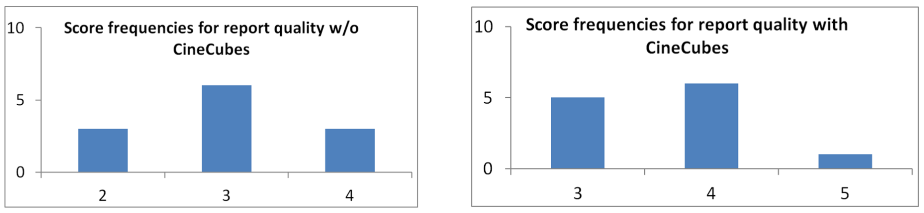


Figure 18: Evaluation of the original and the Cinecubes' report, in a scale of 1 (worst) to 5 (best); x-axis depicts each score and y-axis the number of users that assigned it

reporting of facts and dislike information provided in large volumes to them. The free-form comments of the users and a post-mortem discussion with them confirmed this observation. The contextualization and the summary acts provide too much information (and in fact, too many highlights). So, although the peak is in the median value (3), the average value for both Act I and the Summary Act was 3.4 stars and the distribution of values towards the high end, the phenomenon was not so heavy tailed on the higher values as for Act II and Act 0. *Lesson learned: above all, be concise!*

The textual part was quite appreciated by most of the users; at the same time, out of 5 users that worked with audio, the result was split in half in terms of likes and dislikes. This is both due to the quality of the produced audio by the TTS and the quality of the text that is served to it as input. A lesson learned here is that *audio seems to be useful for some users but not for all*; so, it should be optional, which can provide gains in terms of efficiency without affecting effectiveness.

5.4.3. Evaluation of the produced reports

The users were also asked to assess the quality of the produced report with the benefit of the hindsight. The results are depicted in Fig. 18.

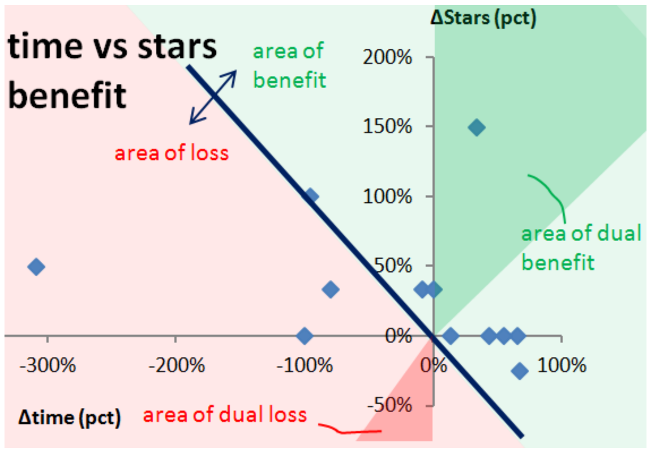
Overall, the distribution appears shifted by one star upwards, with the median shifting from 3 to 4. The average value was raised from 3 to 3.7 which is a 23% improvement of quality. The free-form comments indicated that the score would have been higher if the tool automatically produced graphs and charts (an issue of small research but high practical value).

5.4.4. Time considerations

We asked the users to measure the time they spent for the creation of each report. In Fig. 19 we depict the actual data as well as their visual representation. The graphical representation of Fig. 19 compares the benefit in time (x-axis) over the benefit in stars (y-axis). In other words, does it pay off to spend more time working with the system for the quality of the report one gets? The diagonal line splits the plane in two parts: the right, green part is the area where you get more quality for the time you invest; the left, rose part is an area of loss. The intensely colored parts of the two areas are parts with two-fold benefit (more quality for less time) or loss (less quality for more time).

The findings are quite interesting. A first very interesting observation lies in the fact that CineCubes did not result in clear time gains, as we would expect. In fact, there was a large number of people who spent more time with CineCubes than with the simple querying system! Although this originally did strike us as a failure, a better look at the data (and the graph) refutes this result. When we sorted the data by time spent without CineCubes (second column), it was clear that the users who demonstrated this kind of time loss were the ones who spent too little time (way less than the rest) for their original report. The small amount of time devoted to the original report, skyrockets the percentage deficit (a user who spends 10 minutes for the original report and 20 minutes for Cinecubes, gets a 100% time penalty). At the same time, this resulted also in an original report of rather poor quality, and significant improvements in the quality of the report, too. This also explains why there are no users with dual loss. Again, the explanation for the time increase is that the users spent extra time to go through the highlights offered by CineCubes.

A second observation concerns the people who spent less time with CineCubes than without it. These are people who invested more time working with data than the previous group. In all but one cases, there was no loss of quality for this group of users. So, clearly, for the people who would spend at least 30 minutes for their original report, there is a benefit in time gains. In fact, in all but one cases, the benefit rises with the time spent in the original report (the relationship between $\Delta Time$ and the $pct\ \Delta Time$ for the people with a positive time gain is almost linear, with a Pearson correlation of 0.940; the same applies for the correlation of the time spent without Cinecubes and $pct\ \Delta Time$ with a Pearson correlation of 0.868). It is interesting, that because these users devoted quite some time working with the data in the first place, they had a quite satisfactory report in the first place (in all but one cases, no less than 3 stars). Therefore, the improvement in terms of stars is on average half star out of five (although the distribution of values is clearly biased, as the last column of the data in Fig. 19 indicates). The speedup however rises on average to 37.5 minutes (or 46.00% as percentage) for these cases.



	Time to complete report (mins)				Quality of Report (stars)			
User Id	w/o CC	with CC	Δtime	pct Δtime	w/o CC	with CC	ΔStars	pct Δstars
3	10	20	-10	-100,00%	3	3	0	0,00%
4	11	45	-34	-309,09%	2	3	1	50,00%
2	23	45	-22	-95,65%	2	4	2	100,00%
5	23	25	-2	-8,70%	3	4	1	**33,33%**
21	25	45	-20	-80,00%	3	4	1	**33,33%**
8	25	25	0	0,00%	3	4	1	**33,33%**
12	30	26	4	13,33%	4	4	0	0,00%
17	60	40	20	33,33%	2	5	3	150,00%
6	70	40	30	42,86%	4	4	0	0,00%
1	71	25	46	64,79%	3	3	0	0,00%
15	100	45	55	55,00%	3	3	0	0,00%
16	105	35	70	66,67%	4	3	-1	-25,00%

Figure 19: Evaluation of the time gains versus the quality gains for the construction of the report with and without CineCubes

Lessons learned. For people in need of a fast report, conciseness is key, as too many results will slow them down; yet, CineCubes allows these people to create reports of better quality. For people who would be willing to spend more time to author a report in the first place, CineCubes speeds up their work by a factor of 46% in average.

6. Related Work

In this section, we discuss related work around the topic of our discourse. Specifically, research pertaining to our work can be identified in the fields of query recommendation, advanced OLAP operators, text synthesis from query results, and data narration. We present each of these categories in the following.

6.1. Query Recommendations

The first area that relates to our work is the area of query recommendation. Roughly speaking, the general theme of this area revolves around the situation where the user has submitted a query to the system and the system suggests one or more related queries to the user as a guide that helps him continue his search. The suggestion can be based on the user's profile, history of queries, history of other users' queries, or other information. There is an excellent survey on the topic by Marcel and Negre [12]; thus, here we restrict ourselves to a handful of characteristic approaches and refer the interested reader to [12] for a broader discussion.

The query recommendations that are related to our work can be classified in two orthogonal taxonomies, already found in [12]. In terms of the *data management environment* within which query recommendation takes place, we can distinguish between works in the general field of databases and works in the specific field of OLAP. In terms of the *means* employed for the recommendation of queries, we can discern methods exploiting profiles, methods exploiting query logs and hybrid methods.

6.1.1. Database-related efforts

Stefanidis, Drosou and Pitoura [13], propose the enrichment of the results of a query with extra tuples that may have potential interest to the user. The method is entitled YMAL ("You May Also Like"), and tries to find tuples in the underlying relational database on the grounds of a principled tuple-recommendation approach. One of the contributions of [13] is that the authors suggest a classification of methods for recommendation: (a) current state based, (b) history based, and (c) based on external sources.

The current-state approach makes use of the current query result and schema in conjunction to the data of a database to produce the YMAL result. To implement this approach the authors suggest three kinds of analysis: (i) local, (ii) global and (iii) hybrid analysis. Local analysis involves finding patterns in the results of a query and searching the rest of the database in order to add to the original result extra tuples that abide by the discovered patterns. The global approach searches the database to find values that are correlated to the values involved in the selection condition of the submitted query; the k most correlated of these values are selected and tuples that contain them are recommended to the user. To calculate relevant tuples, the history-based approach uses (i) the previously submitted queries of the user, and, (ii) similar sessions of other users that have similar behavior of the current user. The last of these approaches, involves external sources and does not search the local database for relevant tuples, but the web or another schema.

Chatzopoulou et al., in [14] propose a recommender system called QueRIE (Query Recommendations for Interactive data Exploration). The main goal of this recommender system is to help the common user, who is not familiar with SQL and database schemata, to find parts of database with useful or interesting information. To this end, the authors have implemented a system with the ability of tracking the querying behavior of a user and generating a personalized query recommendation. The system is built on a simple premise, inspired by Web recommender systems: if a user A has similar querying behavior to user B, then they are likely interested in the same data. Hence, the queries of user B can serve as a guide for user A.

6.1.2. OLAP-related methods

Cariou et al., in [15] describe a method to help user to explore OLAP data. The proposed method combines OLAP and data mining techniques to facilitate the process of the exploration of a data cube by identifying the most relevant dimensions to expand. The implementation of this task is performed in a step by step approach. In each step, the most relevant dimensions from the current session of the user are identified and then, the system suggests to the user which one to explore first. The dimensions are of relatively simple structure with two levels only (ALL and detailed). The main idea behind the method is that each dimension takes a degree of interest. Each time the degree of interest is calculated by the amount of information revealed when including the details of this dimension in the grouping of the detailed data (remember that each dimension has only two levels; thus including it in the group by practically means that the dimension's detailed values split the grouping space with a factor equal to their number).

A different approach for suggesting an OLAP query to user is introduced in the work of Giacometti et al., [16]. Unlike [15], the authors of [16] use the query log of previous users to find similar queries which can give information to user that he may not know it is available. The main idea is to recommend to the user the discoveries detected in former sessions of other users that investigated the same unexpected data as the current session. To this end, the proposed method analyzes the query log to discover pairs of cells at various levels of detail for which the measure values differ significantly. In addition, the method analyzes the current query, in order to detect if a particular pair of cells for which the measure values differ significantly can be related to what is discovered in the log.

Aligon et al., in [17] work along the same context and provide some very interesting insights for log-based OLAP sessions. A first major result of [17] has to do with the fact that the user study conducted in this paper gives a first account of what users deem interesting in characterizing a query. Apparently, users think of the selection predicate as the most characteristic feature of a cube query; other features in decreasing order of importance are the set of groupers and the set of measures. The paper also gives a detailed survey of similarity measures for OLAP queries and an experimental verification of which similarity function seems to capture best the intuition of users for OLAP queries.

6.2. Advanced OLAP operators

Apart from recommending queries to the users, related research has explored the possibility of providing users with explanations for the results they observe in an OLAP report. We distinguish the work of Sarawagi in a series of papers in VLDB and briefly summarize the results.

In [19], we meet the *DIFF* operator is described with the aim to help the analyst get a concise set of tuples explaining the reasons for drops or increases observed at an aggregated level. As input, the operator receives two cells of a report that are different. As output, the operator returns a set of tuples that best describe this difference. To achieve this result, the paper proposes a greedy and a dynamic-programming algorithm. The idea is that the operator keeps as fixed the common selections that characterize the originally selected cells (so, it is important that they do have some common selection conditions for the computation to make sense) and drills-down the levels of aggregation for the involved hierarchy that is produced by the combination of these common dimensions. The crux of the approach is that it computes the respective difference when the data are aggregated for any of the tuples in this multidimensional space. Every tuple in this multi-level space is compared to its "parent" tuple (in one level of aggregation higher) and, if selected, it is placed in the top-N results that will ultimately be displayed to the user. For a tuple to make it in the top-N it has to contribute a significant percentage of the difference of the original cells compared to the contribution of its father.

The same author, Sarawagi, in [20], presents a tool that helps users explore the multidimensional OLAP data using their prior knowledge of the data. This tool uses a profile that tracks down the areas of the cube that the user has visited in the past, and thus, it is aware of what the user already knows about the data. Then, the tool guides the user to unexplored data that he will find most informative. The author in [20] describes a method that uses the classical Maximum Entropy principle and a profile per user to recommend to the user the parts of the cube which contain the most surprising values compared to what the user has already seen

In [21], Sathe and Sarawagi introduce the operator *RELAX* which helps the user of OLAP data to go from a detailed level of information to a more general one, in order to verify whether a pattern observed at the detailed level is also present at a more summarized level. The operator reports in a single step a summary of all possible maximal generalizations along various roll-up paths of the observed sub-cube. The goal is to report all possible consistent and maximal generalizations. The term consistent means that all subsets of dimensions that are examined also abide by the pattern. On the other hand, the term maximal means that there is no superset of dimensions that can yield consistent generalizations. For the implementation of this operator the authors develop a two stage algorithm. In the first stage, the algorithm finds all possible maximal generalizations using aggregation queries. In the second stage, the algorithm uses the results of the first stage and finds summarized exceptions of the generalizations.

6.3. Text synthesis from query results

In [22], Simitsis et al., propose a method to synthesize a textual answer in response to a query over a relational database. The authors employ a graph model with nodes being attributes and relations, edges being part-of relationships and join relationships and labels for relations, attributes and edges (labels are used to produce a text for a query's result). The method takes a query as input, computes its result and tries to produce a sentence for each of the tuples that appear in the result. This is derived by following specific graph navigation patterns, each of which produces a different type of text.

6.4. Data Narration, Narrative Visualization and Visual Analysis

Last but not least, a research area that is closely related to our approach involves data narration. Whereas data visualization involves depicting data to the user in a way that allows the user to extract interesting information easily, data narration tells the story of data, i.e., gives context, explanations and, fundamentally, appropriate visualizations. Due to the close relationship of visualization and narration, the area is also referred to as narrative visualization.

In an interesting article [5], Kosara and McKinley, researchers in Tableau Software, highlight how *storytelling* can be the next step for visualization, and -as we add here- for gaining insights into the observed data. To quote the authors "Humans have always tied facts together into stories, effectively presenting information and making a point in a memorable way". The connection between data and stories is being elevated only very recently; however, it is clear that a story supported by data gains in authority and trust, and at the same time, data-based insights are way more memorable when successfully blended in a story. It is only natural, then, that data storytelling is becoming more and more popular these days, with sites and tools like GapMinder, ManyEyes and Tableau Software allowing people to work with data and gain insights, but also with newspapers and mass media (like New York Times, Washington Post and The Guardian) using infographics to express stories in both a vivid and data-driven, convincing way.

In a highly cited paper, Segel and Heer [6] provide a survey and classification of narrative visualization techniques, along with a very long list of examples that demonstrate actual cases of narrative visualization, mainly in newspapers. The authors reviewed a vast number of examples to come up with a taxonomy for the design space of narrative visualization. The taxonomy organizes the characteristics of visualizations in seven groups, organized in three families. The first family-group concerns the genre of the visualization (e.g., slide show, comic strip, poster, film, etc). The second family concerns the techniques used for the visualization of a story. The first group in this family concerns the structure of the visualization (consistent visual platforms, progress indication, etc), the second group concerns visual highlight features (zoom, motion, audio, etc), and the third group concerns the way transitions among visualized information is made (e.g., via continuity principles, or animation). The third family concerns the structuring and interactivity of a visual presentation. The first group in this family concerns the order of the presentation's parts, the second group concerns the modes of interactivity via which the user can interact with the visualization (e.g., hovering tooltips, tacit tutorials, navigation buttons, etc) and the third group concerns the messaging tools employed to inform the user on important parts of the presentation (e.g., captions, annotations, summaries, etc). One typical difference between traditional storytelling and data narration that [6] highlights concerns the potential for interactivity in the latter. In fact, one of the main problems of data narration is built around the interactivity issue: how does one balance (a) a certain amount of control that the author needs to preserve and impose in order to manage to tell the story in the end, versus (b) the need of the users to go through alternative explorations, pose verification or explanatory questions, and in any case, work with the data in ways not already present in the linear storytelling of the author. The authors highlight three ways of interacting with the users. First, the Martini Glass principle, where the presentation starts with a user-centric part for the exploration of data, continues with a strict, non-interactive sequence to convey its core message and concludes with a large number of choices for follow-up exploration by the user. The Interactive Slideshow principle follows a typical slideshow format (i.e., a sequence of author-driven "slides") but allows some degrees of interaction within each slide, allowing thus the user to interact with the data mid-narrative. The Drill-Down Story principle, gives the most degrees of interaction to the user, by presenting a central theme as a portal for interactive data exploration.

In a similar trend, Hullman and Diakopoulos [23] conduct a similar study to provide a taxonomy of techniques ("rhetoric") used to illustrate or obscure information. Information access rhetoric methods concern which data to include or exclude, and at what level of aggregation or abstraction. Provenance rhetoric methods concern ways to highlight or obscure the source of information and the uncertainty involved in the reporting of facts and estimations. Mapping rhetoric concerns techniques like visual metaphors, contrast, color coding etc, used to map data to a visual representation that conveys a message. Linguist-based rhetoric methods are focused on the textual level and concern how the presenter uses text to make a point or make the user be involved.

A central role in data narration is played by interactive *visual analytics* that facilitate the extraction of information from data via interactive visualization and automated information mining. Ben Shneiderman gave the the famous "Visual Information Seeking Mantra" back in 1996 [24], as the foundation of visual design: Overview first, zoom and filter, then details-on-demand. Fifteen years later, in [25], Heer and Schneiderman provide a taxonomy of visual tools that facilitate the interaction of a user with data, in what the authors call "analytic dialogues". The taxonomy consists of 12 typical task types grouped in categories as follows: (a) data and view specification (visualize, filter, sort, and derive), practically facilitating the setup and focus of a data visualization task, (b) view manipulation (select, navigate, coordinate, and organize), practically covering the part where a user explores the presented data interactively, and (c) analysis process and provenance (record, annotate, share, and guide), covering the part where the user has gained insight and is preparing his data-driven story's presentation.

6.5. Relationship of our work with the state of the art

Concerning all the above works, our method comes with an extensible architecture that is especially constructed with a mindset of being able to plug in more and more of them, both at the part where new queries can be added and in the part where new analyses can be performed over their results. Our Act II resembles the DIFF operator to a certain extent, in the sense that it tries to explain the reasons of the originally observed result. DIFF goes one step further, in providing maximal explanations by picking the most profitable rows. Although DIFF can be integrated in our tool, the emphasis so far has been in coming up with a prototype that can provide a reasonable CineCube movie; research results like DIFF can be integrated in the tool in subsequent tool extensions and revisions. The same applies for all the other advanced OLAP operators.

Concerning text synthesis, we avoid describing the result of a query row-by-row, as [22] does. On the contrary, we provide an extensible architecture where each highlight extraction method comes with a generic text to describe the detected highlights. Of course, improvements on the produced text are clearly part of future work.

We would also like to highlight that our method is synchronized with the key findings of [17], as (a) the main criterion that we use for suggestions in Act I is what [17] identified as the key characteristic feature of cube queries, i.e., the selection condition and (b) the key feature for explaining results in Act II is the second most characteristic feature, groupers.

6.6. Relationship to our previous work

A first version of this paper has appeared in [26]. In the present version of the paper, we extend [26] in the following ways. We provide a detailed explanation of the internal architecture and the extensibility mechanisms of CineCubes. We also provide an assessment on the effort needed to extend CineCubes. We discuss in detail the results of our experimental study, both in terms of the anatomy of the time spent in the different tasks and acts and in terms of usability, via a user study. We accompany these extensions, with explanations, an extended discussion of our motivations, and a detailed survey of the related literature that were not present in the short version of this paper.

7. Discussion

In this paper we have introduced CineCubes, a system that allows the automatic generation of a CineCube movie, over an OLAP database, with a simple user query as starting point. To produce a movie, we need several "episodes", text, and voice. To this end, we have automated (a) the process of complementing the original query with additional queries that provide contextualization (by comparing its results to the results of queries carrying similar information) and (b) the process of extracting meaningful relationships within the data, as we search for interesting patterns in the results of all these queries. Moreover, we have also automated the generation of text describing these findings and their conversion of this text to audio. Finally, we have shown that all the above can be packaged in a PowerPoint presentation, practically presenting a small data movie to the user.

We believe that our method creates new research ground, by bundling all the individual steps in a single-yet-extensible framework that allows data workers gain insights. This is -in our point of view- the core contribution of this paper. Naturally, as typically happens in science, this new research ground can be further expanded in a systematic way. In the following, we list a few important problems and opportunities.

Extensibility. Cinecubes comes with an extensible architecture that is especially constructed with a mindset of hosting more and more techniques both from existing and foreseeable research results in the areas of knowledge extraction, query recommendation, text analysis, trend prediction, and data visualization. We firmly believe that this extensibility can and should be exploited via a synergy with the research community in order to further enhance the benefits of this approach.

Efficiency. Scaling with data size and complexity, let along with user needs, *in user time,* is also necessary for an effort like this to succeed. Techniques like multi-query optimization have a good chance to succeed, especially since we operate with a known workload of queries as well as under the divine simplicity of OLAP.

Can *I* be the director? Interactively maybe? Personalization and interactivity are two clear paths for extending the approach mentioned here. Interactivity, i.e., the possibility of allowing the user to intervene and semi-automatically guide the query generation can be served in many ways (e.g., the Martini Glass and the Interactive SlideShow of [6]). This user-driven interaction can be aided by incorporating extra knowledge into the report generation – e.g., via user profiles or user logs, like in [16]– that guide the users in their explorations around the basic results that they see in a CineCubes movie.

Be compendious; if not, at least be concise! The single most important challenge that the research problem of answer-with-a-movie faces is the *identification of what to exclude.* The problem is not to add more and more recommendations or findings (at the price of time expenses): this can be done both effectively (too many algorithms to consider) and efficiently (or, at least, tolerably in terms of user time). The main problem is that it is very hard to keep the story both interesting and informative and, at the same time, automate the discovery of highlights and findings. To address this task, a clearly important topic of research involves the automatic merging, ranking and pruning of highlights.

8. Acknowledgments

The authors would like to thank the anonymous reviewers for constructive comments and suggestions.

This research has been co-financed by the European Union (European Social Fund - ESF) and Greek national funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF) - Research Funding Program: Thales. Investing in knowledge society through the European Social Fund.

References

[1] G. Dove, S. Jones, Narrative visualization: Sharing insights into complex data, in: Presented in Interfaces and Human Computer Interaction (IHCI 2012), 2012, available at http://openaccess.city.ac.uk/1134/.

[2] P. Hanrahan, Analytic database technology for a new kind of user - the data enthusiast, in: Keynote speech at the ACM SIGMOD International Conference on Management of Data (SIGMOD 2012), 2012, available at http://www.graphics.stanford.edu/ hanrahan/talks/enthusiasts.pdf.

[3] The Apache Organization, See https://poi.apache.org/, The Apache POI Project (2013).

[4] DFKI, See http://mary.dfki.de/, The MARY Text-to-Speech System (2013).

[5] R. Kosara, J. D. Mackinlay, Storytelling: The next step for visualization, IEEE Computer 46 (5) (2013) 44–50.

[6] E. Segel, J. Heer, Narrative visualization: Telling stories with data, IEEE Transactions on Visualization and Computer Graphics 16 (6) (2010) 1139–1148.

[7] C. S. Jensen, T. B. Pedersen, C. Thomsen, Multidimensional Databases and Data Warehousing, Synthesis Lectures on Data Management, Morgan & Claypool Publishers, 2010.

[8] P. Vassiliadis, S. Skiadopoulos, Modelling and optimisation issues for multidimensional databases, in: 12th InternationalConference on Advanced Information Systems Engineering (CAiSE 2000), 2000, pp. 482–497.

[9] R. McKee, Story: substance, structure, style and the principles of screenwriting, HarperKollins pubs., 1997.

[10] E. R. Tufte, Visual Explanations: Images and Quantities, Evidence and Narrative, Graphics Press, 1997.

[11] A. S. Maniatis, P. Vassiliadis, S. Skiadopoulos, Y. Vassiliou, G. Mavrogonatos, I. Michalarias, A presentation model and non-traditional visualization for OLAP, IJDWM 1 (1) (2005) 1–36.

[12] P. Marcel, E. Negre, 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 (EDA 2011), 2011, pp. 119–134.

[13] K. Stefanidis, M. Drosou, E. Pitoura, "You May Also Like" Results in Relational Databases, in: 3rd International Workshop on Personalized Access, Profile Management, and Context Awareness: Databases (PersDB 2009), 2009.

[14] G. Chatzopoulou, M. Eirinaki, S. Koshy, S. Mittal, N. Polyzotis, J. S. V. Varman, The querie system for personalized query recommendations, IEEE Data Eng. Bulletin 34 (2) (2011) 55–60.

[15] V. Cariou, J. Cubillé, C. Derquenne, S. Goutier, F. Guisnel, H. Klajnmic, Built-in indicators to discover interesting drill paths in a cube, in: Proceedings of 10th International Conference on Data Warehousing and Knowledge Discovery (DaWaK 2008), 2008, pp. 33–44.

[16] A. Giacometti, P. Marcel, E. Negre, A. Soulet, Query recommendations for OLAP discovery-driven analysis, IJDWM 7 (2) (2011) 1–25.

[17] J. Aligon, M. Golfarelli, P. Marcel, S. Rizzi, E. Turricchia, Similarity measures for OLAP sessions, in: Knowledge And Information Systems (KAIS), 2013, accepted paper, available at http://www.julien.aligon.fr/wp-content/uploads/2012/09/kais.pdf.

[18] C. Sapia, On modeling and predicting query behavior in olap systems, in: Proceedings of the Intl. Workshop on Design and Management of Data Warehouses, DMDW'99, 1999, p. 2.

[19] S. Sarawagi, Explaining differences in multidimensional aggregates, in: Proceedings of 25th International Conference on Very Large Data Bases (VLDB'99), 1999, pp. 42–53.

[20] S. Sarawagi, User-adaptive exploration of multidimensional data, in: Proceedings of 26th International Conference on Very Large Data Bases (VLDB 2000), 2000, pp. 307–316.

[21] G. Sathe, S. Sarawagi, Intelligent rollups in multidimensional olap data, in: Proceedings of 27th International Conference on Very Large Data Bases (VLDB 2001), 2001, pp. 531–540.

[22] A. Simitsis, G. Koutrika, Y. Alexandrakis, Y. E. Ioannidis, Synthesizing structured text from logical database subsets, in: 11th International Conference on Extending Database Technology (EDBT 2008), 2008, pp. 428–439.

[23] J. Hullman, N. Diakopoulos, Visualization rhetoric: Framing effects in narrative visualization, IEEE Transactions on Visualization and Computer Graphics 17 (12) (2011) 2231–2240.

[24] B. Shneiderman, The eyes have it: A task by data type taxonomy for information visualizations, in: Proceedings of the 1996 IEEE Symposium on Visual Languages (VL'96), 1996, pp. 336–343.

[25] J. Heer, B. Shneiderman, Interactive dynamics for visual analysis, ACM Queue 10 (2) (2012) 30.

[26] D. Gkesoulis, P. Vassiliadis, Cinecubes: cubes as movie stars with little effort, in: Proceedings of the 16th International Workshop on Data Warehousing and OLAP (DOLAP 2013), 2013, pp. 3–10.