CineCubes: Cubes As Movie Stars with Little Effort

Dimitrios Gkesoulis Panos Vassiliadis
Dept. of Computer Science, Univ. of Ioannina
Ioannina, 45110, Hellas
{dgesouli, pvassil}@cs.uoi.gr

ABSTRACT

In this paper we investigate how we can exploit the existence of a star schema in order to answer user OLAP queries with *CineCube* movies. Our method, implemented in an actual system, 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. 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 cube movie, which is wrapped as a PowerPoint presentation and returned to the user.

Categories and Subject Descriptors
H.2.4 [**Database Management**]: Systems – *query processing*

General Terms
Algorithms, Design, Human Factors

Keywords
OLAP, management of query results, query recommendation

1. INTRODUCTION

Can we answer user queries with 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 make a first attempt towards showing 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. Moreover, we provide an extensible method to accompany a query result with results of complementary queries that allow the qualitative assessment of its information content.* Interestingly, *a meaningful 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 movie where cubes star.*

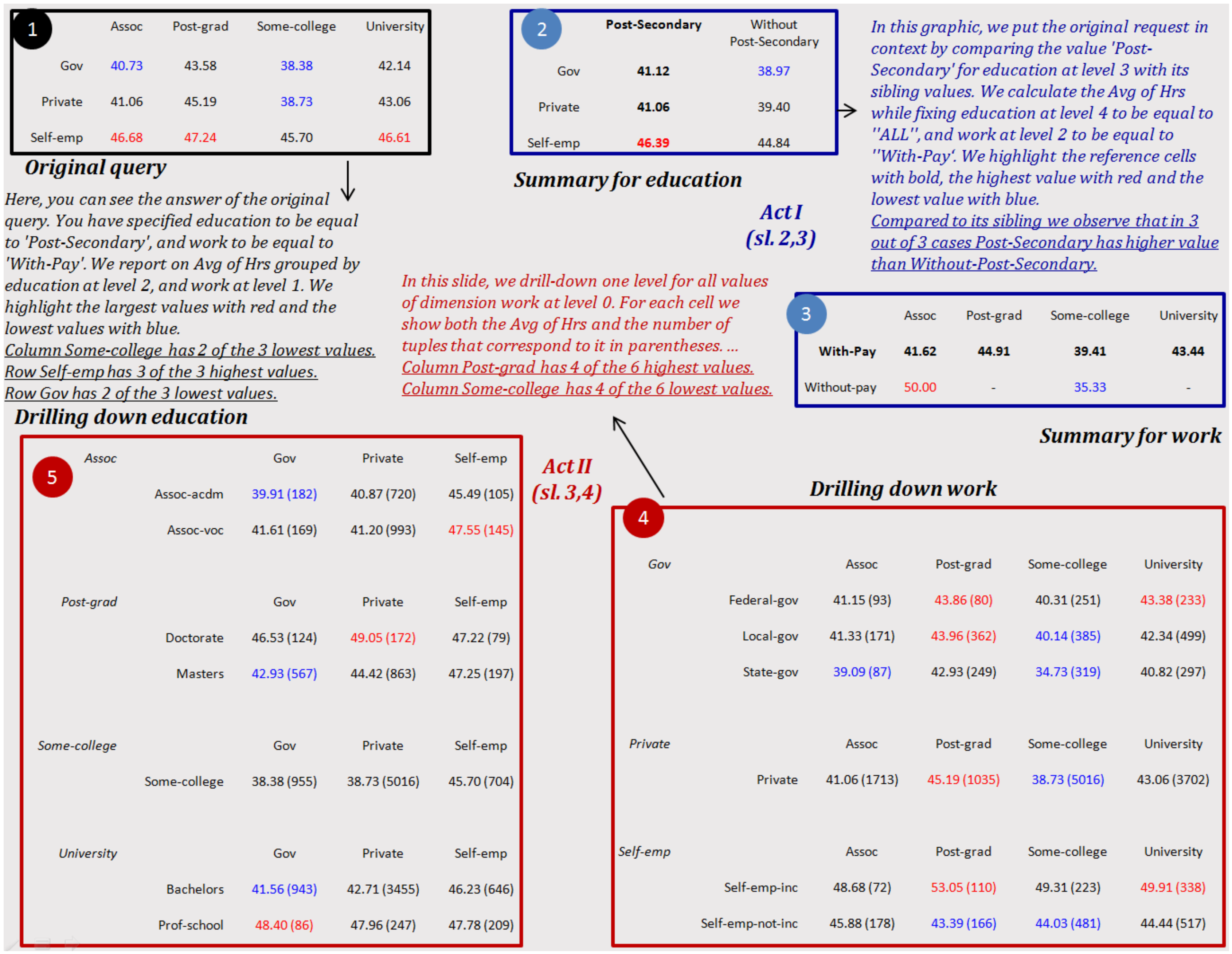
Assumptions. In this paper we make a realistic assumption that empowers us with the ability to address the challenge in a clear setting. We assume the existence of a star schema with clean, reconciled hierarchies of reference data; we also assume that the end users are interested in working with OLAP queries over these data. We exploit the star schema in order to generate complementary queries automatically.

The movie's parts and their extension. Much like movies, we organize our stories in acts, with each act including several episodes all serving the same purpose. Our method involves *two extensibility mechanisms, (i) one concerning the generation of complementary queries that contextualize the original result and give insight and (ii) another concerning the automatic identification of interesting information within the results of each query*. We further exploit the outcome of the latter mechanism, as it is the main means via which we accompany results with automatically generated text (which in turn, is then fed to text-to-speech conversion in order to generate audio).

Low technical barrier. 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 [2] and for text to speech conversion [9] allow us to produce a pptx 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.

Contribution & call to arms. The individual parts of the method are not the core contribution of the paper; however, it is their principled and extensible bundling in a single, extensible tool that creates a research opportunity and an actual contribution. The fundamental message carried from this paper is that it is feasible (and we have done it) to drastically change the way users interact with business intelligence tools via simple programmatic APIs. Moreover, we can systematically expand this research ground by plugging in more and more techniques both from existing and foreseeable research results in the areas of text commenting, query recommendation and data visualization.

Roadmap. In Section 2, we give an overview of the method, mostly via a reference example and explain the low technical barrier of the method. In Section 3, we discuss our method's internals. In Section 4, we show experimental results. In Section 5, we discuss related work. We conclude with a presentation of open issues in Section 6.



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.

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)

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)

Figure 1. An excerpt of a CineCube story over the Adult data set

2. METHOD OVERVIEW

2.1 Constructing a CineCube Story

A really useful characteristic of cubes is that *dimensions* provide a *context* for facts [6]. 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:

(a) 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.

(b) 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.

(c) $\gamma(M)$ 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 make a short 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 5) 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 [17].

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 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).

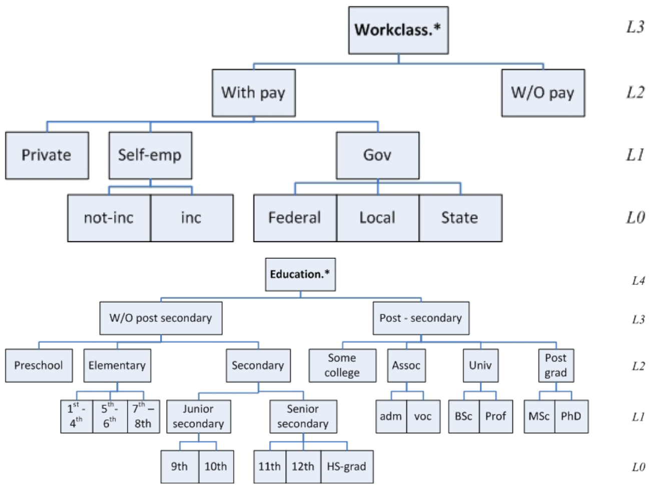


Figure 2. Dimensions Workclass and Education

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 Fig. 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 Fig. 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 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.

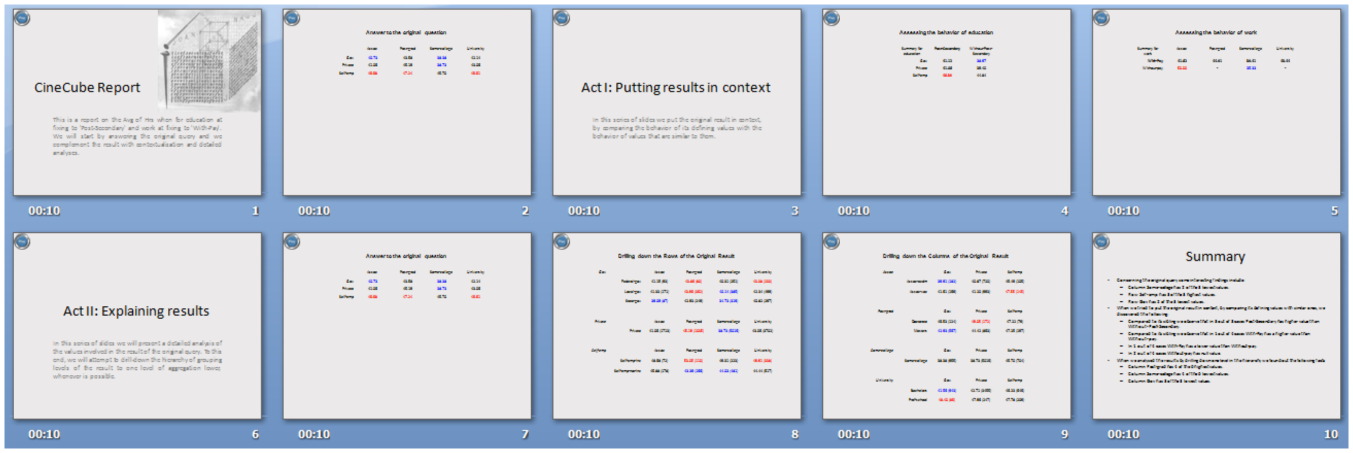

CineCube Report
Answer to the original question
Act I: Putting results in context
Assessing the behavior of education
Assessing the behavior of work
Act II: Explaining results
Answer to the original question
Drilling down the Rows of the Original Result
Drilling down the Columns of the Original Result
Summary
00:10
00:10
00:10
00:10
00:10
00:10
00:10
00:10
00:10
00:10

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

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: 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 [10].

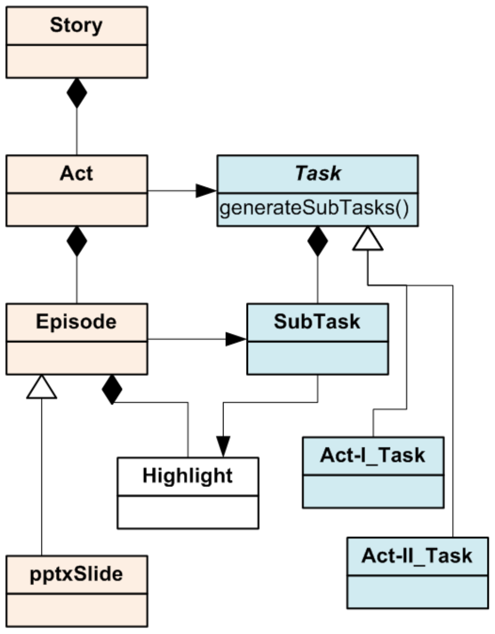


Figure 4. Extensibility mechanism for CineCubes

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 is accompanied by a *procedural-based structure*, with a set of classes that actually get the job done. Here, we introduce the *two extensibility mechanisms* that allow our method to be extensible to all sorts of algorithms for extra results and discoveries. *There are two "dimensions" of extensibility: (i) what kind of query results (episodes) we collect from the database, and, (ii) how we automatically discover important findings within these results.*

The first extensibility mechanism concerns the generation of queries (and slides) within each *Act*. The abstract class *Task* is the generator of the queries of each *Act*: therefore, we materialize it differently for each kind of *Act* (here we have 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*.

The second extensibility mechanism concerns the determination of key findings, or *Highlight*s within each *Episode*. We fundamentally consider the presentation of results as a 2D matrix on the screen[1]; 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). Class *Highlight* is a point of extensibility where

[1] 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* [16].

methods for result extraction can be added to search for more results within the answer of a query.

There are several other classes that accompany the above core of the method which are omitted from this discussion for lack of space. These classes concern the management of cubes and their relationship with a relational database, the construction of the text, the derivation of the audio for the constructed text and so on.

2.4 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. In this subsection, we explain the main technologies via which our PowerPoint presentations are programmatically constructed.

Apache POI [2] 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:

```
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”); …
```

As we will discuss later, 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 [9], 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”)); …
```

Naturally, there are several nuts and bolts to fine tune. *However, 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.*

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* (in the form of $anc_{L_1}^{L_2}$) 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 [17], accurately summarized in [7] – for lack of space we refer the interested reader to these publications for a full description.

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 $[L_1,\dots,L_n,M_1,\dots,M_m]$), is an expression of the form:

$$c=(DS^0,\phi,[L_1,\dots,L_n,M_1,\dots,M_m],[agg_1(M_1^0),\dots,agg_m(M_m^0)]),$$

where:

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

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

```
SELECT L₁,…,Lₙ, agg₁(M₁⁰),…,aggₘ(Mₘ⁰)
FROM DS⁰ INNER JOIN D₁ … INNER JOIN Dₙ
WHERE ϕ
GROUP BY L₁,…,Lₙ
```

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:

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

The results of a cube query of this form can be 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. Also, although, there are several other 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.

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 $L_i \in \text{children}(\text{parent}(v_i))$.

Formally, given an original query

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

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

$$q^s = (DS^0, \phi_1 \wedge \dots \phi_x^* \wedge \dots \wedge \phi_k, [L_\alpha, L_\beta], agg(M)), \phi_i: L_i=v_i,\ i=1,\dots,x-1,x+1,\dots,k, \phi_x^*: L_{x+1}=anc_{L_x}^{L_{x+1}}(v)$$

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

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

its two *marginal sibling queries* are

$$q_a^S = (DS^0, \phi_1 \wedge \dots \phi_x^* \wedge \dots \wedge \phi_k, [L_\alpha, L_x], agg(M)), \phi_i: L_i=v_i,\ i=1,\dots,x-1,x+1,\dots,k, \phi_x^*: L_{x+1}=anc_{L_x}^{L_{x+1}}(v)$$

$$q_b^S = (DS^0, \phi_1 \wedge \dots \phi_x^* \wedge \dots \wedge \phi_k, [L_x, L_\beta], agg(M)), \phi_i: L_i=v_i,\ i=1,\dots,x-1,x+1,\dots,k, \phi_x^*: L_{x+1}=anc_{L_x}^{L_{x+1}}(v)$$

Example. The original query is expressed as:

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

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):

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

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

$$q = (DS^0, \phi_1 \wedge \dots \wedge \phi_k, [L_\alpha, L_\beta], agg(M)), \quad \phi_i: L_i=v_i,\ i=1,\dots,k$$

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

$$q^c = (DS^0, \phi_1 \wedge \dots \wedge \phi_k \wedge \phi_c, [L_\alpha, L_\beta], agg(M)), \phi_i: L_i=v_i,\ i=1,\dots,k,\ \phi_C: \phi_\alpha^c \wedge \phi_\beta^c \equiv L_\alpha=v_\alpha^c \wedge L_\beta=v_\beta^c$$

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

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

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

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*.

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

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

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

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

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 Act*s 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 computation of the contents and presentation of the *Operational Acts* is outlined in the Algorithm of Figure 5. Here, we would like to stress the extensibility aspect again: depending on the *Act* (and more specifically, on its operational *Task* counterpart), the queries of the subtasks are specialized per slide. Moreover, highlights, text and audio are produced via dedicated manager classes (not shown in Fig. 4 for lack of space).

The *Summary Act* is simply a slide with the text of the highlights copied to it, organized per act. However, the *Wrapping-up* Act introduces a few programmatic tasks worth mentioning here. Basically, 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. To deliver a presentation in the form that we wish to have it, we proceed as follows: (i) we unzip the pptx in a temporary folder (remember: 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); (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.

4. EXPERIMENTS

We have experimented with the Adult data set by assessing the time needed for generating a presentation for different kinds of original queries. 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.

	# 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 Gen. & 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**

Figure 6. Time breakdown (msec) for the method's parts

We have measured the time needed to perform each part of the method. We varied the number of atomic selection conditions within the WHERE clause and measure the time needed per step of the method (measured in milliseconds). 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 Fig. 6). 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 number of texts generated slowly increases too. Concerning every other part 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. RELATED WORK

Strongly related to our work is the area of query recommendation, where the user submits a query to the system and the system suggests one or more related queries to the user as a guide for continuing his search. There is an excellent survey on the topic by [8] that organizes work in two orthogonal taxonomies. In terms of the *data management environment* we can distinguish between works in the general field of databases [15, 4] and works in the specific field of OLAP [3,5]. In terms of the *means* employed for the recommendation of queries, we can discern methods exploiting profiles, methods exploiting query logs and hybrid methods. [1] provides interesting insights for OLAP sessions. A second area of research involves advanced OLAP searches (practically in the realm of knowledge extraction). The *DIFF* operator [11] returns a concise set of tuples explaining the reasons for drops or increases observed at an aggregated level. The operator *RELAX* [12], is used to verify whether a pattern observed at the detailed level is also present at a more summarized level. Finally, [14] produces a textual description of a result, generating text on tuple-at-a-time basis, in a similar fashion that we do for highlights and [13] provides a survey and classification of narrative visualization techniques.

6. DISCUSSION OF OPEN ISSUES

Extensibility. Concerning all the above works, our tool comes with an extensible architecture that is especially constructed with a mindset of plugging 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. 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*. There are plenty of works in query recommendation (see discussion above), pattern verification [12], trend analysis, future prediction, to name only a few, that can be added to the tasks included in a tool.

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, esp., 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. The enrichment of the architecture with extra knowledge –e.g., user profiles or crowd-wisdom (via user logs)- and the possibility of intervening and semi-automatically guiding the query generation are topics with clear potential.

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 ranking and pruning of highlights.

7. ACKNOWLEDGMENTS

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.

8. REFERENCES

[1] J. Aligon, M. Golfarelli, P. Marcel, S. Rizzi, E. Turricchia. Similarity Measures for OLAP Sessions. Accepted in *Knowledge And Information Systems* (KAIS).

Available at http://www.julien.aligon.fr/wp-content/uploads/2012/09/kais.pdf

[2] The Apache POI Project. See https://poi.apache.org/

[3] V. Cariou, J. Cubillé, C. Derquenne, S. Goutier, F.Guisnel, H. Klajnmic, 2008. Built-In Indicators to Discover Interesting Drill Paths in a Cube. *DaWaK* (Turin, Italy, 2008), pp. 33-44

[4] G. Chatzopoulou, M. Eirinaki, S. Koshy, S. Mittal, N. Polyzotis, J. Varman, 2011. The QueRIE system for Personalized Query Recommendations. *IEEE Data Eng. Bull.* 34,2 (2011), pp. 55-60

[5] A. Giacometti, P. Marcel, E. Negre, A. Soulet, 2011. Query Recommendations for OLAP Discovery-Driven Analysis. *IJDWM* 7,2 (2011), 1-25

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

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

[8] P. Marcel, E. Negre, 2011. A survey of query recommendation techniques for data warehouse exploration. *EDA* (Clermont-Ferrand, France, 2011), pp. 119-134

[9] DFKI. The MARY Text-to-Speech System. See http://mary.dfki.de/

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

[11] S. Sarawagi, 1999. Explaining Differences in Multidimensional Aggregates. *VLDB* (Edinburgh, Scotland, 1999), pp. 42-53

[12] G. Sathe, S. Sarawagi, 2001. Intelligent Rollups in Multidimensional OLAP Data. *VLDB* (Roma, Italy 2001), pp.531-540

[13] E. Segel, J. Heer. Narrative Visualization: Telling Stories with Data. *IEEE Trans. Visualization & Comp. Graphics*, 16,6 (2010), 1139-1148.

[14] A. Simitsis, G. Koutrika, Y. Alexandrakis, Y.E. Ioannidis, 2008. Synthesizing structured text from logical database subsets. *EDBT* (Nantes, France, 2008) pp. 428-439.

[15] K. Stefanidis, M. Drosou, E. Pitoura, 2009. "You May Also Like" Results in Relational Databases. *PersDB* (Lyon, France, 2009).

[16] E.R. Tufte, 1997. *Visual Explanations*. Graphics Press

[17] P. Vassiliadis, S. Skiadopoulos, 2000. Modelling and Optimization Issues for Multidimensional Databases. *CAiSE* (Stokholm, Sweden, 2000), pp. 482-497