Information Systems 121 (2024) 102338

Contents lists available at ScienceDirect

Information Systems

journal homepage: www.elsevier.com/locate/is

Explaining cube measures through Intentional Analytics

Matteo Francia [a], Stefano Rizzi [a,*], Patrick Marcel [b]

[a] DISI, University of Bologna, Italy
[b] LIFO, University of Orléans, France

ARTICLE INFO

Keywords:
Data cube
OLAP
Analytics
Correlation

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 consider models that explain m in terms of one or more other measures. We propose a syntax for the operator and discuss how enhanced cubes are built by (i) finding the relationship between m and the other cube measures via regression analysis and cross-correlation, and (ii) highlighting the most interesting one. Finally, we test the operator implementation in terms of efficiency and effectiveness.

1. Introduction

Despite the OLAP (On-Line Analytical Processing) paradigm's enormous success in helping decision makers analyze multidimensional cubes, it is now obvious that this paradigm cannot, by itself, satisfy the sophisticated needs of new-generation users. The *Intentional Analytics Model* (IAM) suggests pairing OLAP with analytics as one of the approaches adopted by research to improve OLAP [1]. The two basic tenets of the IAM are: (i) users explore the data space by expressing their analysis *intentions*, and (ii) they obtain multidimensional data as well as knowledge insights in the form of models as a result. 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 Fig. 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 [2–4]. 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 [5]. In our context, we concentrate on providing explanation models for a measure the user is observing (*target measure*); thus, the goal of the explain operator will be to provide an answer to the user asking "why does measure m show these values?".

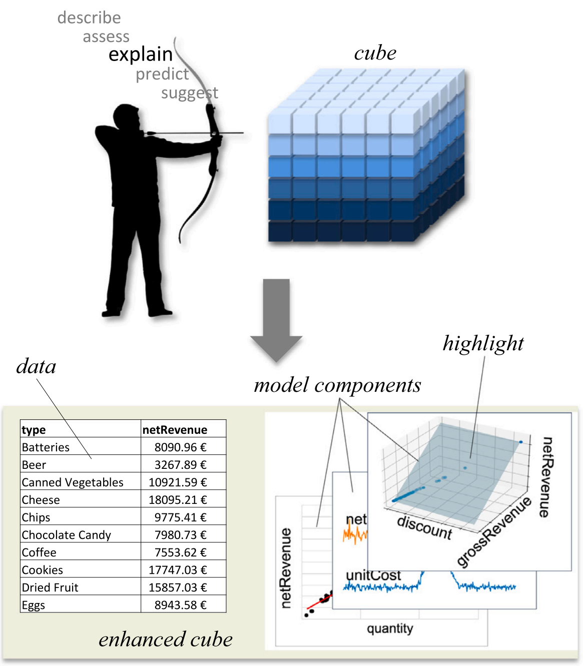
As envisioned in [1], several types of models can be used to this end, for instance:

- use regression analysis to correlate the values taken by m with those taken by one or more other measures m', m'', etc. (e.g., sales revenues are roughly proportional to the quantity sold);
- use cross-correlation to match a time series of m with one of another measure m', by also considering that there may be a delay between the two (e.g., the trend of deaths for a disease follows the one of infections with a 2-weeks delay);
- 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 [6] (e.g., the sales of panettone are always high in December);
- find the cube facts that give the highest contributions to m [7].

In this work we focus on models that explain m in terms of one or more other measures (*candidate measures*). While in a previous paper [8] we

* Corresponding author.
E-mail addresses: m.francia@unibo.it (M. Francia), stefano.rizzi@unibo.it (S. Rizzi), patrick.marcel@univ-tours.fr (P. Marcel).

https://doi.org/10.1016/j.is.2023.102338
Received 25 July 2023; Received in revised form 14 December 2023; Accepted 15 December 2023
Available online 18 December 2023



type	netRevenue
Batteries	8090.96 €
Beer	3267.89 €
Canned Vegetables	10921.59 €
Cheese	18095.21 €
Chips	9775.41 €
Chocolate Candy	7980.73 €
Coffee	7553.62 €
Cookies	17747.03 €
Dried Fruit	15857.03 €
Eggs	8943.58 €

Fig. 1. The IAM approach.

have only considered models that establish a polynomial relationship[1] between m and another measure m', here we extend our approach by also including multivariable linear regression and cross-correlation.

Example 1. Let a SALES cube be given, whose schema is shown in Fig. 2, and let the user's intention be

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

where netRevenue is the target measure and all the other measures of SALES (namely, quantity, unitPrice, grossRevenue, and discount) are candidate to be used for explanation). Fig. 1 shows the result of this intention, evaluated as follows. First, the subset of facts for 2022 (for clause) are selected from the SALES cube (with clause) and aggregated by product type (by clause; in OLAP terms, a slice-and-dice and a roll-up operator are applied). Then, regression analysis is used to compare the netRevenue measure with the candidate measures and find a set of components corresponding to (i) polynomials that best approximates the relationship of netRevenue with *each* candidate measure, (ii) the linear combination that best approximates the relationship of netRevenue with *all* candidate measures, and (iii) the best cross-correlation of netRevenue with *each* candidate measure. Finally, a measure of interest that expresses how well the values of netRevenue are replicated by each component is computed for all the components obtained, and the most interesting one (i.e., the highlight) is shown to the user. In the SALES cube, measure netRevenue is actually calculated as grossRevenue − discount; thus, not surprisingly, in Fig. 1 the highlight shows that netRevenue is a linear combination of grossRevenue and discount. □

The original contributions we give in this paper compared to [8] are listed below:

1. Besides polynomial univariable regression, we also consider models based on multivariable linear regression and cross-correlation.
2. For each model type, we give a definition of interest.
3. We formalize a join operator between cubes (in the same direction of the drill-across OLAP operator), aimed at making more candidate measures available for explanation and creating more precise models.
4. We extend the syntax of explain to cope with the new model types considered, to operate on two or more cubes, and to support derived measures, i.e., measures computed from other measures via algebraic expressions.
5. We define domination rules to cope with overlapping components of different models.
6. We present the results of a comprehensive set of experimental tests aimed to evaluate our approach not only from the point of view of efficiency, but also from that of effectiveness.

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 of the different types 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 and effectiveness, 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 1 (*Hierarchy and Cube Schema*)**.** A *hierarchy* is a triple $h = (L_h, \succeq_h, \geq_h)$ where:

(i) L_h is a set of categorical *levels*, each coupled with a *domain* $Dom(l)$ including a set of *members*;
(ii) $\succeq_h$ 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 $\succeq_h$ is called *dimension*. The part-of partial order is such that, for each couple of levels l and l' such that $l \succeq_h l'$, for each member $u \in Dom(l)$ there is exactly one member $u' \in Dom(l')$ such that $u \geq_h u'$.

Definition 2 (*Cube Schema*)**.** 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 $op(m) \in \{\mathtt{sum}, \mathtt{avg}, \dots\}$.

Example 2. For our working example we will use the SALES and PURCHASE cubes, whose conceptual schemata are depicted in Fig. 2 using the DFM [9]. Formally, it is SALES = (H, M) with

$H = \{h_{\text{Date}}, h_{\text{Product}}, h_{\text{Store}}\}$;

$M = \{\text{quantity}, \text{unitPrice}, \text{grossRevenue}, \text{discount}, \text{netRevenue}\}$;

date $\succeq$ month $\succeq$ year;

product $\succeq$ type $\succeq$ category;

store $\succeq$ city $\succeq$ country

and $op(\text{quantity}) = op(\text{grossRevenue}) = op(\text{discount}) = op(\text{netRevenue}) = \mathtt{sum}$, $op(\text{unitPrice}) = \mathtt{avg}$. In the part-of order of the Product hierarchy

[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 cube queries; it has no impact within the scope of this paper since we focus on models that operate at a fixed group-by set, the one stated in each intention.

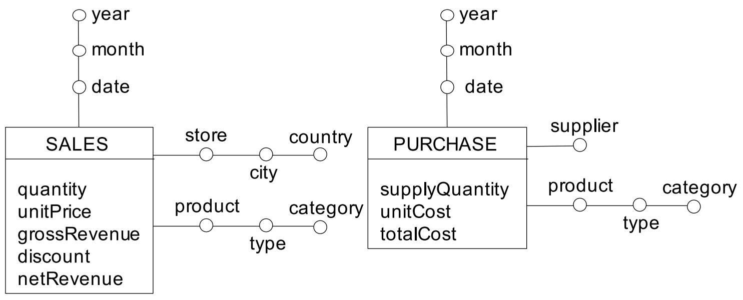


Fig. 2. Conceptual schemata for the SALES and PURCHASE cubes.

it is, for instance, Orange $\geq_{\text{Product}}$ Fresh Fruit $\geq_{\text{Product}}$ Fruit. The PURCHASE cube is similar, except that it has a Supplier hierarchy instead of Store. □

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 3 (*Group-by Set and Coordinate*)**.** Given cube schema $\mathcal{C} = (H, M)$, a *group-by set* of $\mathcal{C}$ 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 $\mathcal{C}$ by the roll-up orders of the hierarchies in H, is denoted with $\succeq_H$. 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 $G \succeq_H G'$, 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-ups* to γ'. Conventionally, each coordinate roll-ups to itself.

Example 3. Two group-by sets of SALES are $G^1 = \{\text{date}, \text{type}, \text{country}\}$ and $G^2 = \{\text{month}, \text{category}\}$, where $G^1 \succeq_H G^2$. 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 4 (*Cube*)**.** A *cube* over $\mathcal{C}$ is a triple $C = (G_C, M_C, \omega_C)$ where:

(i) G_C is a group-by set of $\mathcal{C}$;
(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 tuple of 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 $C^\top$ whose group-by set is the top of $\succeq_H$ (i.e., it is the finest group-by set of $\mathcal{C}$) and such that $M_C = M$, is called a *base cube*.

Definition 5 (*Cube Query*)**.** A *query* over cube schema $\mathcal{C}$ 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 $C^\top$ be a cube over $\mathcal{C}$. The result of applying q to $C^\top$ is a cube $C = q(C^\top)$ 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 $C^\top$ that roll-up to γ.

Example 4. The cube query over SALES used in Example 1 is $q = (G_q, P_q, M_q)$ where $G_q = \{\text{type}\}$, $P_q = \{\text{year} = \text{'2022'}\}$, and $M_q = \{\text{netRevenue}\}$. Let SALES$_1$ be the resulting cube; a coordinate of this cube is $\langle$Batteries$\rangle$ with associated value €8090.96 for netRevenue. □

To let our explain operator search for measure relationships across two or more related cubes, simulating the drill-across OLAP operator, we give a definition of cube *joinability*. Intuitively, two cubes are joinable if they share, either completely or partially, at least one hierarchy.[3] To simplify the definition we assume that a hierarchy cannot be partially shared; for discussion of how to cope with partially overlapped hierarchies (e.g., date ≥ month ≥ year and date ≥ year) we refer the reader to [10].

Definition 6 (*Joinability and Join*)**.** Let $C_1, \ldots, C_v$ be v cubes over cube schemata $\mathcal{C}_1, \ldots, \mathcal{C}_v$, respectively, be given, with $\mathcal{C}_i = (H_i, M_i)$. We say these cubes are *joinable* if $\bigcap_{i=1}^{v} H_i \neq \emptyset$, i.e., they share at least one hierarchy. The cube C resulting from *join* between these cubes, denoted $\bigwedge_{i=1}^{v} C_i$, has schema

$$\mathcal{C} = (\bigcap_{i=1}^{v} H_i, \bigcup_{i=1}^{v} M_i)$$

Let $G^\top$ be the finest group-by set of $\mathcal{C}$, and $q_i = (G^\top, TRUE, M_i)$ for $i = 1, \ldots, v$ be the queries that aggregate each cube C_i at $G^\top$. The coordinates of C are the intersection of the coordinates of cubes $q_1(C_1), \ldots, q_v(C_v)$, i.e., the common coordinates of $C_1, \ldots, C_v$ aggregated at $G^\top$; each coordinate of C is associated with all the measure values associated to the corresponding coordinates of the C_i's.

Intuitively, the schema of the join C of two or more cubes features the intersection of their hierarchies and the union of their measures, so its group-by set is the finest common group-by set and its measures values, for each coordinate, are those of the corresponding coordinates in the joined cubes.

Example 5. Cubes SALES and PURCHASE are joinable; their join, SP = SALES ∧ PURCHASE, features the Date and Product hierarchies and has measures quantity, unitPrice, … totalCost (see Fig. 3). The finest group-by set of SP is $G^\top = \{\text{date}, \text{product}\}$; an example coordinate of SP is $\gamma = \langle$2022-04-15, Orange$\rangle$. □

[3] Note that this definition of cube joinability is similar to the one given in [4], but more general since it does not require that the group-by of one cube is coarser than the one of the other cube.

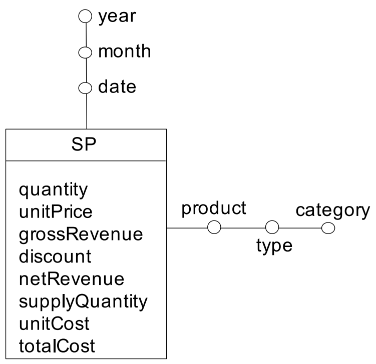


Fig. 3. Conceptual schema for the cube resulting from the join of SALES and PURCHASE.

3. Enhanced cubes

Models are concise, information-rich knowledge artifacts [11] that represent relationships hiding in the cube facts. 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. To make our approach more flexible, in this paper we give users the possibility of working with *derived measures*, i.e., measures computed on-the-fly from other measures via an algebraic expression. In the following, the term "measure" will be used in a general way to also include derived measures.

Definition 7 (*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 (possibly resulting from a join);
(iv) m is the target measure of C;
(v) In is the set of r candidate measures of C supplied to alg to compute the model;
(vi) Out is the set of model components.

In this paper we consider three types of models, namely:

- *Polynomial regression*, which establishes a polynomial relationship between m and one other measure via regression analysis. There are r components; each component $c_i \in Out$ shows the relationship of the target measures m with one candidate measure $m_i \in In$.
- *Multivariable linear regression*, which establishes a linear relationship between m and a set of other measures. The model includes exactly one component $c \in Out$ showing the relationship of m with all candidate measures in In.
- *Cross-correlation*, which finds the similarity of two series as a function of the displacement of one relative to the other. There are r components; each component $c_i \in Out$ shows the relationship of m with one candidate measure $m_i \in In$.

The form taken by components depends on the model type as follows.

Definition 8 (*Component*). For t = polynomialRegression, a component c_i is a triple $c_i = (m_i, d_i, coeff_i)$ where:

(i) m_i is the candidate measure;
(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.

For t = multivariableRegression, there is a single component $c = coeff$, where:

(i) $coeff$ is an array of the $r+1$ coefficients of the linear polynomial $\alpha^1(m_1, \dots, m_r)$ that best approximates m with reference to the facts in C.

Finally, for t = crossCorrelation, a component c_i is a couple $c_i = (m_i, disp_i)$ where:

(i) m_i is the candidate measure;
(ii) $disp_i$ is the displacement yielding maximum correlation between the series of values of m and the one of m_i.

Example 6. A possible polynomial regression model over the SALES$_1$ cube computed in Example 4 is characterized by

t = polynomialRegression; alg = Polyfit; C = SALES$_1$;

m = netRevenue; In = {quantity, discount}; $Out = \{c_1, c_2\}$

where

c_1 = (quantity, 1, [2.15, −171.88]);

c_2 = (discount, 1, [19.00, 27.42])

According to this model, the relationships of netRevenue with quantity and discount are described, respectively, as

netRevenue = α^1(quantity) = 2.15 · quantity − 171.88

netRevenue = α^1(discount) = 19.00 · discount + 27.42

An example of multivariable linear regression model over the same cube is

t = multivariableRegression; alg = MultiReg; C = SALES$_1$;

m = netRevenue; In = {grossRevenue, discount, quantity}; $Out = \{c_3\}$

$c_3 = [1.0, -1.0, 0.0, 0.0]$

According to this model, the relationships of netRevenue with grossRevenue, discount, and quantity is described as

α^1(grossRevenue, discount, quantity) = grossRevenue − discount

Finally, an example of cross-correlation model over the SP$_1$ cube, obtained by joining SALES and PURCHASE (which produces cube SP in Example 5) and then grouping by date, is

t = crossCorrelation; alg = CrossCorr; C = SP$_1$;

m = unitPrice; In = {unitCost}; $Out = \{c_4\}$

c_4 = (unitCost, 27)

According to this model, the daily trend of the average unitPrice is displaced by 27 days with respect to that of the average unitCost. □

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

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

Function $interest()$ measures the interest of each component on a continuous scale from 0 to 1; how this is done is the subject of Section 4.6.

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 these relationships, with a highlight on the most interesting one.

4.1. Syntax

Let $C_1^\top, \dots, C_p^\top$ be p joinable cubes, C_0 be their join, and $C = (H, M)$ be the schema of C_0. The syntax for explain is (optional parts are in brackets):

with $C_1^\top, \dots, C_p^\top$ explain $expr_0$ [as m]

by $l_1, \dots, l_n$ [for P]

[against $expr_1$ [as m_1], …, $expr_r$ [as m_r]]

[using $t_1, \dots, t_z$]

[range b]

where each $expr_i$ is an algebraic expression involving one or more measures in C; P is a set of selection predicates, each expressed on one level of H; $\{l_1, \dots, l_n\}$ is a group-by set of H; $t_i \in$ {polynomialRegression, multivariableRegression, crossCorrelation} is a model type; b is a positive integer. The different clauses take the following roles:

- The with clause specifies the cubes(s) on which the intention is executed.
- The explain clause specifies the target measure.
- The by clause specifies how the cube(s) must be aggregated.
- The for clause specifies a selection on the cubes(s).
- The against clause specifies the candidate measures.
- The as clause gives names to derived measures specified via expressions.
- The using clause specifies which model types are to be computed.
- The range clause specifies the maximum displacement allowed for cross-correlation.

Model type crossCorrelation can be computed only when the by clause includes exactly one level l, and l has type interval (e.g., a date).[4] Model type multivariableRegression is computed only when the against clause includes more than one candidate measure.

Example 7. Two examples of explain intentions on the SALES cube are, besides the one in Example 1,

with SALES, PURCHASE explain unitPrice by date

against unitCost using crossCorrelation range 60

with SALES explain (grossRevenue − netRevenue) as diff by year

The first one leads to the computation of the cross-correlation model in Example 6.

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[5]:

1. If $p > 1$, i.e., two or more cubes are specified in the with clause, compute the cube C_0 resulting from their join.
2. Execute query $q = (G_q, P_q, M_q)$ over C_0, where $G_q = \{l_1, \dots, l_n\}$, $P_q = P$, and $M_q = \{m, m_1, \dots, m_r\}$. Let $C = q(C_0)$ be the cube resulting from the execution of q over C_0.
3. For $1 \le j \le z$, compute model $\mathcal{M}_j = (t_j, alg_j, C, m, \{m_1, \dots, m_r\}, Out_j)$; see Sections 4.3, 4.4, and and 4.5 for a description of how each model is computed and its components are determined.
4. For each $c \in Out_j$ compute $interest(c)$. Essentially, the interest of component c measures how well c can replicate the values of m; see Section 4.6 for an explanation of how this is done for the different model types.
5. Find the highlight $\bar{c} = argmax_{c \in \bigcup_j Out_j}(interest(c))$.
6. Return the enhanced cube E consisting of C, $\{\mathcal{M}_1, \dots, \mathcal{M}_z\}$, and $\bar{c}$.

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, models are created for each measure in M (except m).
- If the using clause is not specified, all three model types are considered.

Example 8. The second intention in Example 7 is executed by first computing the cube C that aggregates SALES by year and projects on measures grossRevenue and netRevenue; the difference between these two measures is computed and named diff. Then, three models are computed: (i) a polynomial regression with 3 components (one for quantity, one for unitPrice, and one for discount, see Section 4.3); (ii) a multivariable linear regression with one component (relating diff to the other 3 measures, see Section 4.4); and a cross-correlation with 3 components (one for quantity, one for unitPrice, and one for discount, see Section 4.5). Finally, the interest is computed for the 7 components obtained (see Section 4.6); the highlight (the most interesting component, i.e., the one that best replicates the values of diff) turns out to be the polynomial expressing diff in terms of discount), which is returned to the user together with C.

4.3. Polynomial regression

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 [14]. 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 and the more complex – so the less interpretable – the model).

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 [15]

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

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.[6]

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 [16]). 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.

[4] Value types can be either *nominal* (qualitative and unordered), *ordinal* (qualitative and ordered), *interval* (quantitative with no zero point, supports the computation of a distance between values), and *ratio* (quantitative with zero point) [12,13].

[5] In the following, for simplicity, we will use labels m, $m_1, \dots, m_r$ to denote measures even in the case they correspond to expressions specifying derived measures.

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

Algorithm 1 Polyfit

```
Require: $m$: target measure; $m_i$: candidate measure
Ensure: $\alpha$: optimal polynomial
 1: $e^* \leftarrow +\infty$                                  ▷ Approximation error
 2: $d \leftarrow 0$                                          ▷ Degree
 3: $stop \leftarrow$ False                                   ▷ Stop condition
 4: do
 5:     ▷ Find the best polynomial of degree $d$...
 6:     $\alpha^d \leftarrow OrdinaryLeastSquares(m, m_i, d)$
 7:     $e \leftarrow error(m, m_i, d)$                       ▷ ...and compute its error
 8:     if $e < e^*$ then                                     ▷ If a better approximation is found...
 9:         $e^* \leftarrow e$                                ▷ ...update the error, ...
10:         $d \leftarrow d + 1$                              ▷ ...increment the degree and iterate, ...
11:     else
12:         $stop \leftarrow$ True                            ▷ ...otherwise stop
13: while $!stop \wedge (|C| \geq 10 \cdot d)$
14: return $\alpha^{d-1}$                                     ▷ Return the polynomial
```

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[7]: 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 target measure m, 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 4–13). We compute the best polynomial with the given degree d through ordinary least squares optimization (Line 6) and the error of the polynomial (Line 7). If the current error is better than the one obtained so far (Line 8), we update it (Line 9), increase the polynomial degree (Line 10), and continue with the iteration (Line 13). Otherwise, we terminate the iteration (Line 12). In any case, the iteration stops if $|C| < 10 \cdot d$ (Line 13) [17]. Finally, we return the best polynomial (i.e., the one computed before the current iteration).

Example 9. The following intention:

```
with SALES explain netRevenue by type for year = '2022'
    against unitPrice using polynomialRegression
```

is executed by first computing the cube that aggregates sales by type for 2022 and then applying the Polyfit algorithm to obtain a model with one component for measure unitPrice. Algorithm 1 iteratively finds the following polynomials with degrees from 0 to 3 (see Fig. 4):

$$\text{netRevenue} = 10737.6$$

$$\text{netRevenue} = 191.74 \cdot \text{unitPrice} - 8098.16$$

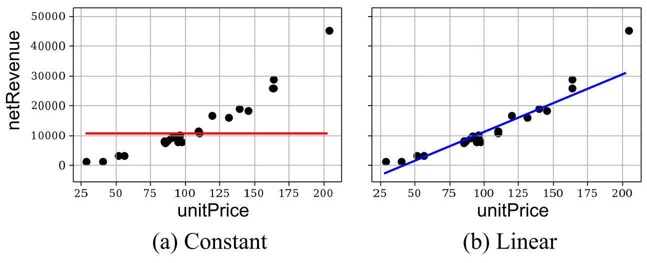
$$\text{netRevenue} = 1.1 \cdot \text{unitPrice}^2 - 22.78 \cdot \text{unitPrice} + 1409.33$$

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

As shown in Fig. 5, the quadratic polynomial is returned since the cubic one has a higher error.[8] □

[7] "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 [17].

[8] 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.



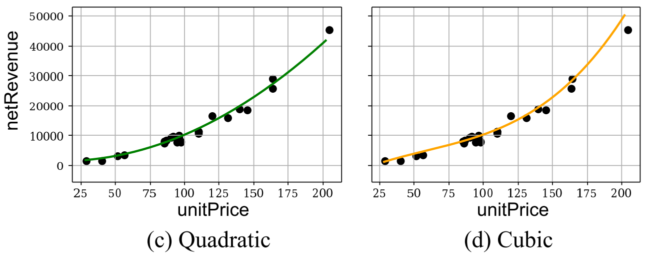


Fig. 4. Approximating polynomials with degrees 0 (a), 1 (b), 2 (c), and 3 (d) for the unitPrice measure from Example 9.

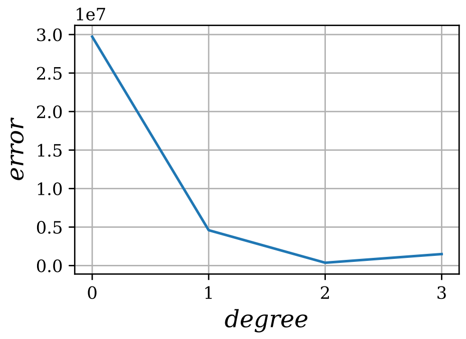


Fig. 5. Error in function of the degree for the polynomials in Fig. 4.

4.4. Multivariable linear regression

Given three or more variables in a dataset, multivariable linear regression (or simply *MultiReg*) explains the dependent variable (in our case, the target measure) by a linear function of "significant" independent ones (the candidate measures); significant means that independent variables are omitted if they have no effect on the dependent one. Finding the best subset of independent variables is impractical because of the combinatorial explosion of the number of subsets; thus, greedy algorithms are commonly adopted to this end.

Specifically, the greedy algorithm we adopt here is called *Recursive Feature Elimination* [18]; since variable selection can be non-optimal when it comes to removing several variables at a time, this algorithm finds the best linear function by minimizing the error while estimating the effect of removing one variable at a time. The process is summarized in Algorithm 2: (i) the linear regression model is trained with all the candidate measures by minimizing the approximation error through ordinary least squares [19]; (ii) the candidate measures are ranked by the absolute value of their coefficients; and (iii) the candidate measures with smallest ranking are iteratively removed. The best subset of measures is returned by choosing the subset that gives the least averaged error across different folds of the dataset; for all measures removed, the corresponding coefficients in the *coeff* array that models the polynomial within the component are set to 0. Note that, if too many measures were involved, the greedy search could be prematurely stopped based on the elbow method; this is not our case, since the results of OLAP queries usually include a limited number of measures and tuples (i.e., the cardinality of the result is "small") [20].

Algorithm 2 MultiReg

```
Require: $m$: target measure; $M$: candidate measures
Ensure: $\alpha^*$: optimal linear model
 1: $e^* \leftarrow +\infty$                                    ▷ Approximation error
 2: $M' \leftarrow M$                                           ▷ Current measures
 3: $\alpha^* \leftarrow \emptyset$                             ▷ Initialize the best polynomial
 4: do
 5:     ▷ Find the best model with measures $M'$...
 6:     $\alpha \leftarrow OrdinaryLeastSquares(m, M')$
 7:     $e \leftarrow error(m, M')$                             ▷ ...and compute its error
 8:     if $e < e^*$ then                                       ▷ If a better approximation has been found...
 9:         $e^* \leftarrow e$                                  ▷ ...update the error
10:         $\alpha^* \leftarrow \alpha$                        ▷ ...and update the model
11:     ▷ Remove the measure with the coefficient closest to 0
12:     $M' \leftarrow M' \setminus \{argmin_{m_i \in M'}(|coeff(m_i, \alpha)|)\}$
13: while $|M'| > 0$
14: return $\alpha^*$                                           ▷ Return the best linear model
```

Example 10. Given again the intention in Example 1,

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

a multivariable linear regression model based on all the other cube measures has to be computed. Algorithm 2 first ranks the measures as grossRevenue, discount, quantity, and unitPrice. Then, it iteratively removes the measures one by one (starting from unitPrice). Finally, the best linear model, involving grossRevenue and discount with their coefficients being respectively 1.0 and −1.0, is returned (indeed, netRevenue is computed in the SALES cube as grossRevenue−discount).

4.5. Cross-correlation

Given a series of a dependent variable and a series of an independent variable, cross-correlation (or simply *CrossCorr*) explains the former as a function of the best displacement relative to the latter. In our scenario, the dependent variable is the target measure and the independent variable is the candidate measure; to enable a series-wise comparison, the cube must be aggregated by a level of interval type (e.g., a temporal level such as date or month).

The process is summarized in Algorithm 3. For simplicity, we assume that the two series are complete, i.e., that no event is missing.[9] We normalize the cross-correlation function to get a time-dependent Pearson correlation coefficient, with 1 indicating perfect correlation and −1 indicating perfect anti-correlation; the correlation ρ with displacement τ between the two series in X (corresponding to the candidate measure m_i) and Y (corresponding to the target measure m) is calculated as follows:

$$\rho_{X,Y}(\tau) = \frac{\sum_i (x_i - \mu_x)(y_{i+k} - \mu_y)}{\sqrt{\sum_i (x_i - \mu_x)^2} \cdot \sqrt{\sum_i (y_{i+k} - \mu_y)^2}} \quad (1)$$

where μ and σ represent the average and deviation of each series. Then, finding the best cross-correlation requires determining the displacement τ yielding the maximum absolute correlation, i.e., $argmax_{-maxDisp \leq \tau \leq +maxDisp}(\rho_{X,Y}(\tau))$, where $maxDisp$ defines the boundaries of the search, i.e., the initial displacement. Unless differently specified, we pick $maxDisp = |X|$ so the search for the best displacement is carried out in $[-|X|, |X|]$; in other words, all the possible displacements are tried, starting from the one where the first series is shifted so that its beginning is matched with the end of the second series, and finishing with the opposite. However, using the range clause in the explain intention, it is possible to inject some a priori knowledge in the search for an explanation by specifying the maximum displacement allowed; this allows to constrain the search space on the one hand, to avoid "false positives" (i.e., displacements with high correlations but outside the range of interest) on the other.

[9] Several imputation methods can be adopted in case of missing events, ranging from arithmetic average for "steady" patterns to more complex methods for periodical ones [21].

Algorithm 3 CrossCorr

```
Require: $m$: target measure; $m_i$: candidate measure; $maxDisp$: maximum displacement
Ensure: $\tau^*$: optimal delay
 1: $corr^* \leftarrow -\infty$                                 ▷ Correlation
 2: $\tau \leftarrow -maxDisp$                                  ▷ Initial displacement
 3: do
 4:     ▷ Compute the (absolute) correlation with displacement $\tau$
 5:     $corr \leftarrow |\rho_{m_i,m}(\tau)|$
 6:     if $corr > corr^*$ then                                 ▷ If a better correlation has been found...
 7:         $corr^* \leftarrow corr$                            ▷ ...update the correlation
 8:         $\tau^* \leftarrow \tau$                            ▷ ...set the best displacement
 9:     $\tau \leftarrow \tau + 1$                              ▷ ...and increment the displacement
10: while $\tau \leq maxDisp$
11: return $\tau^*$                                             ▷ Return the best displacement
```

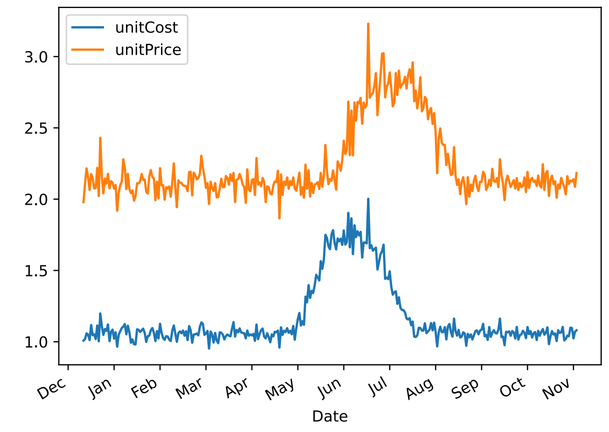


Fig. 6. Cross-correlation between unitPrice and unitCost for Example 11.

Example 11. Consider again the intention from Example 7,

with SALES, PURCHASE explain unitPrice by date
    against unitCost using crossCorrelation range 60

After joining the SALES and PURCHASE cubes and aggregating the result by date, Algorithm 3 starts by computing the cross-correlation of unitPrice and unitCost with initial displacement −60, then it incrementally increases the displacement up to 60. During this process, the best displacement (namely, 27) is found and stored, and eventually returned. Fig. 6 shows the time series for unitPrice and unitCost as computed by the intention.

4.6. Measuring the interest of components

Measuring the interest of components is crucial, since it allows for ranking them when visualizing enhanced cubes and for determining the highlight. We recall that function $interest()$ ranges in [0..1]; its definition depends on the type of model to which the component belongs:

- **Polynomial regression model.** Let c_i be the component explaining m based on m_i; we evaluate the interest of c_i, $interest(c_i)$, as the *coefficient of determination* R2 [22], which measures how well the polynomial in m_i fits the values of the target measure m; if R2 is negative, function $interest()$ returns 0.[10] The better the model, the closer the value of R2 to 1.

- **Multivariable linear regression model**. Let c_i be the component explaining m based on the independent variables $m_1, \dots, m_r$; also in this case, we evaluate the interest of c_i, $interest(c_i)$, as the coefficient of determination R2.
- **Cross-correlation model**. Let c_i be the component explaining m based on m_i; we evaluate the interest of c_i, $interest(c_i)$, via the absolute value of the correlation ρ as defined in Eq. (1). The better the model, the closer the value of ρ to 1.

Note that, while in principle R2 is applicable to cross-correlation as well, they can be discordant due to their different semantics: cross-correlation is high when two measures are "well-aligned", while R2 is high when a measure is a good approximation of another (in other words, alignment does not imply a good fit). For instance, given unitPrice and unitCost from Fig. 6, the cross-correlation value is 0.90 while R2 is negative even if computed after aligning the two measures by the displacement found; this means that the values of unitPrice are better approximated by their average rather than by the values of unitCost.

Example 12. With reference to Example 6, for the regression models on SALES_1 it is

$$interest(c_1) = 0.96$$
$$interest(c_2) = 0.99$$
$$interest(c_3) = 1.00$$

Thus, the highlight is c_3. For the cross-correlation model on SP_1, it is $interest(c_4) = 0.90$.

4.7. Model overlaps

In certain situations, some components of different models can carry the same information. For instance, if measure m is explained against a set of measures $m_1, \dots, m_r$ but is linearly dependent on measure m_1, (i) polynomial regression will return a polynomial $m = \alpha^{d_1}(m_1)$, with $d_1 = 1$; (ii) multivariable linear regression will return a polynomial $m = \alpha^1(m_1, \dots, m_r)$ where all the coefficients except the one for m_1 are null; (iii) cross-correlation will return displacement 0 between m and m_1. All three components will have high interest.

To avoid returning redundant information to users, but also to properly rank components, we introduce three domination rules. Let m be the target measure of an intention, $m_i \in In$ be a measure used for explaining m, and $Out = \bigcup_{j=1}^{z} Out_j$ be the set of all components returned by that intention:

D.1 If Out includes a polynomial regression component $c_i^{\text{poly}} = (m_i, d_i, coeff_i)$ and a multivariable linear regression component $c^{\text{multi}} = coeff$ where all coefficients except the one for m_i are 0, then c_i^{poly} dominates c^{multi}. The rationale for this rule is that multivariable linear regression always return linear polynomials, while polynomial regression can return polynomials with any degree; thus, c_i^{poly} may be more informative/accurate than c^{multi}. Indeed, in this case m is roughly expressed as $coeff_i \cdot m_i$ by c^{multi} and more accurately expressed as $\alpha^{d_i}(m_i)$ by c_i^{poly}. Clearly, in case $d_i = 1$, the two components are fully equivalent.

D.2 If Out includes a polynomial regression component $c_i^{\text{poly}} = (m_i, d_i, coeff_i)$ and a cross-correlation component $c_i^{\text{cross}} = (m_i, disp_i)$ where $disp_i = 0$, then c_i^{poly} dominates c_i^{cross}. The rationale for this rule is that cross-correlation does not return any expression relating m to m_i, hence, c_i^{poly} is more informative/accurate than c_i^{cross}. Indeed, in this case m is roughly expressed as m_i by c_i^{cross} and more accurately expressed as $\alpha^{d_i}(m_i)$ by c_i^{poly}.

D.3 If Out includes a multivariable linear regression component $c^{\text{multi}} = coeff$ where all coefficients except the one for m_i are 0 and a cross-correlation component $c_i^{\text{cross}} = (m_i, disp_i)$ where $disp_i = 0$, then c^{multi} dominates c_i^{cross}. The rationale for this rule is that cross-correlation does not return any expression relating m to m_i, hence, c^{multi} is more informative/accurate than c_i^{cross}. Indeed, in this case m is roughly expressed as m_i by c_i^{cross} and more accurately expressed as $coeff_i \cdot m_i$ by c^{multi}.

Noticeably, these rules can be applied in the user interface in two ways (depending on the user's preferences): (i) to hide the dominated components from the user's view, or (ii) to sort components yielding the same interest. Besides, they could be used to improve performances by avoiding to compute some components.

Example 13. Consider again the intention

```
with SALES explain (grossRevenue – netRevenue) as diff by year
```

from Example 8. Since netRevenue is actually computed as grossRevenue − discount, we expect that this relationship is used as an explanation for derived measure diff, i.e., that diff=discount is the relationship returned. Indeed, three components have maximum interest in this case:

$$c_1^{\text{poly}} = (\text{discount}, 1, [1, 0]), \quad interest(c_1^{\text{poly}}) = 1.0$$
$$c^{\text{multi}} = [0, 0, 1, 0], \quad interest(c^{\text{multi}}) = 1.0$$
$$c_1^{\text{cross}} = (\text{discount}, 0), \quad interest(c_1^{\text{cross}}) = 1.0$$

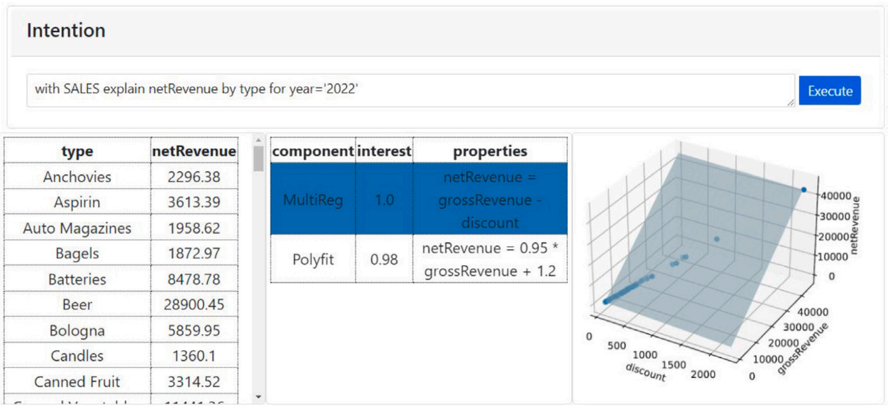
The first component expresses diff as a polynomial of degree 1 in discount, with coefficients 1 and 0. The second one expresses diff as a linear polynomial in all candidate measures, with all coefficients set to 0 except the one for discount. Finally, c_1^{cross} tells us that the time series for diff is very correlated with the one for discount, with no displacement. In fact, all three components convey the same information to the user. The dominating component returned as the highlight is, in this case, c_1^{poly}.

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 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 the target measure and the candidate measure(s). Specifically:

- For polynomial regression, we use a scatter chart superimposed with the function plotting the approximating polynomial (see [8] for an example).
- For multivariable linear regression, if the number of candidate measures with non-null coefficients is 2, we use a 3-D scatter chart superimposed with the plan plotting the approximating polynomial (see Example 14); otherwise, no chart is created.

[10] R2 compares how well a model fits m in comparison with its simple average value. More formally, it represents to what extent the variation in the dependent variable m is predictable from a model in the independent variable(s) m_i. Should the average value be a better approximation than the model, R2 is negative.



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

type	netRevenue
Anchovies	2296.38
Aspirin	3613.39
Auto Magazines	1958.62
Bagels	1872.97
Batteries	8478.78
Beer	28900.45
Bologna	5859.95
Candles	1360.1
Canned Fruit	3314.52

component	interest	properties
MultiReg	1.0	netRevenue = grossRevenue - discount
Polyfit	0.98	netRevenue = 0.95 * grossRevenue + 1.2

Fig. 7. The visualization obtained for the intention in Example 1.

Table 1
Test results in function of the cube cardinality.

$\|C\|$	Complexity (numb. of char.)			Time (s)				
	Intention	Query	Python	Query	CrossCorr	MultiReg	Polyfit	Total
12	116	268	2165	0.03	0.01	0.01	0.01	0.06
36	132	397	2841	0.04	–	0.01	0.02	0.07
333	115	265	2165	0.04	0.01	0.01	0.02	0.08
540	134	403	2841	0.05	–	0.01	0.02	0.08
1224	137	412	2841	0.06	–	0.01	0.02	0.09
12 113	133	400	2841	0.05	–	0.02	0.03	0.10
16 949	129	395	2841	0.07	–	0.02	0.04	0.13
18 492	128	385	2841	0.06	–	0.02	0.04	0.12
20 525	128	392	2841	0.07	–	0.02	0.04	0.13
77 832	127	382	2841	0.08	–	0.06	0.10	0.24
86 832	140	509	2841	0.10	–	0.05	0.11	0.26

- For cross-correlation, we use a multiple line chart showing the two series (the one for the target measure and the one for the candidate measure, see Example 11 for an example).

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

Example 14. Fig. 7 shows the visualization obtained when the intention in Example 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 (multivariable) linear polynomial that approximates netRevenue in function of grossRevenue and discount, so the chart area shows the 3-D relationship between these three measures.

6. Evaluation

The prototype we developed to test our approach uses the simple multidimensional engine described in [23], 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 and cross-correlation 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.

6.1. Efficiency

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 $|C|$ of the cube to which a model is bound, i.e., the one resulting from the by and for clauses (which determines the time required to compute a single model component), and the number of cube 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 11 intentions with increasing cardinalities; in each intention we explained the quantity measure against netRevenue, grossRevenue, discount, and unitPrice (i.e., $|M| = 5$). The intentions were computed on cubes obtained by progressively including in the group-by set levels from the Date, Product, and Store hierarchies; for polynomial regression, polynomials up to the 5th degree were considered. Note that cross-correlation could be computed for two intentions only, namely, the ones yielding $|C| = 12$ and $|C| = 333$, where the group-by levels of type interval are month and date, respectively.

The tests were run on an Intel(R) Core(TM)i7-6700 CPU@3.40 GHz CPU with 8 GB 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 87 000 facts.[11] Additionally, we measured the complexity (as the number of characters [24]) of writing explain intentions vs. the underlying cube query. It turns out that our approach saves 95% of complexity with respect to writing cube queries in SQL and writing the Python implementation necessary to compute the models (640 characters for cross-correlation, 676 for multivariate linear regression, and 1525 for polynomial regression).

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). Fig. 8 shows the performance when m_0 is explained against an increasing number of measures, up to $|M| - 1$.

As expected, cross-correlation and polynomial regression scale linearly with respect to the number of measures (because they compute a component for every candidate measure independently of the others).

[11] 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 [20].

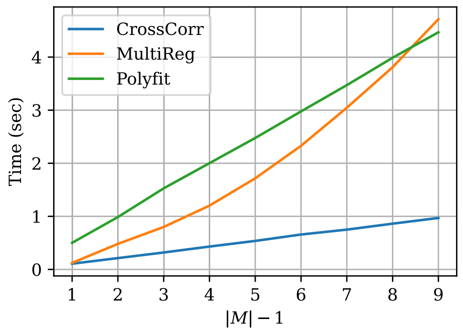


Fig. 8. Scalability test.

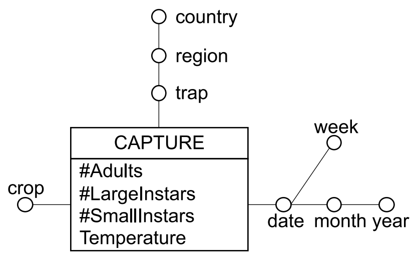


Fig. 9. Conceptual schema for the CAPTURE cube.

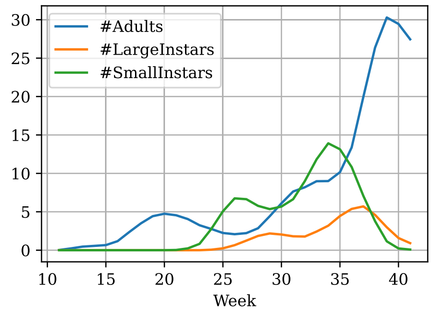


Fig. 10. Temporal trends in the CAPTURE cube.

Conversely, multivariable linear regression scales quadratically with respect to the number of measures since the complexity of Ordinary Least Square is $O(|M|^2 \cdot |C|)$. Note that also polynomial regression uses Ordinary Least Square optimization, but in this case the complexity is related to the degree of the polynomial ($O(d^2 \cdot |C|)$) and not to the number of candidate measures; in these tests we considered polynomials up to the 5th degree, which explains why polynomial regression scales linearly. Overall, given 9 measures and 10^6 facts, computing an explanation takes around 10 s, thus fulfilling the requirement of near-real-time response typical of analytical workloads.

6.2. Effectiveness

We tested our approach in terms of effectiveness from three points of view: using synthetic data, using real data, and asking for the feedback of a set of users. The results are described in the following subsection.

6.2.1. Synthetic dataset

In this test we artificially injected three patterns into the data of the SALES cube, to check that they are properly detected by the explanations generated:

1. Measure discount is computed by applying to grossRevenue a percentage randomly chosen among 0%, 5%, and 10% (in the average, 5%).
2. Measure netRevenue is computed as grossRevenue − discount.
3. Measure unitCost is computed as unitPrice/2 plus a uniformly distributed random noise in $[-\frac{\text{unitPrice}}{10}, \frac{\text{unitPrice}}{10}]$ and displaced ahead by 30 days, to simulate that the fluctuations in the price of products follow the ones in their cost.

Then, we expressed three intentions to verify that explain is capable to detect these patterns:

$I1$: with SALES explain discount by month

$I2$: with SALES explain netRevenue by type for year = '2022'

$I3$: with SALES, PURCHASE explain unitPrice by date against unitCost

The highlights returned by each intention are, respectively,

c_1^{poly} =(grossRevenue, 1, [0.05, −1.3])

c_2^{multi} =[grossRevenue = 1.0, discount = −1.0, quantity = 0.0, unitPrice = 0.0]

c_3^{cross} =(unitCost, 27)

which shows that explain detects the patterns we injected into the cube. Indeed, c_1^{poly} shows that discount is about 5% of grossRevenue; c_2^{multi} shows that netRevenue can be computed as the difference between grossRevenue and discount; c_3^{cross} shows that unitPrice is delayed by 27 days with respect to unitCost.

The reason why these explanations are not 100% precise (e.g., the displacement detected is 27 rather than 30) is that some noise is introduced when computing unitCost (as mentioned above) and also by aggregating and averaging measures at different levels of detail. Indeed, while unitCost is computed out of unitPrice at the finest cube granularity, $I3$ aggregates and averages unitPrice and unitCost over all products and all stores in the same date, thereby cumulating and propagating the noise at a coarser level of detail.

6.2.2. Real dataset

As a second test to verify that the explanations provided by our approach can effectively detect patterns present in the data, we evaluated it against the CAPTURE cube, whose conceptual schema is depicted in Fig. 9; the dataset has been collected from a real case study in the field of precision agriculture, precisely, in the context of the Agro.Big.Data.Science project [25]. This cube describes the captures of the brown marmorated stink bug (*Halyomorpha halys*), one of the main insect pest species causing economic damages to agricultural assets, in different dates, traps, and crops. Captures are characterized by the age of the insects. The cube contains four measures: the amount of captured adults, large instars, and small instars as well as the air temperature. Since it is well known that the spreading of Halyomorpha halys follows seasonal peaks [26,27], we verified if an explain intention is capable of highlighting such temporal pattern (shown in Fig. 10) as the most interesting.

The intention we formulated to this end is the following:

I : with CAPTURE explain #Adults by week

The components returned by the intention, ranked by decreasing interest, are:

c_1^{cross} = (#LargeInstars, 3),	$interest(c_1^{cross}) = 0.9$
c_2^{cross} = (#SmallInstars, 5),	$interest(c_2^{cross}) = 0.8$
c_3^{multi} = [#LargeInstars = 2.71, #SmallInstars = 0.0],	$interest(c_3^{multi}) = 0.2$
c_4^{poly} = (#LargeInstars, 1, [2.77, 5.03]),	$interest(c_4^{poly}) = 0$
c_5^{poly} = (#SmallInstars, 0, [9.21]),	$interest(c_5^{poly}) = 0$

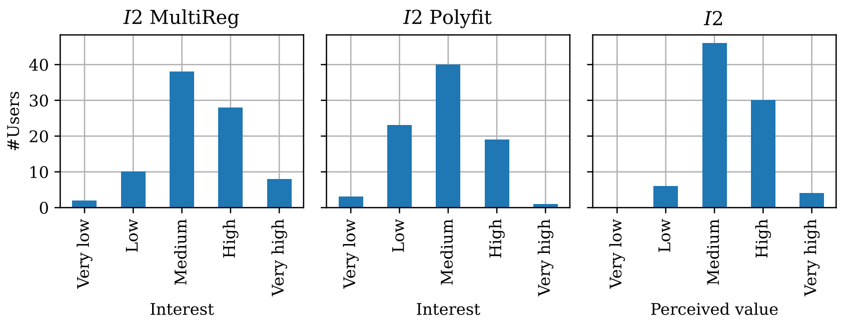


Fig. 11. Component interest and overall perceived value for intention $I2$.

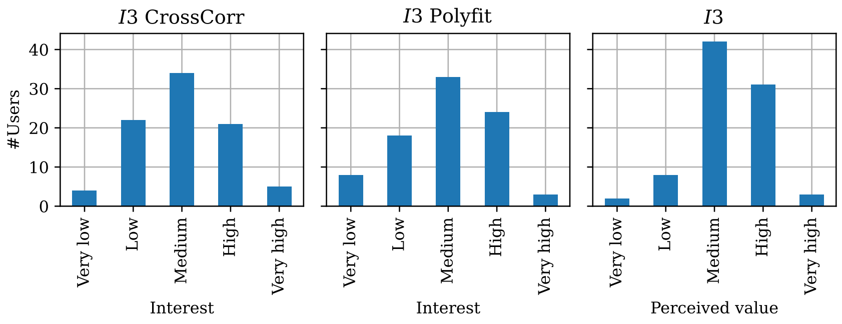


Fig. 12. Component interest and overall perceived value for intention $I3$.

The first component explains measure #Adults by pointing out that the number of adults shows a 3-week delay from the number of large instars, while the second one points out that it shows a 5-week delay from the number of small instars. This confirms that the explain operator retrieves the seasonal patterns and correctly returns the temporal displacement between adults and instars as the most interesting components.

6.2.3. *Tests with users*

As a last test of effectiveness, we experimented our approach with 86 users, mainly master students with advanced or basic knowledge of business intelligence and data warehousing. After giving them a 5-minute introduction to the explain operator and its syntax, we proceeded as follows:

(i) We showed them intentions $I2$ and $I3$ (we omitted $I1$ for the sake of time) together with the resulting cubes.

(ii) For each intention:

a. We asked them to explain the behavior of the target measure on their own by visually inspecting the cube data.
b. We proposed to them, as possible explanations, the two components with maximum interest returned by the intention.
c. We asked them to rate on a 5-values Likert scale the interest of the explanations we provided and their overall perceived value (e.g., how well our explanations were aligned with their own explanation)

(iii) We asked them to rate the overall user experience.

Figs. 11, 12, and 13 show the results in the form of bar charts. In $I2$ (Fig. 11), consistently with the interest of the components, the users deemed the result of multivariable linear regression (our highlight, i.e., the fact that netRevenue = grossRevenue − discount) more interesting than the one of polynomial regression. Surprisingly, in $I3$ (Fig. 12), the users deemed polynomial regression as interesting as cross-correlation, since Polyfit returns a more detailed explanation (a polynomial rather than a simple measure of displacement between the two time series). Overall, the perceived value of both the explanations and the user experience (Fig. 13) are good. This suggests that indeed explain achieves good results, although it could be improved by refining the explanations provided and adding the ones suggested by the users. Interestingly, the main users' suggestions concern the adoption of qualitative models (e.g., grossRevenue *is proportional to* quantity, uniprice, and netRevenue but *not to* discount; unitPrice *is always higher than* unitCost); although we agree that these models could be more intuitive that the quantitative ones we generate, we observe that they seem more aimed at providing *descriptions* rather than explanations.

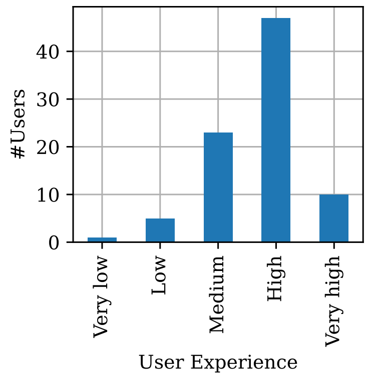


Fig. 13. User experience rating.

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 [28]. Later on, data-to-model unification was addressed in MauveDB [29], 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 [30]. More recently, Northstar [31] 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 [1], 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 multidimensional data for user's evaluation, was the goal of some approaches commonly labeled as OLAM [32]. In this context, k-means clustering is used in [33] to dynamically create semantically-rich aggregates of facts other than those statically provided by dimension hierarchies. Similarly, the shrink operator is proposed in [34] to compute small-size approximations of a cube via agglomerative clustering. Other operators that enrich data with knowledge extraction results are DIFF [35], which returns a set of tuples that most successfully describe the difference of values between two facts of a cube, and RELAX [36], which verifies whether a pattern observed at a certain level of detail is also present at a coarser level of detail, too. Finally, in [37] 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 [38]. See [39] 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 [40]. 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 [7,41]; the intuition is that tuples with high contribution tend to be interesting explanations to query answers. Similarly, in [40] 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 [42] or to return more specific explanations concerning subgroups of answers determined via clustering [43]. Other approaches to query explanation rely on ontologies [44,45].

Causality poses additional challenges when the query contains aggregates [7], as in our scenario. The DIFF operator [35] 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 [46], outliers are explained in terms of properties of the tuples used to compute these outliers, while [47] explains outliers in aggregation queries through counter-balancing. Specifically, this explanation determines the predicates that, when applied to the input data, cause the outliers to disappear. LensXPlain [48] 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 [49]. 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* [50]. 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) [19].

Regression is used to explain query results in the XAXA approach [50]. 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.

Some examples of possible alternatives to polynomial regression are: (i) multivariable regression, where the explanation is expressed as the relationship between a set of variables [51]; (ii) symbolic regression, where the explanation is expressed as a combination of mathematical expressions [52]; and (iii) HSIC lasso, where the explanation is expressed using a feature selection method that also considers non-linear relationships between variables [53]. In principle, all these regression techniques could be plugged into our approach; in this paper we only considered multivariable regression, while investigating the applicability of symbolic regression and HSIC lasso are left for future work.

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) shift towards models that explain measure values in terms of dimension members, also considering aggregation; (ii) generalize the definition of model to cope with these additional model types; and (iii) experiment other interest metrics [54]. In particular, as to the last point, we plan to consider the framework proposed in [39] to evaluate explanations in terms of *succinctness* (large explanations will probably be not well understandable), *interpretability* (the suitability of an explanation will depend on the target users), and *actionability* (explanations should point to actionable suggestions).

CRediT authorship contribution statement

Matteo Francia: Writing – review & editing, Writing – original draft, Validation, Software, Investigation, Data curation, Conceptualization. **Stefano Rizzi:** Writing – review & editing, Writing – original draft, Supervision, Investigation, Formal analysis, Conceptualization. **Patrick Marcel:** Writing – review & editing, Writing – original draft, Supervision, Investigation, Formal analysis, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

[1] P. Vassiliadis, P. Marcel, S. Rizzi, Beyond roll-up's and drill-down's: An intentional analytics model to reinvent OLAP, Inf. Syst. 85 (2019) 68–91.

[2] M. Francia, M. Golfarelli, P. Marcel, S. Rizzi, P. Vassiliadis, Assess queries for interactive analysis of data cubes, in: Proceedings of EDBT, Nicosia, Cyprus, 2021, pp. 121–132.

[3] M. Francia, P. Marcel, V. Peralta, S. Rizzi, Enhancing cubes with models to describe multidimensional data, Inf. Syst. Front. 24 (1) (2022) 31–48.

[4] M. Francia, M. Golfarelli, P. Marcel, S. Rizzi, P. Vassiliadis, Suggesting assess queries for interactive analysis of multidimensional data, IEEE Trans. Knowl. Data Eng. 35 (9) (2023) 6421–6434.

[5] G.R. Mayes, Theories of Explanation, Internet Encyclopedia of Philosophy, 2018.

[6] C.C. Aggarwal, M.A. Bhuiyan, M.A. Hasan, Frequent pattern mining algorithms: A survey, in: C.C. Aggarwal, J. Han (Eds.), Frequent Pattern Mining, Springer, 2014, pp. 19–64.

[7] A. Meliou, W. Gatterbauer, J.Y. Halpern, C. Koch, K.F. Moore, D. Suciu, Causality in databases, IEEE Data Eng. Bull. 33 (3) (2010) 59–67.

[8] M. Francia, S. Rizzi, P. Marcel, The whys and wherefores of cubes, in: Proceedings of DOLAP, Ioannina, Greece, 2023, pp. 43–50.

[9] M. Golfarelli, S. Rizzi, Data Warehouse Design: Modern Principles and Methodologies, McGraw-Hill, 2009.

[10] M. Golfarelli, D. Maio, S. Rizzi, The dimensional fact model: A conceptual model for data warehouses, Int. J. Coop Inf. Syst. 7 (2–3) (1998) 215–247.

[11] M. Terrovitis, P. Vassiliadis, S. Skiadopoulos, E. Bertino, B. Catania, A. Maddalena, S. Rizzi, Modeling and language support for the management of pattern-bases, Data Knowl. Eng. 62 (2) (2007) 368–397.

[12] S.S. Stevens, On the theory of scales of measurement, Science 103 (2684) (1946) 677–680.

[13] J. Zhang, A representational analysis of relational information displays, Int. J. Hum.-Comput. Stud. 45 (1) (1996) 59–74.

[14] E. Ostertagová, Modelling using polynomial regression, Procedia Eng. 48 (2012) 500–506.

[15] P.J. Bickel, K.A. Doksum, Mathematical Statistics: Basic Ideas and Selected Topics, Volumes I-II Package, Chapman and Hall/CRC, 2015.

[16] J. Mark, M.A. Goldberg, Multiple regression analysis and mass assessment: A review of the issues, Apprais. J. 56 (1) (1988) 89–109.

[17] P. Peduzzi, J. Concato, E. Kemper, T.R. Holford, A.R. Feinstein, A simulation study of the number of events per variable in logistic regression analysis, J. Clin. Epidemiol. 49 (12) (1996) 1373–1379.

[18] I. Guyon, J. Weston, S. Barnhill, V. Vapnik, Gene selection for cancer classification using support vector machines, Mach. Learn. 46 (2002) 389–422.

[19] P.-N. Tan, M. Steinbach, V. Kumar, Introduction To Data Mining, Pearson Education India, 2016.

[20] M. Francia, M. Golfarelli, S. Rizzi, A-BI$^+$: A framework for augmented business intelligence, Inf. Syst. 92 (2020) 101520.

[21] C. Yozgatligil, S. Aslan, C. Iyigun, I. Batmaz, Comparison of missing value imputation methods in time series: the case of Turkish meteorological data, Theor. Appl. Climatol. 112 (2013) 143–167.

[22] R.G.D. Steel, J.H. Torrie, Principles and Procedures of Statistics, with Special Reference To the Biological Sciences, McGraw-Hill, New York, 1960.

[23] M. Francia, E. Gallinucci, M. Golfarelli, COOL: A framework for conversational OLAP, Inf. Syst. 104 (2022) 101752.

[24] S. Jain, D. Moritz, D. Halperin, B. Howe, E. Lazowska, SQLShare: Results from a multi-year SQL-as-a-service experiment, in: Proceedings of SIGMOD, San Francisco, CA, USA, 2016, pp. 281–293.

[25] Agro.big.data.science, 2019, https://big.csr.unibo.it/projects/cimice/monitoring.php?lan=EN, accessed 2023-11-08.

[26] E. Costi, T. Haye, L. Maistrello, Biological parameters of the invasive brown marmorated stink bug, Halyomorpha halys, in southern Europe, J. Pest Sci. 90 (4) (2017) 1059–1067.

[27] M. Rot, L. Maistrello, E. Costi, S. Trdan, Biological parameters, phenology and temperature requirements of Halyomorpha halys (Hemiptera: Pentatomidae) in the sub-Mediterranean climate of Western Slovenia, Insects 13 (10) (2022) 956.

[28] L.D. Raedt, A perspective on inductive databases, SIGKDD Explorations 4 (2) (2002) 69–77.

[29] A. Deshpande, S. Madden, MauveDB: supporting model-based user views in database systems, in: Proceedings of SIGMOD, Chicago, IL, USA, 2006, pp. 73–84.

[30] T.B. Pedersen, Warehousing the world: A vision for data warehouse research, in: S. Kozielski, R. Wrembel (Eds.), New Trends in Data Warehousing and Data Analysis, in: Annals of Information Systems, vol. 3, Springer, 2009, pp. 1–17.

[31] T. Kraska, Northstar: An interactive data science system, Proc. VLDB Endow. 11 (12) (2018) 2150–2164.

[32] J. Han, OLAP mining: Integration of OLAP with data mining, in: Proceedings of Working Conf. on Database Semantics, Leysin, Switzerland, 1997, pp. 3–20.

[33] F. Bentayeb, C. Favre, RoK: Roll-up with the K-means clustering method for recommending OLAP queries, in: Proceedings of DEXA, Linz, Austria, 2009, pp. 501–515.

[34] M. Golfarelli, S. Graziani, S. Rizzi, Shrink: An OLAP operation for balancing precision and size of pivot tables, Data Knowl. Eng. 93 (2014) 19–41.

[35] S. Sarawagi, Explaining differences in multidimensional aggregates, in: Proceedings of VLDB, Edinburgh, Scotland, 1999, pp. 42–53.

[36] G. Sathe, S. Sarawagi, Intelligent rollups in multidimensional OLAP data, in: Proceedings of VLDB, Rome, Italy, 2001, pp. 531–540.

[37] B. Chen, L. Chen, Y. Lin, R. Ramakrishnan, Prediction cubes, in: Proceedings of VLDB, Trondheim, Norway, 2005, pp. 982–993.

[38] A. Meliou, S. Roy, D. Suciu, Causality and explanations in databases, Proc. VLDB Endow. 7 (13) (2014) 1715–1716.

[39] B. Glavic, A. Meliou, S. Roy, Trends in explanations: Understanding and debugging data-driven systems, Found. Trends Databases 11 (3) (2021) 226–318.

[40] S. Roy, D. Suciu, A formal approach to finding explanations for database queries, in: Proceedings of SIGMOD, Snowbird, UT, USA, 2014, pp. 1579–1590.

[41] A. Meliou, W. Gatterbauer, K.F. Moore, D. Suciu, The complexity of causality and responsibility for query answers and non-answers, Proc. VLDB Endow. 4 (1) (2010) 34–45.

[42] S. Roy, L.J. Orr, D. Suciu, Explaining query answers with explanation-ready databases, Proc. VLDB Endow. 9 (4) (2015) 348–359.

[43] A. Moreau, O. Pivert, G. Smits, A clustering-based approach to the explanation of database query answers, in: Proceedings of FQAS, Cracow, Poland, 2015, pp. 307–319.

[44] Z. Wang, M. Chitsaz, K. Wang, J. Du, Towards scalable and complete query explanation with OWL 2 EL ontologies, in: Proceedings of CIKM, Melbourne, Australia, 2015, pp. 743–752.

[45] F. Croce, M. Lenzerini, A framework for explaining query answers in DL-Lite, in: Proceedings of EKAW, Nancy, France, 2018, pp. 83–97.

[46] E. Wu, S. Madden, Scorpion: Explaining away outliers in aggregate queries, Proc. VLDB Endow. 6 (8) (2013) 553–564.

[47] Z. Miao, Q. Zeng, B. Glavic, S. Roy, Going beyond provenance: Explaining query answers with pattern-based counterbalances, in: Proceedings of SIGMOD, Amsterdam, The Netherlands, 2019, pp. 485–502.

[48] Z. Miao, A. Lee, S. Roy, LensXPlain: Visualizing and explaining contributing subsets for aggregate query answers, Proc. VLDB Endow. 12 (12) (2019) 1898–1901.

[49] K.E. Gebaly, P. Agrawal, L. Golab, F. Korn, D. Srivastava, Interpretable and informative explanations of outcomes, Proc. VLDB Endow. 8 (1) (2014) 61–72.

[50] F. Savva, C. Anagnostopoulos, P. Triantafillou, Explaining aggregates for exploratory analytics, in: Proceedings of BigData, Seattle, WA, USA, 2018, pp. 478–487.

[51] B. Hidalgo, M. Goodman, Multivariate or multivariable regression? Am. J. Public Health 103 (1) (2013) 39–40.

[52] L. Billard, E. Diday, Symbolic regression analysis, in: K. Jajuga, A. Sokołowski, H.-H. Bock (Eds.), Classification, Clustering, and Data Analysis: Recent Advances and Applications, Springer, 2002, pp. 281–288.

[53] W. Ren, B. Li, M. Han, A novel Granger causality method based on HSIC-Lasso for revealing nonlinear relationship between multivariate time series, Physica A 541 (2020) 123245.

[54] A. Chanson, B. Crulis, K. Drushku, N. Labroche, P. Marcel, Profiling user belief in BI exploration for measuring subjective interestingness, in: Proceedings of DOLAP, Lisbon, Portugal, 2019, pp. 1–9.