The Whys and Wherefores of Cubes

Matteo Francia
DISI - University of Bologna
Bologna, Italy
m.francia@unibo.it

Stefano Rizzi
DISI - University of Bologna
Bologna, Italy
stefano.rizzi@unibo.it

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

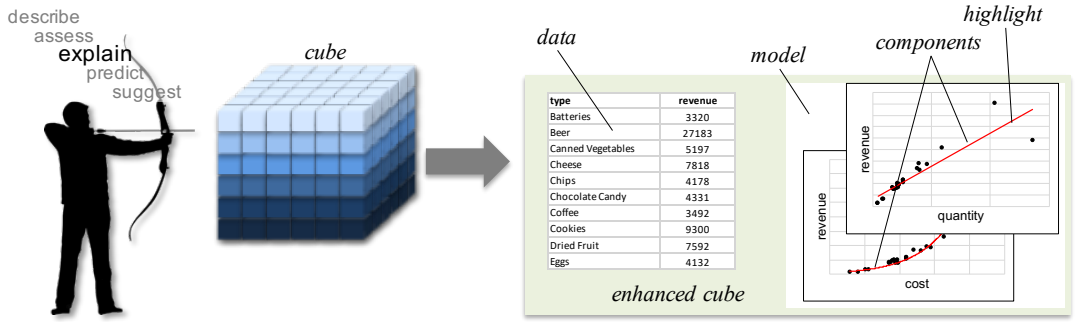


Figure 1: The IAM approach

ABSTRACT

The Intentional Analytics Model (IAM) has been devised to couple OLAP and analytics by (i) letting users express their analysis intentions on multidimensional data cubes and (ii) returning enhanced cubes, i.e., multidimensional data annotated with knowledge insights in the form of models (e.g., correlations). Five intention operators were proposed to this end; of these, describe and assess have been investigated in previous papers. In this work we enrich the IAM picture by focusing on the explain operator, whose goal is to provide an answer to the user asking "why does measure *m* show these values?". Specifically, we propose a syntax for the operator and discuss how enhanced cubes are built by (i) finding the polynomials that best approximate the relationship between *m* and the other cube measures, and (ii) highlighting the most interesting one. Finally, we test the operator implementation in terms of efficiency.

KEYWORDS

data cube, OLAP, analytics, correlation

1 INTRODUCTION

Despite the huge success of the OLAP (On-Line Analytical Processing) paradigm in supporting decision makers for their analyses of multidimensional cubes, it is now clear that this paradigm, alone, does no longer meet the sophisticated requirements of new-generation users. Among the directions taken by research to enhance OLAP, the *Intentional Analytics Model* (IAM) suggests to couple it with analytics [40]. The IAM approach relies on two main ideas: (i) users explore the data space by expressing their analysis *intentions* and (ii) in return they receive both multidimensional data and knowledge insights in the form of models. To achieve (i) five intention operators were proposed, namely, describe (describes one or more cube measures at some aggregation level, possibly focused on some level members), assess (judges one or more cube measures with reference to some benchmark), explain (reveals the reason behind the values of a measure, for instance by correlating it with other measures), predict (shows data not in the original cubes, derived for instance with regression), and suggest (shows data similar to those the current user, or similar users, have been interested in). As to (ii), first-class citizens of the IAM are *enhanced cubes*, defined as multidimensional cubes coupled with *highlights*, i.e., interesting components of models automatically extracted from cubes. An overview of the approach is shown in Figure 1. Noticeably, having different models automatically computed and evaluated in terms of their interest relieves the user from the time-wasting effort of trying different possibilities.

Among the five intention operators, describe and assess have been investigated in previous papers [9, 10, 12]. In this paper we enrich the IAM picture by focusing on the explain operator. An *explanation* is essentially a description of causation for an observed phenomenon; in practice, it answers the *why?* question for that phenomenon by providing a causal model for it [22]. In our context, we concentrate on providing explanation models for a measure the user is observing; thus, the goal of the explain operator will be to provide an answer to the user asking "why does measure *m* show these values?".

As envisioned in [40], several types of models can be used to this end, for instance:

- use regression analysis to correlate the values taken by *m* with those taken by another measure m' (e.g., sales revenues are roughly proportional to the quantity sold);
- find a Granger causality relationship [17] between *m* and another measure m' (e.g., a peak of deaths occurs after about one week from a peak of infections);
- establish an analogy between the values of *m* at different aggregation levels (e.g., the trend of sales revenues for beer closely reflects the one of revenues for drinks);
- find recurrent patterns that relate *m* to members and/or other measures [1] (e.g., the sales of panettone are always high in December)
- find the cube facts that give the highest contributions to *m* [23].

To give a proof-of-concept for explain, in this paper we restrict to the first model type, specifically, the one that establishes a polynomial relationship between m and m'.[1]

Example 1.1. Let a SALES cube be given, and let the user's intention be

with SALES explain revenue by type for year='2022'

First, the subset of facts for 2022 are selected from the SALES cube and aggregated by product type (in OLAP terms, a slice-and-dice and a roll-up operator are applied). Then, regression analysis is used to compare the revenue measure with each other cube measure and find the polynomials that best approximates their relationship. Finally, a measure of interest is computed for the components (i.e., for the polynomials) obtained, and the most interesting one is shown to the user (in Figure 1, the one showing that revenue is roughly proportional to quantity). □

The paper outline is as follows. After introducing a formalism to manipulate cubes and queries in Section 2, in Section 3 we introduce models and enhanced cubes. In Section 4 we give the syntax of explain and illustrate how models are built. Then, in Section 5 we explain how enhanced cubes are visualized. Finally, in Section 6 we test the operator implementation in terms of efficiency, in Section 7 we discuss the related literature, and in Section 8 we draw the conclusions.

2 FORMALITIES

To simplify the formalization and without loss of generality,[2] we will restrict to consider linear hierarchies.

Definition 2.1 (Hierarchy and Cube Schema). A *hierarchy* is a triple $h = (L_h, \geq_h, \geq_h)$ where:

(i) L_h is a set of categorical *levels*, each coupled with a *domain* $Dom(l)$ including a set of *members*;
(ii) $\geq_h$ is a *roll-up* total order of L_h; and
(iii) $\geq_h$ is a *part-of* partial order of $\bigcup_{l \in L_h} Dom(l)$.

The top level of $\geq_h$ is called *dimension*. The part-of partial order is such that, for each couple of levels l and l' such that $l \geq_h l'$, for each member $u \in Dom(l)$ there is exactly one member $u' \in Dom(l')$ such that $u \geq_h u'$. A *cube schema* is a couple $C = (H, M)$ where:

(i) H is a set of hierarchies;
(ii) M is a set of numerical measures, where each measure $m \in M$ is coupled with one aggregation operator $op(m) \in \{\mathsf{sum}, \mathsf{avg}, \ldots\}$.

Example 2.2. For our working example we will use the SALES cube, whose conceptual schema is depicted in Figure 2 using the DFM [16]. Formally, it is SALES $= (H, M)$ with

$H = \{h_{\text{Date}}, h_{\text{Product}}, h_{\text{Store}}\}$;
$M = \{\text{quantity}, \text{revenue}, \text{cost}\}$;
date $\geq$ month $\geq$ year;
product $\geq$ type $\geq$ category;
store $\geq$ city $\geq$ country

[1] Since the term *correlation* in statistics is mainly used to denote linear relationships, to avoid misunderstandings we will use the general term *relationship* instead.

[2] The presence of branches and diamonds in the hierarchies only affects the definition of group-by sets and, consequently, the definition of roll-up partial order and the computation of derived cubes; it has no impact within the scope of this paper since we focus on regression-based models which operate at a fixed group-by set, the one stated in each intention.

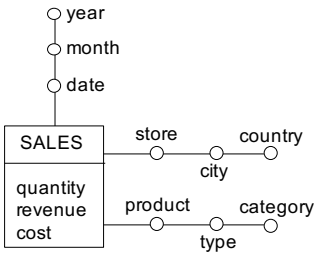


Figure 2: A conceptual schema for the SALES cube

and $op(\text{quantity}) = op(\text{revenue}) = op(\text{cost}) = \mathsf{sum}$. In the part-of order of the Product hierarchy it is, for instance, Orange $\geq_{\text{Product}}$ Fresh Fruit $\geq_{\text{Product}}$ Fruit. □

Aggregation is the basic mechanism to query cubes, and it is captured by the following definition of group-by set. As normally done when working with the multidimensional model, if a hierarchy h does not appear in a group-by set it is implicitly assumed that a complete aggregation is done along h.

Definition 2.3 (Group-by Set and Coordinate). Given cube schema $C = (H, M)$, a *group-by set* of C is a set of levels, at most one from each hierarchy of H. The partial order induced on the set of all group-by sets of C by the roll-up orders of the hierarchies in H, is denoted with $\geq_H$. A *coordinate* of group-by set G is a tuple of members, one for each level of G. Given coordinate γ of group-by set G, another group-by set G' such that $G \geq_H G'$, and the coordinate γ' of G' whose members are related to the corresponding members of γ in the part-of orders, we will say that γ *roll-up*s to γ'. Conventionally, each coordinate roll-ups to itself.

Example 2.4. Two group-by sets of SALES are $G_1 = \{\text{date}, \text{type}, \text{country}\}$ and $G_2 = \{\text{month}, \text{category}\}$, where $G_1 \geq_H G_2$. G_1 aggregates sales by date, product type, and store country, G_2 by month and category. Example of coordinates of the two group-by sets are, respectively, $\gamma_1 = \langle$2022-04-15, Fresh Fruit, Italy$\rangle$ and $\gamma_2 = \langle$2022-04, Fruit$\rangle$, where γ_1 roll-ups to γ_2. □

The instances of a cube schema are called cubes and are defined as follows.

Definition 2.5 (Cube). A *cube* over C is a triple $C = (G_C, M_C, \omega_C)$ where:

(i) G_C is a group-by set of C;
(ii) $M_C \subseteq M$;
(iii) ω_C is a partial function that maps the coordinates of G_C to a numerical value for each measure $m \in M_C$.

Each coordinate γ that participates in ω_C, with its associated measure values, is called a *fact* of C. With a slight abuse of notation, we will write $\gamma \in C$ to state that γ is a fact of C. The value taken by measure m in the fact corresponding to γ is denoted as $\gamma.m$. A cube whose group-by set G_C includes all and only the dimensions of the hierarchies in H and such that $M_C = M$, is called a *base cube*, the others are called *derived cubes*. In OLAP terms, a derived cube is the result of either a roll-up, a slice-and-dice, or a projection made over a base cube; this is formalized as follows.

Definition 2.6 (Cube Query). A *query* over cube schema C is a triple $q = (G_q, P_q, M_q)$ where:

(i) G_q is a group-by set of H;

(ii) P_q is a (possibly empty) set of selection predicates each expressed over one level of H using either a comparison operators (=, ≥, etc.) or the set inclusion operator (e.g., country in 'Italy', 'France');
(iii) $M_q \subseteq M$.

Let C_0 be a base cube over $\mathcal{C}$. The result of applying q to C_0 is a derived cube $C = q(C_0)$ such that (i) $G_C = G_q$, (ii) $M_C = M_q$, and (iii) ω_C assigns to each coordinate $\gamma \in C$ satisfying the conjunction of the predicates in P_q and to each measure $m \in M_C$ the value computed by applying $op(m)$ to the values of m for all the coordinates of C_0 that roll-up to γ.

Example 2.7. The cube query over SALES used in Example 1.1 is $q = (G_q, P_q, M_q)$ where $G_q = \{\text{type}\}$, $P_q = \{\text{year} = \text{'2022'}\}$, and $M_q = \{\text{revenue}\}$. A coordinate of the resulting cube is ⟨Batteries⟩ with associated value 3320 for revenue. □

3 ENHANCED CUBES

Models are concise, information-rich knowledge artifacts [39] that represent relationships hiding in the cube facts. The possible models range from simple functions and measure correlations to more elaborate techniques such as decision trees, clusterings, etc. A model is bound to (i.e., is computed over the levels/measures of) one cube, and is made of a set of components, each component being a specific relationship among cube facts.

Definition 3.1 (Model). A *model* is a tuple $\mathcal{M} = (t, alg, C, m, In, Out)$ where:

(i) t is the model type;
(ii) alg is the algorithm used to compute Out;
(iii) C is the cube to which the model is bound;
(iv) m is the measure in C to be explained;
(v) In is the tuple of levels/measures of C and parameter values supplied to alg to compute the model;
(vi) Out is the set of model components.

In this paper, to give a proof-of-concept of the explain operator, we restrict to consider a single type of model, namely, the one that establishes a polynomial relationship between two measures via regression analysis. In this case, In is the set of measures whose relationship with m is described; besides, each component $c_i \in Out$ shows the relationship of m with one measure $m_i \in In$.[3]

Definition 3.2 (Component). For a model of type polynomial, a *component* c_i is a triple $c_i = (m_i, d_i, coeff_i)$ where:

(i) m_i is the measure in C whose relationship with m is described;
(ii) d_i is the degree of the polynomial used to describe the relationship between m and m_i;
(iii) $coeff_i$ is an array of the $d_i + 1$ coefficients of the polynomial $\alpha^{d_i}(m_i)$ that best approximates m with reference to the facts in C.

[3]We represent each polynomial with a component of the same model rather than as a stand-alone model to be compatible with the IAM approach and with the formalization of the describe and assess operators.

Example 3.3. A possible model over the SALES cube is characterized by

$$t = \text{regression};$$
$$alg = \text{Polyfit};$$
$$C = \text{SALES};$$
$$m = \text{revenue};$$
$$In = \{\text{quantity}, \text{cost}\};$$
$$Out = \{c_1, c_2\}$$

where

$$c_1 = (\text{quantity}, 1, [0.98, 4909.52]);$$
$$c_2 = (\text{cost}, 2, [1.1, 22.78, 1409.33])$$

According to this model, the relationships of revenue with quantity and cost are described, respectively, as

$$\text{revenue} = \alpha^1(\text{quantity}) = 0.98 \cdot \text{quantity} + 4909.52$$
$$\text{revenue} = \alpha^2(\text{cost}) = 1.1 \cdot \text{cost}^2 - 22.78 \cdot \text{cost} + 1409.33$$

□

As the last step in the IAM approach, cube C is enhanced by associating it with a set of models bound to C and with a *highlight*, i.e., with the most interesting model component:

Definition 3.4 (Enhanced cube). An *enhanced cube* E is a triple of a cube C, a set of models $\{\mathcal{M}_1, \ldots, \mathcal{M}_z\}$ bound to C, and a highlight

$$\bar{c} = argmax_{\{c_i \in \bigcup_{j=1}^{z} Out_j\}}(interest(c_i))$$

In our scenario only polynomial models are considered, so an enhanced cube includes a single model with one component for each measure in In. Let c_i be the component associated to m_i; we evaluate the interest of c_i, $interest(c_i)$, as the *coefficient of determination* R2 [37], which measures how well the values of m are replicated by the model in m_i via the variation in the dependent variable m that is predictable from the independent variable m_i. The better the model, the closer the value of R2 to 1.

Example 3.5. With reference to Example 3.3, it is

$$interest(c_1) = 0.61$$
$$interest(c_2) = 0.99$$

Thus, the highlight is c_2.

4 THE EXPLAIN OPERATOR

The explain operator provides an answer to the user asking "why is this happening?" "why does measure m show these values?" by describing the relationship between m and the other cube measures, possibly focused on one or more level members, at some given granularity. The cube is enhanced by showing the polynomials that best approximate these relationships, with a highlight on the most interesting one.

4.1 Syntax

Let C_0 be a base cube over cube schema $\mathcal{C} = (H, M)$. The syntax for explain is (optional parts are in brackets):

with C_0 explain m
[for P] by $l_1, \ldots, l_n$
[against m_1 [degree d_1], …, m_r [degree d_r]]

where $m \in M$ is a measure of $\mathcal{C}$; P is a set of selection predicates, each over one level of H; $\{l_1, \ldots, l_n\}$ is a group-by set of H;

$m_1, \ldots, m_r$ are measures of M (different from m); the d_i's ($d_i > 0$) are integers denoting, for each m_i, the degree of the polynomial to be computed.

Example 4.1. Examples of explain intentions on the SALES cube are, besides the one in Example 1.1,

```
with SALES explain cost by date, product
                  against quantity degree 1
with SALES explain revenue by year
                  against cost
```

4.2 Semantics

The execution plan corresponding to a fully-specified intention, i.e., one where all optional clauses have been specified, is as follows:

1. Execute query $q = (G_q, P_q, M_q)$, where $G_q = \{l_1, \ldots, l_n\}$, $P_q = P$, and $M_q = \{m, m_1, \ldots, m_r\}$. Let $C = q(C_0)$ be the cube resulting from the execution of q over C_0.
2. Compute model $\mathcal{M} = (\text{polynomial}, \text{Polyfit}, C, m, \{m_1, \ldots, m_r\}, \{c_1, \ldots, c_r\})$, where $c_i = (m_i, d_i, coeff_i)$. The best approximating polynomial of degree d_i is determined via *ordinary least squares* [38], which finds —with a complexity of $O(d_i^2|C|)$, where $|C|$ is the number of facts in C— the polynomial coefficients that minimize the sum of squared errors between each m_i (independent variables) and m (dependent variable).
3. For each c_i compute $interest(c_i)$.
4. Find the highlight $\overline{c} = argmax_{\{1 \le i \le r\}}(interest(c_i))$.
5. Return the enhanced cube E consisting of C, $\{\mathcal{M}\}$, and $\overline{c}$.

Partially-specified intentions are interpreted as follows:

- If the for clause is not specified, we consider $P_q = TRUE$.
- If the against clause is not specified, a component is created for each measure in M (except m).
- If the degree clause is not specified for one or more measures, the value of d_i is determined automatically by polynomial fitting as discussed in Section 4.3.

Example 4.2. The first intention in Example 4.1 is executed by first computing the derived cube C that aggregates SALES by {date, product} and projects on measures cost and quantity. Then, a model $\mathcal{M}$ including a single component c (a linear polynomial approximating cost in function of quantity) is determined. Finally, the enhanced cube including C, $\mathcal{M}$, and the highlight c is returned.

4.3 Finding the optimal degree

Given two variables in a dataset, polynomial fitting (or simply *Polyfit*) summarizes their relationships by the polynomial function of the lowest degree that best approximates their values [28]. Finding the best polynomial function requires minimizing an error function that balances the approximation error and the polynomial degree (the higher the degree, the lower the error but the higher the overfitting).

In our scenario, the goal is to approximate m with a polynomial in m_i, and the dataset is the set of facts of cube C. Let α^d denote the polynomial of degree d in m_i that best approximates m; then, the fitting error (namely, the mean squared error) can be expressed in function of d as [3]

$$error(m, m_i, d) = \frac{\sum_{\gamma \in C}(\alpha^d(\gamma.m_i) - \gamma.m)^2}{|C| - d - 1}$$

where the γ's are the coordinates of C. Intuitively, this formula measures the average squared approximation error with a penalty on the degree d: among the polynomials with similar approximation errors, the one with the lowest degree is preferred[4].

To find the best degree d_i for each m_i we follow a step-wise forward-selection regression approach. We start with a constant polynomial, then we iteratively test the addition of higher-degree coefficients in the polynomial with a chosen fitness criterion (as suggested in [21]). Specifically, we divide the query result into train and test, with 70% and 30% facts respectively. We fit the polynomial to the training data, then we assess its $error()$ against the test set. We stop when, after reaching a good model, the error increases again; intuitively, we test how well the polynomial generalizes and we stop when higher-degree polynomials are overfitting the query result. Note that there is a possibility that a local minimum is reached by following this approach. For instance, when fitting quadratic data, a cubic polynomial α^3 could be worst than the quadratic one α^2; so the search would stop, while a quartic polynomial α^4 whose cubic and quartic terms tend to 0 might be (slightly) better than α^2. However, we argue that in this case a simple model should be preferred to a more complex one, i.e., to a polynomial with a higher degree.

To ensure that a polynomial is trained on a "sufficient" amount of facts, we apply the one-to-ten rule of thumb[5]: the polynomial with degree d is considered only if the query result contains at least $d \cdot 10$ tuples. The pseudocode is sketched in Algorithm 1. Given the measure m to be explained, we first initialize the approximation error (Line 1), the initial degree (Line 2), and the Boolean stop condition (Line 3); then, the iteration begins (Lines 4-12). We compute the best polynomial with the given degree d through ordinary least squares optimization (Line 5) and the error of the polynomial (Line 6). If the current error is better than the one obtained so far (Line 7), we update it (Line 8), increase the polynomial degree (Line 9), and continue with the iteration (Line 13). Otherwise, we terminate the iteration (Line 11). In any case, the iteration stops if $|C| < 10d$ (Line 13) [30]. Finally, we return the best polynomial (i.e., the one computed before the current iteration).

Example 4.3. The intention in Example 1.1,

```
with SALES explain revenue by type for year='2022'
```

is executed by first computing the derived cube C that aggregates sales by type for 2022. Then, the model seen in Example 3.3 (featuring two components, one for quantity and one for cost) is determined. To this end, since no degree clause is declared in the intention, the optimal degree must be determined both for quantity and one cost as in Algorithm 1. Algorithm 1 iteratively finds the following polynomials with degrees from 0 to 3 (see Figure 3):

$$\text{revenue} = 10737.6$$

$$\text{revenue} = 191.74 \cdot \text{cost} - 8098.16$$

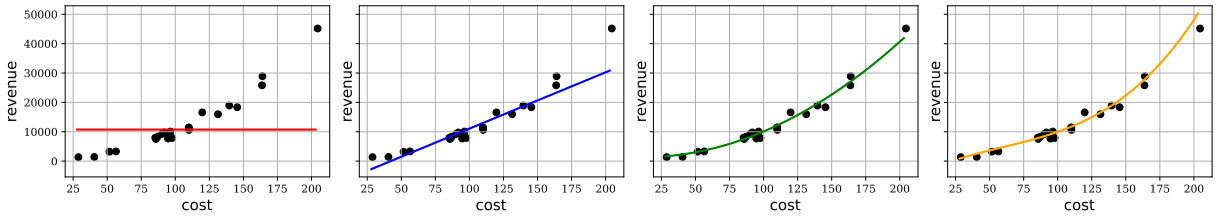
$$\text{revenue} = 1.1 \cdot \text{cost}^2 - 22.78 \cdot \text{cost} + 1409.33$$

$$\text{revenue} = 0.01 \cdot \text{cost}^3 - 1.73 \cdot \text{cost}^2 + 215.22 \cdot \text{cost} - 4027.88$$

As shown in Figure 4, the quadratic polynomial is returned since

[4] If $|C| \le d + 1$, no polynomial of degree d can be fitted, hence the error is not computed.

[5] "One to ten" or "one in ten" is a rule of thumb for how many parameters can be estimated from data when doing regression: a minimum of 10 observations per parameter is deemed necessary to avoid overfitting [30].



(a) Constant (b) Linear (c) Quadratic (d) Cubic

Figure 3: Approximating polynomials with degrees 0 (a), 1 (b), 2 (c), and 3 (d) for the cost measure from Example 4.3

Algorithm 1 Polyfit

Require: m: measure to explain; m_i: measure used for explanation
Ensure: α: optimal polynomial

```
e* ← +∞                                   ▷ Approximation error
d ← 0                                     ▷ Degree
stop ← False                              ▷ Stop condition
do
    ▷ Find the best polynomial of degree d...
    α^d ← OrdinaryLeastSquares(m, m_i, d)
    e ← error(m, m_i, d)                  ▷ ...and compute its error
    if e < e* then                        ▷ If a better approximation is found...
        e* ← e                            ▷ ...update the error, ...
        d ← d + 1                         ▷ ...increment the degree and iterate, ...
    else
        stop ← True                       ▷ ...otherwise stop
while !stop ∧ (|C| ≥ 10d)
return α^(d−1)                            ▷ Return the polynomial
```

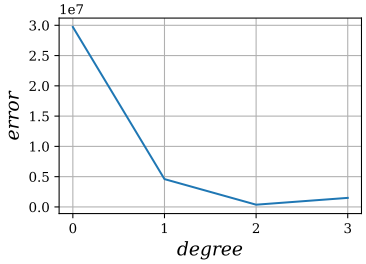


Figure 4: Error in function of the degree for the polynomials in Figure 3

the cubic one has a higher error.[6] □

5 VISUALIZING ENHANCED CUBES

As previously done for the describe and assess IAM operators, to give an effective visualization of the enhanced cubes built for explain intentions we couple a text-based representation (a pivot table and a ranked component list) with a graphical one (a chart) and with an ad-hoc interaction paradigm. Specifically, the visualization of enhanced cube $E = (C, \mathcal{M}, \bar{c})$ relies on three distinct but inter-related areas: a *table* area that shows the facts of C using a pivot table; a *component* area that shows a list of model components (i.e., approximating polynomials) sorted by their interest, with $\bar{c}$ at the top; a *chart* area that uses a scatter chart to display, for each component c_i of $\mathcal{M}$, the relationship between m and m_i as well as the function plotting the approximating polynomial.

The interaction paradigm we adopt is component-driven: clicking on one component c_i in the component area leads to show the corresponding approximating polynomial in the chart area. The highlight is selected by default.

Example 5.1. Figure 5 shows the visualization obtained when the intention in Example 1.1 is formulated. On the left, the table area; on the right, the chart area; in the middle, the component area. The highlight is a quadratic polynomial that approximates revenue in function of cost, so the chart area shows the relationship between these two measures and the approximating parabola.

[6]The polynomial of degree 3 resembles a parabola since the cubic term tends to 0, hence, its sum of squared errors is similar to the one of the polynomial of degree 2. However, the error function penalizes it due to the higher degree.

$\lvert C\rvert$	Complexity (numb. of char.)		Time (sec.)		
	Intention	Query	Model	Query	Total
36	59	397	0.31	0.04	0.35
323	42	265	0.32	0.03	0.35
540	61	403	0.32	0.05	0.37
1224	64	412	0.32	0.05	0.37
12113	60	400	0.34	0.05	0.39
16949	56	395	0.34	0.07	0.41
18492	55	385	0.35	0.06	0.41
20525	55	392	0.35	0.07	0.42
77832	54	382	0.40	0.07	0.47
86832	67	509	0.41	0.09	0.50

Table 1: Test results in function of the cube cardinality

6 EVALUATION

The prototype we developed to test our approach uses the simple multidimensional engine described in [8], which in turn relies on the MySQL DBMS to execute queries on a star schema based on multidimensional metadata (in principle, the prototype could work on top of any other multidimensional engine). The algorithms used for regression analysis are imported from the Scikit-Learn Python library. Finally, the web-based visualization is implemented in JavaScript and exploits the D3 library for chart visualization. The code is publicly available at https://github.com/big-unibo/explain.

To verify the feasibility of our approach from the computational point of view, we made some scalability tests. Two main factors affect performances: the cardinality of the cube to which a model is bound, $|C|$ (which determines the time required to compute a single model component), and the number of cube

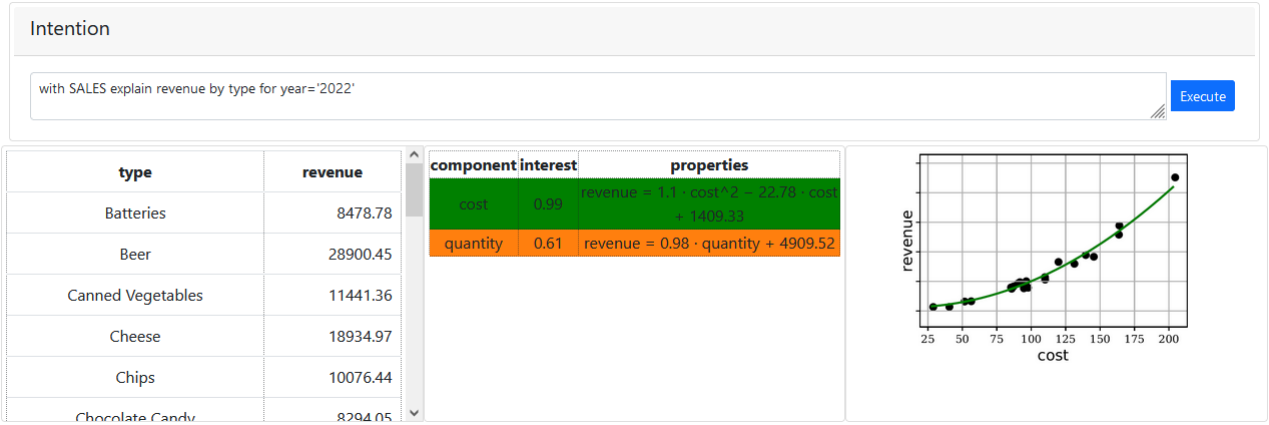


Figure 5: The visualization obtained for the intention in Example 1.1

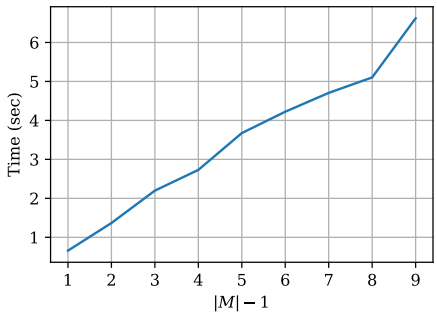


Figure 6: Scalability test

measures, $|M|$ (which determines the number of model components to be computed).

To evaluate scalability with reference to cube cardinality, we populated the SALES cube using the FoodMart data (https://github.com/julianhyde/foodmart-data-mysql) and considered 10 intentions with increasing cardinalities; in each intention we explained the revenue measure against both quantity and cost. Intentions were computed on cubes obtained by progressively including in the group-by set levels from the time, product, and customer hierarchies; for regression, polynomials up to the 5th degree were considered. The tests were run on an Intel(R) Core(TM)i7-6700 CPU@3.40GHz CPU with 8GB RAM; each intention was executed 10 times and the average results are reported. Table 1 shows the time (in seconds) necessary to query the base cube and to compute the models. Remarkably, it turns out that less than one second is necessary to explain a cube of almost 87000 facts.[7] Additionally, we measured the complexity (as the number of characters [19]) of writing explain intentions vs. the underlying cube query. It turns out that our approach saves 85% of complexity with respect to writing cube queries (and without considering the complexity of extracting regression models, which would make our approach even more convenient).

To evaluate scalability with reference to the number of measures, we created a cube with $|C| = 10^6$ facts and $|M| = 10$ measures (one randomly generated, m_0, and 9 more measures whose values we generated using polynomials in m_0 with increasing degrees). Figure 6 shows the performance when m_0 is explained against an increasing number of measures, up to $|M| - 1$. As expected, our approach scales linearly in the number of measures, and given 9 measures and 10^6 facts, the computation of an explanation takes less than 7 seconds, thus fulfilling the requirement of near-real-time response typical of analytical workloads.

[7] Since explain intentions are formulated over analytical workloads, cardinalities $|C|$ of OLAP query results in the order of 10^4 are already large enough to be considered unrealistic [11]).

7 RELATED WORK

7.1 OLAP + analytics

The idea of coupling data and analytical models was born in the 90's with inductive databases, where data were coupled with patterns meant as generalizations of the data [31]. Later on, data-to-model unification was addressed in MauveDB [7], which provides a language for specifying model-based views of data using common statistical models. However, achieving a unified view of data and models was still seen as a research challenge in business intelligence a few years later [29]. More recently, Northstar [20] has been proposed as a system to support interactive data science by enabling users to switch between data exploration and model building, adopting a real-time strategy for hyper-parameter tuning. Finally, the coupling of data and models is at the core of the IAM vision [40], on which this paper relies. The three basic pillars of IAM are (i) the redefinition of query as expressing the user's intention rather than explicitly declaring what data are to be retrieved, (ii) the extension of query results from plain data cubes to cubes enhanced with models and highlights, and (iii) the characterization of model components in terms of their interest to users.

The coupling of the OLAP paradigm and data mining to create an approach where concise patterns are extracted from multi-dimensional data for user's evaluation, was the goal of some approaches commonly labeled as OLAM [18]. In this context, k-means clustering is used in [2] to dynamically create semantically-rich aggregates of facts other than those statically provided by dimension hierarchies. Similarly, the shrink operator is proposed in [15] to compute small-size approximations of a cube via agglomerative clustering. Other operators that enrich data with knowledge extraction results are DIFF [34], which returns a set of tuples that most successfully describe the difference of values between two facts of a cube, and RELAX [35], which verifies whether a pattern observed at a certain level of detail is also

present at a coarser level of detail, too. Finally, in [5] the OLAP paradigm is reused to explore prediction cubes, i.e., cubes where each fact summarizes a predictive model trained on the data corresponding to that fact.

7.2 Query explanation

In an attempt to develop tools for helping users understand data, there have been several efforts in the research community to devise techniques to model explanations for observations made on data [25]. See [14] for a comprehensive analysis of the literature and of the trends in explanation.

A common way to give an explanation is to identify the actual *cause* of the observed outcome [33]. Given the result of a database query, which database tuple(s) caused that output to the query? One way to answer this question is to quantify the contribution that each tuple has to the result and identify the tuples with the highest contributions [23, 24]; the intuition is that tuples with high contribution tend to be interesting explanations to query answers. Similarly, in [33] causality is defined in terms of *intervention*: an input is a cause to an output if we can affect the output by changing the value of that input. Thus, an explanation is defined as a predicate such that, when we remove from the database all tuples satisfying that predicate, the output is significantly affected. Along this direction, techniques were devised to make the search for explanations more efficient by precomputing the effects of potential explanations [32] or to return more specific explanations concerning subgroups of answers determined via clustering [27]. Other approaches to query explanation rely on ontologies [6, 41].

Causality poses additional challenges when the query contains aggregates [23], as in our scenario. The DIFF operator [34] tells users why a given aggregated quantity is lower or higher in one cube fact than in another by returning the set of rows that best explains the observed increase or decrease at the aggregated level. In Scorpion [42], outliers are explained in terms of properties of the tuples used to compute these outliers. Specifically, this explanation determines the predicates that, when applied to the input data, cause the outliers to disappear. LensXPlain [26] explains why some measure value is high or low by identifying subsets of facts that contributed the most toward such observation. The contributions are measured either by *intervention* (if the contributing facts are removed, the value changes in the opposite direction), or by *aggravation* (if only the contributing facts are kept, the value changes more in the same direction).

A different approach to query explanation is taken in [13]. The authors focus on multidimensional data where a binary dimension is present, and explain query results by building *explanation tables* which provide an interpretable and informative summary of the factors affecting the binary dimension.

7.3 Regression

A completely different direction to represent how some data (measures, in our case) is derived and infer causal relationship is to use models built by *regression analysis* [36]. In statistical modeling, regression analysis is a set of statistical processes for estimating the relationships between a dependent variable and one or more independent variables. A common form of regression analysis is *polynomial regression*, which we adopt in this paper; although polynomial regression may use a non-linear model (e.g., a parabola) to fit the data, as a statistical estimation problem it is considered to be linear, since the regression function is linear in the unknown parameters that are estimated from the data. The method we use for polynomial regression is *ordinary least squares*, which computes the unique line (or hyperplane) that minimizes the sum of squared differences between the true data and that line (or hyperplane) [38].

Regression is used to explain query results in the XAXA approach [36]. The authors focus on aggregate queries with a center-radius selection operator, and give explanations using a set of parametric piecewise-linear functions acquired through a statistical learning model. Remarkably, model training is performed by only monitoring queries and their answers online; thus, explanations for future queries can be computed without any database access.

7.4 Discussion

The approach we propose is not competing with the ones mentioned above, but should rather be seen as a modular framework where any approach to explanation of aggregate data could be plugged. The added value lies in the IAM paradigm, i.e., in giving users the possibility of explicitly expressing intentions, in letting the system select the most interesting/suitable explanations, and showing these explanations together with data.

8 CONCLUSION

In this paper we have given a proof-of-concept for explain intentions formulated inside the IAM framework. The explain syntax is flexible enough to suit users who wish to verify a specific hypothesis they made about an inter-measure relationship, as well as users who have no clue so they will let the system find the most interesting relationship. Intention processing takes a few seconds even on very large query results, thus performances are perfectly in line with the interactivity requirements of OLAP sessions.

The main directions for future research we wish to pursue are: (i) evaluate the effectiveness of the approach by experimenting it with real users; (ii) shift towards multivariate regression models to explain relationships between one measure and two or more other measures; (iii) generalize the definition of model to cope with the additional model types mentioned in Section 1; (iv) extend the syntax to allow two or more cubes in the with clause (so as to support drill-across queries) and the declaration of derived measures in the explain clause; and (v) experiment other interest metrics [4]. In particular, as to the last point, we plan to consider the framework proposed in [14] to evaluate explanations in terms of *succinctness* (large explanations will probably we not well understandable), *interpretability* (the suitability of an explanation will depend on the target users), and *actionability* (explanations should point to actionable suggestions).

REFERENCES

[1] Charu C. Aggarwal, Mansurul A Bhuiyan, and Mohammad Al Hasan. 2014. Frequent pattern mining algorithms: A survey. In *Frequent pattern mining*, Charu C. Aggarwal and Jiawei Han (Eds.). Springer, 19–64.

[2] Fadila Bentayeb and Cécile Favre. 2009. RoK: Roll-Up with the K-Means Clustering Method for Recommending OLAP Queries. In *Proceedings of DEXA*. Linz, Austria, 501–515.

[3] Peter J. Bickel and Kjell A. Doksum. 2015. *Mathematical statistics: basic ideas and selected topics, volumes I-II package*. Chapman and Hall/CRC.

[4] Alexandre Chanson, Ben Crulis, Krista Drushku, Nicolas Labroche, and Patrick Marcel. 2019. Profiling User Belief in BI Exploration for Measuring Subjective Interestingness. In *Proceedings of DOLAP*. Lisbon, Portugal, 1–9.

[5] Bee-Chung Chen, Lei Chen, Yi Lin, and Raghu Ramakrishnan. 2005. Prediction Cubes. In *Proceedings of VLDB*. Trondheim, Norway, 982–993.

[6] Federico Croce and Maurizio Lenzerini. 2018. A Framework for Explaining Query Answers in DL-Lite. In *Proceedings of EKAW*. Nancy, France, 83–97.

[7] Amol Deshpande and Samuel Madden. 2006. MauveDB: supporting model-based user views in database systems. In *Proceedings of SIGMOD*. Chicago, IL, USA, 73–84.

[8] Matteo Francia, Enrico Gallinucci, and Matteo Golfarelli. 2022. COOL: A framework for conversational OLAP. *Inf. Syst.* 104 (2022), 101752. https://doi.org/10.1016/j.is.2021.101752

[9] Matteo Francia, Matteo Golfarelli, Patrick Marcel, Stefano Rizzi, and Panos Vassiliadis. 2021. Assess Queries for Interactive Analysis of Data Cubes. In *Proceedings of EDBT*. Nicosia, Cyprus, 121–132.

[10] Matteo Francia, Matteo Golfarelli, Patrick Marcel, Stefano Rizzi, and Panos Vassiliadis. 2023. Suggesting assess queries for interactive analysis of multidimensional data. *IEEE Trans. Knowl. Data Eng., to appear* (2023).

[11] Matteo Francia, Matteo Golfarelli, and Stefano Rizzi. 2020. A-BI$^+$: A framework for Augmented Business Intelligence. *Inf. Syst.* 92 (2020), 101520. https://doi.org/10.1016/j.is.2020.101520

[12] Matteo Francia, Patrick Marcel, Verónika Peralta, and Stefano Rizzi. 2022. Enhancing Cubes with Models to Describe Multidimensional Data. *Inf. Syst. Frontiers* 24, 1 (2022), 31–48.

[13] Kareem El Gebaly, Parag Agrawal, Lukasz Golab, Flip Korn, and Divesh Srivastava. 2014. Interpretable and Informative Explanations of Outcomes. *Proceedings of VLDB Endow.* 8, 1 (2014), 61–72.

[14] Boris Glavic, Alexandra Meliou, and Sudeepa Roy. 2021. Trends in Explanations: Understanding and Debugging Data-driven Systems. *Found. Trends Databases* 11, 3 (2021), 226–318.

[15] Matteo Golfarelli, Simone Graziani, and Stefano Rizzi. 2014. Shrink: An OLAP operation for balancing precision and size of pivot tables. *Data Knowl. Eng.* 93 (2014), 19–41.

[16] Matteo Golfarelli and Stefano Rizzi. 2009. *Data Warehouse design: Modern principles and methodologies.* McGraw-Hill.

[17] Clive W. J. Granger. 1969. Investigating causal relations by econometric models and cross-spectral methods. *Econometrica* 37, 3 (1969), 424–438.

[18] Jiawei Han. 1997. OLAP Mining: Integration of OLAP with Data Mining. In *Proceedings of Working Conf. on Database Semantics*. Leysin, Switzerland, 3–20.

[19] Shrainik Jain, Dominik Moritz, Daniel Halperin, Bill Howe, and Ed Lazowska. 2016. SQLShare: Results from a Multi-Year SQL-as-a-Service Experiment. In *Proceedings of SIGMOD*. San Francisco, CA, USA, 281–293.

[20] Tim Kraska. 2018. Northstar: An Interactive Data Science System. *Proceedings of VLDB Endow.* 11, 12 (2018), 2150–2164.

[21] Jonathan Mark and Michael A Goldberg. 1988. Multiple regression analysis and mass assessment: A review of the issues. *Appraisal Journal* 56, 1 (1988).

[22] G. Randolph Mayes. 2018. Theories of Explanation. *Internet Encyclopedia of Philosophy* (2018).

[23] Alexandra Meliou, Wolfgang Gatterbauer, Joseph Y. Halpern, Christoph Koch, Katherine F. Moore, and Dan Suciu. 2010. Causality in Databases. *IEEE Data Eng. Bull.* 33, 3 (2010), 59–67.

[24] Alexandra Meliou, Wolfgang Gatterbauer, Katherine F. Moore, and Dan Suciu. 2010. The Complexity of Causality and Responsibility for Query Answers and non-Answers. *Proceedings of VLDB Endow.* 4, 1 (2010), 34–45.

[25] Alexandra Meliou, Sudeepa Roy, and Dan Suciu. 2014. Causality and Explanations in Databases. *Proceedings of VLDB Endow.* 7, 13 (2014), 1715–1716.

[26] Zhengjie Miao, Andrew Lee, and Sudeepa Roy. 2019. LensXPlain: Visualizing and Explaining Contributing Subsets for Aggregate Query Answers. *Proceedings of VLDB Endow.* 12, 12 (2019), 1898–1901.

[27] Aurélien Moreau, Olivier Pivert, and Grégory Smits. 2015. A Clustering-Based Approach to the Explanation of Database Query Answers. In *Proceedings of FQAS*. Cracow, Poland, 307–319.

[28] Eva Ostertagová. 2012. Modelling using polynomial regression. *Procedia Engineering* 48 (2012), 500–506.

[29] Torben Bach Pedersen. 2009. Warehousing The World: A Vision for Data Warehouse Research. In *New Trends in Data Warehousing and Data Analysis*, Stanislaw Kozielski and Robert Wrembel (Eds.). Annals of Information Systems, Vol. 3. Springer, 1–17.

[30] Peter Peduzzi, John Concato, Elizabeth Kemper, Theodore R Holford, and Alvan R Feinstein. 1996. A simulation study of the number of events per variable in logistic regression analysis. *Journal of clinical epidemiology* 49, 12 (1996), 1373–1379.

[31] Luc De Raedt. 2002. A Perspective on Inductive Databases. *SIGKDD Explorations* 4, 2 (2002), 69–77.

[32] Sudeepa Roy, Laurel J. Orr, and Dan Suciu. 2015. Explaining Query Answers with Explanation-Ready Databases. *Proceedings of VLDB Endow.* 9, 4 (2015), 348–359.

[33] Sudeepa Roy and Dan Suciu. 2014. A formal approach to finding explanations for database queries. In *Proceedings of SIGMOD*. Snowbird, UT, USA, 1579–1590.

[34] Sunita Sarawagi. 1999. Explaining Differences in Multidimensional Aggregates. In *Proceedings of VLDB*. Edinburgh, Scotland, 42–53.

[35] Gayatri Sathe and Sunita Sarawagi. 2001. Intelligent Rollups in Multidimensional OLAP Data. In *Proceedings of VLDB*. Rome, Italy, 531–540.

[36] Fotis Savva, Christos Anagnostopoulos, and Peter Triantafillou. 2018. Explaining Aggregates for Exploratory Analytics. In *Proceedings of BigData*. Seattle, WA, USA, 478–487.

[37] Robert G. D. Steel and James H. Torrie. 1960. *Principles and procedures of statistics, with special reference to the biological sciences.* McGraw-Hill, New York.

[38] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. 2016. *Introduction to data mining.* Pearson Education India.

[39] Manolis Terrovitis, Panos Vassiliadis, Spiros Skiadopoulos, Elisa Bertino, Barbara Catania, Anna Maddalena, and Stefano Rizzi. 2007. Modeling and language support for the management of pattern-bases. *Data Knowl. Eng.* 62, 2 (2007), 368–397.

[40] Panos Vassiliadis, Patrick Marcel, and Stefano Rizzi. 2019. Beyond Roll-Up's and Drill-Down's: An Intentional Analytics Model to Reinvent OLAP. *Information Systems* 85 (2019), 68–91.

[41] Zhe Wang, Mahsa Chitsaz, Kewen Wang, and Jianfeng Du. 2015. Towards Scalable and Complete Query Explanation with OWL 2 EL Ontologies. In *Proceedings of CIKM*. Melbourne, Australia, 743–752.

[42] Eugene Wu and Samuel Madden. 2013. Scorpion: Explaining Away Outliers in Aggregate Queries. *Proceedings of VLDB Endow.* 6, 8 (2013), 553–564.