Suggesting assess queries for interactive analysis of multidimensional data

Matteo Francia, Matteo Golfarelli, Patrick Marcel, Stefano Rizzi, and Panos Vassiliadis

Abstract—Assessment is the process of comparing the actual to the expected behavior of a business phenomenon and judging the outcome of the comparison. The assess querying operator has been recently proposed to support assessment based on the results of a query on a data cube. This operator requires (i) the specification of an OLAP query to determine a target cube; (ii) the specification of a reference cube of comparison (benchmark), which represents the expected performance; (iii) the specification of how to perform the comparison, and (iv) a labeling function that classifies the result of this comparison. Despite the adoption of a SQL-like syntax that hides the complexity of the assessment process, writing a complete assess statement is not easy. In this paper we focus on making the user experience more comfortable by letting the system suggest suitable completions for partially-specified statements. To this end we propose two interaction modes: progressive refinement and auto-completion, both starting from an assess statement partially declared by the user. These two modes are evaluated both in terms of scalability and user experience, with the support of two experiments made with real users.

Index Terms—OLAP, analytics, data exploration.

1 Introduction

In the context of data analysis, *assessment* is a process aimed at comparing the actual behavior of some phenomenon to the expected behavior and judging, typically through a *labeling*, the outcome of the comparison. For instance, an analyst may want to assess the state of monthly COVID-19 infections in France for 2021. (S)he will then have to issue a query against an OLAP server to obtain a data cube, and then ask: "how good, normal, bad is the situation I observe for this cube as compared to some reference data?". Examples of how to assess the status of each single cell of a cube include its comparison to a golden standard (e.g., compare French infections against the EU average) and to sibling cells (e.g., compare French infections to those in Italy, or infections of April 2021 to those of March 2021), possibly using some numerical property (e.g., country populations) to scale values and make the comparison more meaningful.

Assessment is recognized to be one of the main types of analysis [1] and is consistently reported as a frequent activity of data explorers [2], [3], who often carry it out using SQL in combination with languages like Python and R. In data storytelling, the *compare* pattern (which lies at the core of the assessment process) has been discovered to be the one most used in data stories [4], [5].

Assessment is one of the user's intentions considered in the *Intentional Analytics Model* (IAM), which has been envisioned as a way to tightly couple OLAP and analytics [6]. The IAM approach relies on two cornerstones: (i) users explore the data space by expressing their analysis *intentions* rather than by explicitly stating what data they need, and (ii) in return they receive both multidimensional data and knowledge insights in the form of annotations of interesting subsets of data. In this context, the assess operator has been introduced to complement the traditional OLAP roll-up's and drill-down's by judging a cube measure with reference to some baseline. The idea of how to perform an assessment for the measure values of a cube (called *target cube*) encompasses (a) the specification of another cube, called *benchmark*, that represents the expected or desirable performance of the measure; (b) the *comparison* of the measure under investigation to the benchmark measure (for instance via a simple mathematical difference); and (c) the characterization, or *labeling*, of the status of the target cube based on the result of the comparison. In [7] we have proposed a SQL-like declarative syntax for the assess operator, defined its semantics, and experimentally evaluated alternative plans for its execution.

Example 1. Let a COVID19 cube be given, showing the number of new cases and the number of deaths by date and country. A user's intention can be expressed via the assess operator with this statement:

```
with COVID19 for country = 'Italy' by country, month
assess cases against country = 'France'
using ratio(cases, benchmark.cases)
labels {[0, 0.5): quite lower, [0.5, 0.8): lower,
        [0.8, 1.25]: same,
        (1.25, 2]: higher, (2, +inf]: quite higher}
```

Intuitively, the monthly cases in Italy are assessed against those in France and labeled as higher/same/lower based on their ratio. As also illustrated in Figure 1, the semantics of this statement can be informally explained as follows: (A) access the cube; (B) get the slice for Italy and group it by month; (C) get the slice

- *M. Francia, M. Golfarelli, and S. Rizzi are with DISI - University of Bologna, Italy.*
- *P. Marcel is with the University of Tours, Blois, France.*
- *P. Vassiliadis is with the University of Ioannina, Greece.*

A

	Italy	France	…
2020-03-01	655	115	…
2020-03-02	709	123	…
2020-03-03	768	141	…
…	…	…	…

B

	Italy
2020-03	96000
2020-04	99986
2020-05	35344
2020-06	7291
2020-07	7760

C

	France
2020-03	40044
2020-04	84401
2020-05	27178
2020-06	11183
2020-07	24983

D

	Italy
2020-03	96000, 40044
2020-04	99986, 84401
2020-05	35344, 27178
2020-06	7291, 11183
2020-07	7760, 24983

E

	Italy	
2020-03	2.4	quite higher
2020-04	1.2	same
2020-05	1.3	higher
2020-06	0.7	lower
2020-07	0.3	quite lower

Fig. 1. Assessment data for Example 1

for France and group it by month; (D) join the two slices; and (E) compute the ratio and apply the labeling scheme. □

However, some preliminary tests have shown that, despite the adoption of a SQL-like syntax that hides the complexity of the assessment process, writing a complete assess statement may not be that easy for users. In fact, even for skilled analysts it may be unclear which specific data can be conveniently used as a benchmark, but also which comparison operator and labeling scheme provide the most effective characterization of the assessment. Thus, in this paper we focus on making the user experience more comfortable and fruitful by letting the system suggest suitable completions for partially-specified statements. To this end we propose and compare two interaction modes: *progressive refinement* and *auto-completion*, both starting from an assess statement partially declared by the user.

- In progressive refinement, the system supports users in specifying the missing clauses one by one. The idea is to let users drive the process while still relieving them from the complexity of completing the assessment in all its parts. To ensure that multiple, diverse aspects of the data are covered, so that the user is able to learn about their different properties, we resort to *diversification*, a technique often used to this end in exploratory data analysis [8], [9]. Specifically, as sketched in Figure 2, the system (i) first proposes a subset of benchmarks that can give the user a wide and diversified view of the situation (e.g., compare a slice against a sibling slice); (ii) then, once the user has chosen one of them, the system lets him/her choose a comparison function (e.g., relative difference) among a small set determined again via diversification; and (iii) finally, the system considers the labeling schemes (e.g., quartiles) compatible with the chosen comparison function and suggests a representative subset. All of this is done in a notebook-like mode, accompanying each suggested statement with a chart visualizing its result, so that the user can easily keep the analysis flow under control. Diversification ensures that, each time, the user is presented with more than one options for each choice.
- In auto-completion (Figure 3), the system relies on the same criteria mentioned above to propose one representative statement that completes the one partially specified by the user, and shows its results (much like a "I feel lucky" button). Then the user is free to manually edit the statement to better adjust it to his/her needs, or even switch to the progressive refinement mode.

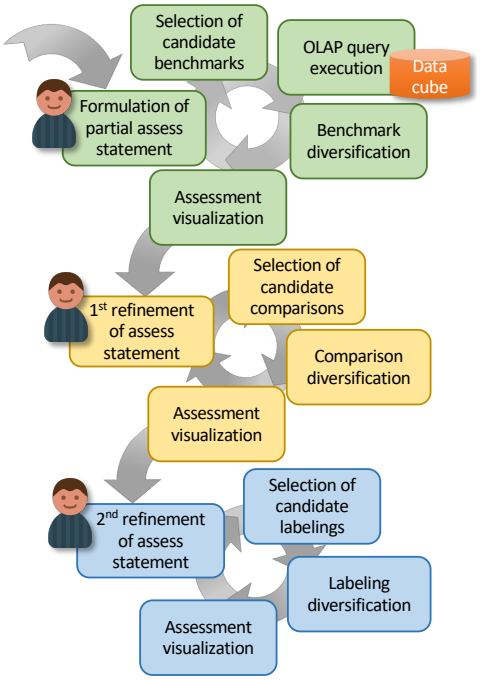

Fig. 2. Functional view of the progressive refinement process

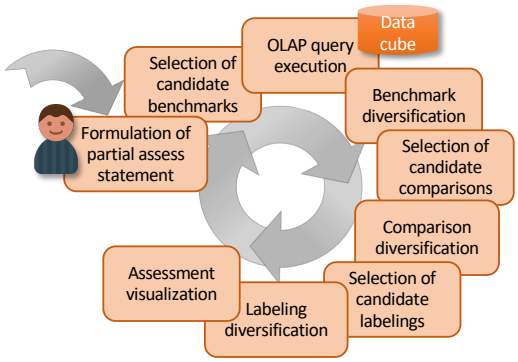

Fig. 3. Functional view of the auto-complete process

The comparison between these two interaction modes is made through a set of experimental tests focused both on efficiency and effectiveness, the latter being assessed with the support of real users.

Roadmap. In Section 2 we formalize the involved concepts and introduce our working example. In Section 3 we explain how assessments are computed, while in Section 4 we describe the operator syntax. In Section 5 we present the basic phases that enable partial assessments to be completed, together with the progressive refinement and auto-completion processes. Section 6 discusses the results of the experimental tests we performed. The paper is completed by Section 7, which discusses the related literature, and Section 8, which summarizes our findings.

2 FORMALITIES

To simplify the formalization, we will restrict to consider linear hierarchies.

Definition 1 (Hierarchy and Cube Schema). A *hierarchy* is a triple $h = (L, \succeq, \geq)$ where:

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

A *property* is a numerical attribute describing a level. Each level l is associated with a (possibly empty) set of *properties*, $Prop(l)$; given member $u \in Dom(l)$ we will denote the value taken by property $d \in Prop(l)$ for member u with $u.d$. The part-of partial order is such that, for each couple of levels l and l' such that $l \succeq l'$, for each member $u \in Dom(l)$ there is exactly one member $u' \in Dom(l')$ such that $u \geq u'$.

A *cube schema* is a couple $\mathcal{C} = (H, M)$ where:

(i) H is a set of hierarchies;
(ii) M is a tuple of numerical measures, each coupled with one aggregation operator $op(m) \in \{\texttt{sum}, \texttt{avg}, \texttt{min}, \texttt{max}\}$.

We will write $l \dot{\succeq} l'$ to denote that l' is the level immediately below l in the roll-up order.

Example 2. As a working example we will use cube schema $\mathsf{COVID19} = (H, M)$, where

$$H = \{h_{\mathsf{Date}}, h_{\mathsf{Country}}\},\ M = \langle \mathsf{cases}, \mathsf{deaths} \rangle,$$
$$L_{\mathsf{Date}} = \{\mathsf{date}, \mathsf{month}, \mathsf{year}\}, \mathsf{date} \succeq \mathsf{month} \succeq \mathsf{year},$$
$$Dom(\mathsf{date}) = \{\text{2021-04-15}, \ldots\},$$
$$Dom(\mathsf{month}) = \{\text{2021-04}, \ldots\}$$
$$Dom(\mathsf{year}) = \{2019, 2020, 2021\},$$
$$L_{\mathsf{Country}} = \{\mathsf{country}, \mathsf{continent}\}, \mathsf{country} \succeq \mathsf{continent}$$
$$Dom(\mathsf{country}) = \{\text{Italy}, \text{France}, \text{Greece}, \ldots\},$$
$$Dom(\mathsf{continent}) = \{\text{Europe}, \text{Australia}, \ldots\}$$

and $op(\mathsf{cases}) = op(\mathsf{deaths}) = \texttt{sum}$. Level country has a property named population, with France.population $= 67060000$. As to the part-of partial order we have, for instance, Greece $\geq$ Europe and 2021-04-15 $\geq$ 2021. See the supplemental material for more details. □

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 (Group-by Set and Coordinate). Given cube schema $\mathcal{C} = (H, M)$, a *group-by set* of $\mathcal{C}$ is a tuple 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 and another group-by set G' such that $G \succeq_H G'$, we will denote with $rup_{G'}(\gamma)$ the coordinate of G' whose members are related to the corresponding members of γ in the part-of orders, and we will say that γ *roll-up*s to $rup_{G'}(\gamma)$. By definition, $rup_G(\gamma) = \gamma$.

Definition 3 (Detailed Cube). Let G_0 be the top group-by set in the $\succeq_H$ partial order (i.e., the finest one). A *detailed cube* over $\mathcal{C}$ is a partial function C_0 that maps the coordinates of G_0 to a numerical value for each measure m in M.

The function is partial since cubes are normally *sparse*: not all possible business events actually occur, and a coordinate participates in the function only if the event it describes took place. Each coordinate γ that participates in C_0, with its associated tuple t of measure values, is called a *cell* of C_0 and denoted $c = \langle \gamma, t \rangle$. With a slight abuse of notation, we will also consider a cube as the set of the coordinates corresponding to its cells, so we will write $\gamma \in C_0$ to state that $\langle \gamma, t \rangle$ is a cell of C_0.

Example 3. Three group-by sets of COVID19 are $G_0 = \langle \mathsf{date}, \mathsf{country} \rangle$, $G_1 = \langle \mathsf{month}, \mathsf{continent} \rangle$, and $G_2 = \langle \mathsf{year} \rangle$, where $G_0 \succeq_H G_1 \succeq_H G_2$. G_0 is the top group-by set. G_1 aggregates cases by date and country, G_2 by year over the whole world. Examples of coordinates of the three group-by sets are, respectively, $\gamma_0 = \langle \text{2020-03-01}, \text{France} \rangle$, $\gamma_1 = \langle \text{2021-03}, \text{Europe} \rangle$, and $\gamma_2 = \langle 2020 \rangle$, where $rup_{G_1}(\gamma_0) = \gamma_1$ and $rup_{G_2}(\gamma_1) = \gamma_2$. An example of cell of a detailed cube over COVID19 is $\langle \gamma_0, \langle 115, 21 \rangle \rangle$ (see Figure 1.A). □

Definition 4 (Cube Query and Derived Cube). Given a detailed cube C_0 over schema $\mathcal{C}$, a *query* over C_0 is a quadruple $q = (C_0, G_q, P_q, m)$ where:

(i) G_q is a group-by set of $\mathcal{C}$;
(ii) P_q is a (possibly empty) set of selection predicates each expressed over one level of H;
(iii) $m \in M$.

The result of q is called a *derived cube*, i.e., a partial function that assigns to each coordinate γ of G_q satisfying the conjunction of the predicates in P_q the value computed by applying $op(m)$ to the values of m for all the coordinates of C_0 that roll-up to γ, provided that such coordinates of C_0 exist.

Like detailed cubes, even derived cubes can be sparse; a coordinate γ does not participate in the function if there is no coordinate in C_0 that rolls-up to γ. Like for detailed cubes, we will write $\gamma \in C$ to state that γ is a coordinate of the derived cube C.

Example 4. A cube query over COVID19 is $q = (C_0, G_q, P_q, m)$ where $G_q = \langle \mathsf{month}, \mathsf{country} \rangle$, $P_q = \{\mathsf{country} = \text{'Italy'}\}$, and $m = \langle \mathsf{cases} \rangle$. The resulting cube is shown in Figure 1.B; one of its cells is $\langle \langle \text{2020-04}, \text{Italy} \rangle, \langle 99986 \rangle \rangle$. See the supplemental material for more details. □

3 COMPUTING AN ASSESSMENT

As explained in [7], the assessment of the values of a measure m in a target cube C is done in four steps:

1) the specification of a *benchmark*, i.e., a cube B such that (i) its cells can be mapped one-to-one with the cells

of C, and (ii) it has a measure m' representing the expected/acceptable/normal performance of m;
2) the cell-wise *comparison* of m to m', which can be done for instance using algebraic/absolute/relative difference or ratio, possibly applying some normalization to m and m' (e.g., to normalize the number of cases based on the country population);
3) the characterization, or *labeling*, of the status of each cell of C based on the result of the comparison; in the simplest case, this is done using a set of rules that map the result of the comparison to a set of predefined labels (e.g., "insufficient", "excellent", etc.).

Each step will be explained in detail in the following subsections.

3.1 Benchmarks

A thorough comparison of a target cube C against a benchmark B would require that the latter comes with the same level members, so that each cell of C can map onto one cell of B. However, due to cube sparsity, there is no guarantee that all cells can be mapped. In the following we provide a broad definition of the conditions under which two cubes are joinable, i.e., one of them can be used as a benchmark to assess the other; in this definition, we just require that the group-by set of the benchmark is coarser that the one of the target cube, so that there is a many-to-one mapping (induced by the part-of order) between the cells of the former and those of the latter.

Definition 5 (Cube Joinability). Let a target cube C and a benchmark B over the same cube schema $\mathcal{C} = (H, M)$ be given. Let $q = (C_0, G, P, m)$ and $q' = (C_0, G', P', m')$ be the queries that resulted in C and B, respectively. We say that C and B are *joinable* if $G \succeq_H G'$.

While in [7] four types of (joinable) benchmarks were defined, within the scope of progressive refinement and auto-completion we focus on two types[1]:

- *Sibling benchmarks*. The idea is to compare the values of a measure m in a slice on member $u \in Dom(l)$ with its values in another slice of C related to a sibling member $u' \in Dom(l)$ (e.g., assess the COVID-19 cases in Italy with reference to those in France, as done in Example 1). Both cubes have the same group-by set, but while the cells in C are those obtained from C_0 using predicate $l = u$, those in B are obtained from C_0 using predicate $l = u'$. Then the cell-to-cell mapping is established by replacing u with u' in each coordinate of C. In case of missing tuples in the benchmark, the corresponding coordinate is removed from the result.
- *Parent benchmarks*. In this case the user wants to assess the values taken by m in each cell of C against the one taken in a parent (aggregated) cell (e.g., assess the cases in Italy with reference to those in Europe). So let G' be a group-by set such that $G \succ_H G'$. Then the cell with coordinate $\gamma \in C$ is mapped onto the one with coordinate $rup_{G'}(\gamma)$.

1. The other types of benchmarks are *constant benchmarks* and *external benchmarks*. A constant benchmark is a predefined target value for a measure; clearly, this value must be provided by the user and cannot be inferred by the system. An external benchmark is a cube acting as a golden standard for the assessment; again, this cube must necessarily be indicated by the user.

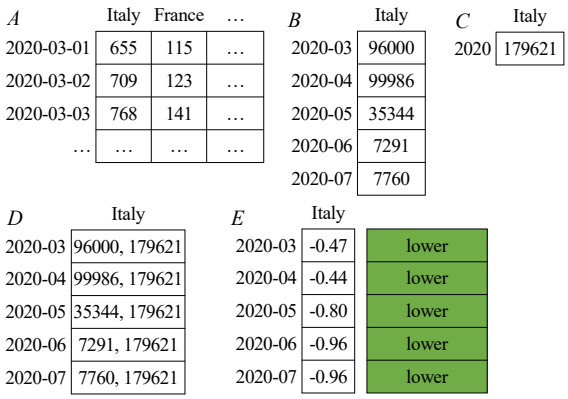

Fig. 4. Assessment data for Example 5

Noticeably, in some cases directly comparing the values of m with its values in the benchmark is not the best option — or even makes little sense. For instance, assume we wish to assess the cases in Italy against those in Luxembourg. The population of Italy is almost 100 times the one of Luxembourg, so a direct comparison between the number of cases in these two countries would be quite unfair. One way to make the comparison fair would be to use some scaling factor, e.g., the population. The same problem arises with parent benchmarks, because comparing the cases in Italy against the total number of cases in Europe would make little sense. While a simple option here would be to compare against the average cases over European countries, a more sophisticated alternative is to use again population as a scaling factor.

Example 5. Let C be the derived cube obtained by query q in Example 4 (total monthly cases in Italy, Figure 1.B). An example of sibling benchmark is B_{sib} returned by q_{sib}, being q_{sib} obtained from q by replacing Italy with France (Figure 1.C). B_{sib} can be used to assess the cases in Italy against those in France, as in Example 1. The cell-to-cell mapping is established by replacing Italy with France; so, for instance, coordinate $\langle$2020-04, Italy$\rangle$ is mapped onto $\langle$2020-04, France$\rangle$ (Figure 1.D). An example of parent benchmark for the same target cube C is B_{par} returned by q_{par}, being q_{par} obtained from q by replacing its group-by set with $G_{par} = \langle$year, country$\rangle$ ($G \succeq_H G_{par}$) and by dividing the yearly totals by 12. This example is shown in Figure 4, where A is the detailed cube C_0 and B is the target cube. B_{par} (Figure 4.C) can be used to assess the monthly cases in Italy against the average monthly cases in Italy during the whole year. The cell-to-cell mapping is established by mapping each month onto its year (Figure 4.D); so, for instance, coordinate $\langle$2020-04, Italy$\rangle$ is mapped onto $\langle$2020, Italy$\rangle$. Note that, in both these examples, descriptive attribute population could be used for scaling. □

3.2 Comparison

The essence of assessment is to contrast the actual performance against its expected value. Thus, the goal of this

step is to provide the means to express and perform the evaluation of how far apart the query result and the benchmark are. We refer to this action as *comparison* to express the idea that this is not necessarily a simple measure difference. Modeling-wise, we assume that a library of comparison functions, all with signature $\delta : \mathbb{R}\times\mathbb{R} \to \mathbb{R}$, is available to the users. Practically, a cell-wise comparison between measures of the target and benchmark can be easily implemented via different functions obeying the above signature. The functions we will consider here are[2]:

$$\mathsf{difference}(m, m') = m - m'$$
$$\mathsf{relDifference}(m, m') = \frac{m - m'}{m'}$$
$$\mathsf{ratio}(m, m') = \frac{m}{m'}$$

An example of comparison based on the ratio function is shown in Figure 1.E.

3.3 Labeling

The goal of this step is to associate each cell of the target cube with a label, taken from a predefined set, to express an evaluation of that cell with reference to the benchmark. We assume a total ordering is defined on labels. Given a finite set of distinct values V, a labeling function is a surjective function that takes the form $\lambda : \mathbb{R} \to V$. Each value resulting from the comparison of a target cube cell with the corresponding benchmark cell is fed to the labeling function, and assigned the appropriate label.

The labeling functions we consider are of two types:

- *Functions based on explicit ranges*. Specifically, we defined functions with 2, 3, and 5 labels that can operate on values resulting from either a difference (centered on 0 and working with absolute values), or a relative difference (centered on 0 and working with percentage values), or a ratio (centered on 1). For instance, the function used in Example 1 relies on 5 labels and operates on ratio-based comparisons.
- *Functions based on the overall value distribution*. Specifically, we consider an equi-depth binning function (quartiles, can be coupled with any comparison function) and two equi-width binning functions (with 3 and 5 labels, can be coupled with both difference and relative difference).

A simple example of a labeling function based on explicit ranges is the one proposed in Example 1, whose resulting labels are shown in Figure 1.E.

Example 6. Consider again Example 5, based on the cube C yielding the monthly cases in Italy. Let us focus on parent benchmark B_{par} which yields, for each year, the average monthly cases in Italy (Figure 4.C). A comparison between C and B_{par} can be done using relDifference, coupled with a labeling defined as follows:

$$\lambda(x) = \begin{cases} \text{lower, if } -\mathsf{inf} \leq x < -0.1 \\ \text{same, if } -0.1 \leq x \leq 0.1 \\ \text{higher, if } 0.1 < x \leq \mathsf{inf} \end{cases}$$

2. When using ratio, in case $m' = 0$ the string inf is returned.

Now consider cell $\langle\langle \text{2020-04}, \text{Italy}\rangle, \langle 99986\rangle\rangle$, which is compared with $\langle\langle 2020, \text{Italy}\rangle, \langle 179621\rangle\rangle$ (Figure 4.D; the total cases in Italy in 2020 were 2155446, and 2155446/12=179621). The relative difference between these two values is -0.44, meaning that the cases in April were 44% less than the monthly average; thus, the cell is labeled as 'lower' (Figure 4.E). □

4 The assess operator

In this section we define the syntax of the assess operator. This syntax relies on the one proposed in [7], extended with a clause to specify normalized comparisons and restricted to sibling and parent benchmarks. The operator takes a detailed cube C_0 in input and returns a derived cube C (the target cube) where each cell is accompanied by the result of the comparison and by the corresponding label. The main parameters that drive the process are (i) the measure m to be assessed, (ii) a selection predicate P along with (iii) a group-by set G to slice and aggregate C_0, (iv) a comparison function, and (v) a labeling function. Additional parameters depend on the benchmark type.

The general syntax for writing an assess statement includes three parts: one (consisting of the with, assess, by, and for clauses) that specifies the target cube; one (consisting of the against and scaled clauses) that specifies the benchmark; one (consisting of the using and labels clauses) that specifies the assessment method:

with C_0 [for P] by G
assess m against $< benchmark >$ [scaled d]
using $< function >$ labels λ

where C_0 is a detailed cube (with schema $\mathcal{C} = (H, M)$), m is a measure of C_0, P is an (optional) set of selection predicates each of type $l = u$ (where l is a level of H and $u \in Dom(l)$), G is a group-by set of $\mathcal{C}$, $< benchmark >$ is the benchmark specification, d is a property, $< function >$ is a comparison function, and λ is a labeling function.

The target cube C is defined by aggregating C_0 on G and selecting the cells that meet the conjunctive predicates in P. As to the benchmark, its specification can take two forms:

- In a sibling benchmark, the for clause must include a predicate which slices the target cube on member u of level $l \in G$. In this case, m is assessed against a benchmark related to a different member of l, say u':

  with C_0 for $p_1, \ldots, p_k, l = u$ by G
  assess m against $l = u'$ [scaled d]
  using $< function >$ labels λ

  where $d \in Prop(l)$. For instance, this clause can be used to assess the monthly cases in $u =$ 'Italy' against those in $u' =$ 'France' as in Example 1. If no scaled clause is specified, the benchmark is characterized by $G' = G$, $P' = P \backslash \{(l = u)\} \cup \{(l = u')\}$, and $m' = m$. In practice, the slicing on u is replaced by one on u'. If the scaled clause is specified, it is used to scale the values of m in the benchmark, so it is

  $$m' = m \cdot \frac{u.d}{u'.d}$$

- Let $G = \langle l_1, \ldots, l, \ldots, l_n \rangle$. In a parent benchmark, the against clause specifies the parent level l', $l \dot{\succeq} l'$, to be used for aggregation:

  with C_0 for P by G
  assess m against l' [scaled d]
  using $< function >$ labels λ

  where $d \in Prop(l)$. For instance, this clause can be used to assess the monthly cases (l = month) against the yearly ones (l' = year). Let $p_l \in P$ be the predicate, if any, expressed in the for clause on level l. The benchmark is characterized by $P' = P \setminus \{p_l\}$ and $G' = \langle l_1, \ldots, l', \ldots, l_n \rangle$. As to the benchmark measure m', we observe that clearly, if $op(m) = \texttt{sum}$, the values of m in the target cube cannot be directly compared to those in the benchmark, so it cannot be $m' = m$.[3] If no scaled clause is specified, a simple average is used; thus, for each coordinate $\gamma = \langle v_1, \ldots, u, \ldots, v_n \rangle \in C$, with $u \in Dom(l)$, it is

  $$m' = m \cdot \frac{1}{|Dom_u(l)|}$$

  where $Dom_u(l) = \{u_i \in Dom(l) \text{ s.t. } u_i \geq u'\}$, $u' \in Dom(l')$, and $u \geq u'$ (intuitively, for each member u, m' is the average of m over all the children of u's parent in the part-of order). This means that, for instance, the cases of April 2020 would be assessed against the average of the monthly cases of 2020. If the scaled clause is specified, it is used to scale the values of m in the benchmark; thus, for each coordinate $\gamma = \langle v_1, \ldots, u, \ldots, v_n \rangle \in C$ it is

  $$m' = m \cdot \frac{u.d}{\sum_{u_i \in Dom_u(l)} u_i.d}$$

Finally, as to the assessment method, its specification is based on the using and labels clauses:

- The using clause specifies a function (e.g., a difference or a ratio) that describes how the comparison is made; a keyword benchmark is used to distinguish the cells of the target cube from the corresponding ones in the benchmark.
- The labels clause specifies a labeling function, either based on explicit ranges (e.g., negative values are bad, positive values are good) or on the overall value distribution (e.g., quartiles), to be applied to the result of the computation specified by the using clause.

In all cases above, the result returned to the user includes, for each cell, (i) its coordinate, (ii) the value of m for that coordinate, (iii) the value of m' for the benchmark, (iv) the value resulting from the comparison, and (v) the corresponding label.

3. For simplicity here we assume that scaling is applied to additive measures and properties only. Investigating how to properly scale measure values in the other cases is left for future work.

Example 7. A statement based on a sibling benchmark has already been shown in Example 1. One based on a parent benchmark is given below:

with COVID19 for country = 'Italy' by country, month
assess cases against year
using ratio(cases, benchmark.cases)
labels {[0, 0.5): quite lower, [0.5, 0.8): lower,
[0.8, 1.25]: same,
(1.25, 2]: higher, (2, +inf]: quite higher}

This one assesses the monthly cases in Italy against the average cases for each year. Thus, as already explained in Example 6, the cases in Italy for April 2020 are compared against one twelfth of the cases in Italy for the whole 2020. Similarly, the following one assess the monthly cases in Italy against the European average scaled by the country population:

with COVID19 for country = 'Italy' by country, month
assess cases against continent scaled population
using ratio(cases, benchmark.cases)
labels {[0, 0.5): quite lower, [0.5, 0.8): lower,
[0.8, 1.25]: same,
(1.25, 2]: higher, (2, +inf]: quite higher}

5 Completing partial assessments

In this section we discuss how a partial assessment specified by the user can be completed. The partial statement must include at least the with, by, and assess clauses, while the for clause is optional:

with C_0 [for P] by G assess m

We will assume for simplicity that no other clauses are specified by the user at this stage; clearly, if either the against, using, or labels clauses are included in the partial statement, the corresponding refinement step will be omitted.

After explaining how we determine the sets of candidate benchmarks, comparisons, and labelings, we describe the diversification step aimed at picking some representative candidates to be suggested to the user. Then we explain how the results of each step are visualized. Finally, we describe the overall processes of refinement and auto-completion.

5.1 Selection of candidates

Let P and G be, respectively, the (possibly empty) set of predicates (for clause) and the group-by set (by clause) in the partial assessment.

(i) The set of candidate benchmarks, S, is made of both parent and sibling benchmarks. Specifically:

- for each $l \in G$, S includes a parent benchmark on the level l' immediately below l in the roll-up order ($l \dot{\succeq} l'$);
- for each $l \in G$, if there is a predicate $(l = u) \in P$, then S includes a sibling benchmark for each other member of $Dom_u(l)$.

(ii) The set of candidate comparisons, R, includes the three functions listed in Section 3.2, each applied with no scaled clause and with a scaled clause on each property in $Prop(l)$. The ratio function is excluded when the domain of the measure to be assessed includes negative values.
(iii) The set of candidate labelings, T, includes all the functions that are compatible with the comparison function selected.

Example 8. Let the following one be the partial statement formulated by the user:

with COVID19 for country = 'Italy' by country, month
assess cases

Then, the set of candidate benchmarks includes two parent benchmarks and one sibling benchmark for each other European country, resulting in the following possible refinements: against continent, against year, against country = 'France', etc. As to candidate comparisons, level country has one population property, so R includes six comparisons (the three functions of Section 3.2, each applied with and without a scaled population clause).

5.2 Diversification

In querying and analysis applications, the capability of ranking data with respect to *diversity* features is becoming more and more valuable [9]. The idea is that not only the retrieved elements should be as relevant as possible to the query, but also that the result set should be as diverse as possible. Specifically, diversification is achieved by maximizing the sum of inter-element distances amongst elements of the result set [8].

Diversification is applied in our approach to select representative candidates among benchmarks, comparisons, and labelings. To this end we adopted the k-medoids algorithm [10], which partitions the candidates into k clusters while minimizing intra-cluster distance, and returns cluster centroids as representative candidates. The number of clusters k determines the alternatives that will be offered to users. Setting k to a fixed, small value ensures adaptation to our scenario, in which there are interaction and visualization constraints; however, it is also possible to automatically tune k based on the actual distribution of values (e.g., as done in [11]).

Diversifying benchmarks, comparisons, and labelings requires different distance functions, as explained below; we recall that such functions are the ones to be minimized for building clusters of similar elements.

(i) *Benchmark diversification*. Picking a subset of representative benchmarks from S is based on the distance of their measure values. We adopt the Euclidean distance since measure values belong to $\mathbb{R}$ and have the same semantics.
(ii) *Comparison diversification*. Directly diversifying comparisons results —e.g., the results of a ratio and a difference— is not sound, as comparison functions convey different semantics. Thus, to pick a subset of representative comparisons from R we rely on the similarity between meta-features describing the result of the comparison. Specifically, we compute the mean, variance, and skewness of each comparison and use these values as the components of the vectors to be diversified [12]. Euclidean distance between meta-feature vectors is then adopted.
(iii) *Labeling diversification*. Picking a subset of representative labelings from T is based on the *agreement* between label values; being ordinal, label values act as ranks of the comparison result. In particular, we adopt the Kendall's Tau distance [13], a metric that counts the number of disagreements between two ranking lists by counting the swaps necessary to place one list in the same order as the other list. Specifically, the Kendall's tau distance between two rankings λ_1 and λ_2, denoted $K(\lambda_1, \lambda_2)$, is defined as the number of pairs of candidates whose relative ordering is different between λ_1 and λ_2.

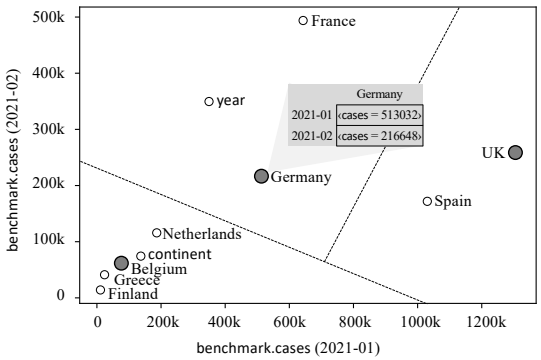

Fig. 5. Benchmark diversification; representative candidates are shown as larger, grey circles

Once pairwise distances between candidate items (either benchmarks or comparisons or labelings) are computed, these items are split into k clusters whose centroids are picked as the k representatives.

Example 9. Consider again the partial intention in Example 8, restricted for simplicity to two months: January and February 2021. Figure 5 positions some candidate benchmarks (2 parent benchmarks and 10 sibling benchmarks for European countries) within a scatter chart based on the number of cases they yield in the two selected months, showing the three clusters obtained and the corresponding representative benchmarks. An example of computation of the Euclidean distance between two candidate benchmarks to be used for clustering is: $\sqrt{(513032-186524)^2+(216648-115767)^2} = 341737$, where 513032 and 216648 (186524 and 115767) are the numbers of cases in Germany (Netherlands) in January and February 2021, respectively.

5.3 Visualization

This step aims at letting the user preview the assessment results at each refinement, so that (s)he can pick the suggestion in terms of benchmark/comparison/labeling (s)he deems most interesting for completing the statement. To choose an approach for visualization we considered that:

1) The dimensionality of the result will be $b+2$, where b is the number of levels in the by clause that are not also

included in the for clause (plus one dimension for the result of comparison and one for the labels).
2) The by clause can include at most two levels besides those also appearing in the for clause, so $b \leq 2$ and the dimensionality will be at most 4.
3) Level members typically have either nominal, ordinal, or interval type, while the comparison result is normally a ratio value and the labels are ordinals.
4) The visualization should be oriented to comparison, since the user will use it to pick the most interesting suggestion.

Based on the above, and taking into account the commonly adopted data visualization guidelines [14], we opted for using small multiples [15] featuring either bar graphs or bubble graphs. After each refinement, a small multiple is drawn featuring one graph for each suggested assessments. Bar graphs are used when dimensionality is 2 or 3; the vertical axis shows the member names, while the bar length and color show, respectively, the comparison value and the label of each cell. Bubble graphs are used when dimensionality is 4; the two axes show the member names for the two grouping levels, while the bubble size and color show, respectively, the comparison value and the label of each cell. The most proper color code can be interactively selected by the user (e.g., a red-to-green code can be selected to emphasize that some labels are "preferred" to others).

Figure 6 shows an example of visualization (obtained through the web-based interface for our prototype implementation) after the first step of progressive refinement. The two benchmarks proposed are of parent and sibling type, respectively; the default comparison and labeling functions used are shown in angular brackets.

5.4 Progressive refinement

As already mentioned, the basic idea of this interaction mode is to let the users drive the process step by step while relieving them from the complexity of completing the assessment in all its parts. We emphasize that this complexity arises not so much from the syntax of the operator, but rather from the difficulty in choosing a proper benchmark, comparison, and labeling among the set of possible alternatives.

The first refinement stage is aimed at the selection of a benchmark and takes place in four steps (see Figure 2):

1.1 *Selection of candidate benchmarks*. The set of candidate benchmarks S is determined (Section 5.1, (i)).
1.2 *OLAP query execution*. Given the levels in the group-by set G expressed in the by clause and the selection P in the for clause, the goal of this step is to retrieve the target cube C and join it with the corresponding (parent and sibling) benchmarks. A parent benchmark B on level l' (e.g., continent), direct parent of $l \in G$ (country), is obtained with a cube query whose selection is P and whose group-by set is $G \setminus \{l\} \cup \{l'\}$. B is then joined with C. Noticeably, this join can be avoided for sibling benchmarks. By issuing a single cube query whose group-by set is G and whose selection drops predicate $l = u$ from P, both the target slice and the sibling slices are retrieved; sibling slices are then pivoted into measures as explained in [7].
1.3 *Benchmark diversification* (Section 5.2, (i)) is applied to obtain a set $\overline{S} \subseteq S$ of k representative benchmarks. Each benchmark in $\overline{S}$ is then associated with one default comparison (difference) and with one default labeling (quartiles) to obtain complete statements.
1.4 *Assessment visualization* and *Refinement of assess statement*. The user previews the results of the assessments based on $\overline{S}$ and on the default comparison and labeling (Section 5.3), and selects one of the proposed assessments for further refinement.

Let $B \in \overline{S}$ be the benchmark used in the assessment selected by the user. The second refinement stage is aimed at the selection of a comparison to be coupled with B and takes place in three steps:

2.1 *Selection of candidate comparisons*. The set of candidate comparisons R is determined (Section 5.1, (ii)).
2.2 *Comparison diversification* (Section 5.2, (ii)) is applied to obtain a set $\overline{R}$ of k representative comparisons. Each comparison in $\overline{R}$ is then associated with one default labeling (quartiles) to obtain complete statements.
2.3 *Assessment visualization* and *Refinement of assess statement*. The user previews the results of the assessments based on B, on $\overline{R}$, and on the default labeling (Section 5.3), and selects one of the proposed assessments for further refinement.

Let $r \in \overline{R}$ be the comparison used in the assessment selected by the user. The last stage is aimed at the selection of a labeling to be coupled with B and r and takes place in three steps:

3.1 *Selection of candidate labelings*. The set of candidate labelings T is determined (Section 5.1, (iii)).
3.2 *Labeling diversification* (Section 5.2, (iii)) is applied to obtain a set $\overline{T}$ of k representative labelings.
3.3 *Assessment visualization*. The user previews the results of the assessments based on B, r, and $\overline{T}$ (Section 5.3).

Note that, at each stage, the user can edit all the assess statements either to complete and execute them, or to modify the suggestions given by the system.

In terms of computational cost, in Section 6.1 we demonstrate that -quite expectedly- among all these steps, the one taking most time is by far 1.2, which reads the target and benchmark data from the DBMS.

5.5 Auto-completion

The goal of this interaction mode is to give the users who formulate a partial intention an immediate and representative completion, to let them quickly evaluate its results. Clearly, the users can then manually edit the statement in each of its clauses to better tune it to their analysis interests. The process can be sketched as follows:

1 *Selection of candidate benchmarks*. The set of candidate benchmarks S is determined (Section 5.1, (i)).
2 *OLAP query execution*. Same as for progressive refinement.
3 *Benchmark diversification* (Section 5.2, (i)) is applied to obtain one representative benchmark $B \in S$; specifically, B is the centroid of S obtained via the k-medoid algorithm with $k = 1$.
4 *Selection of candidate comparisons*. The set of candidate comparisons R is determined (Section 5.1, (ii)).

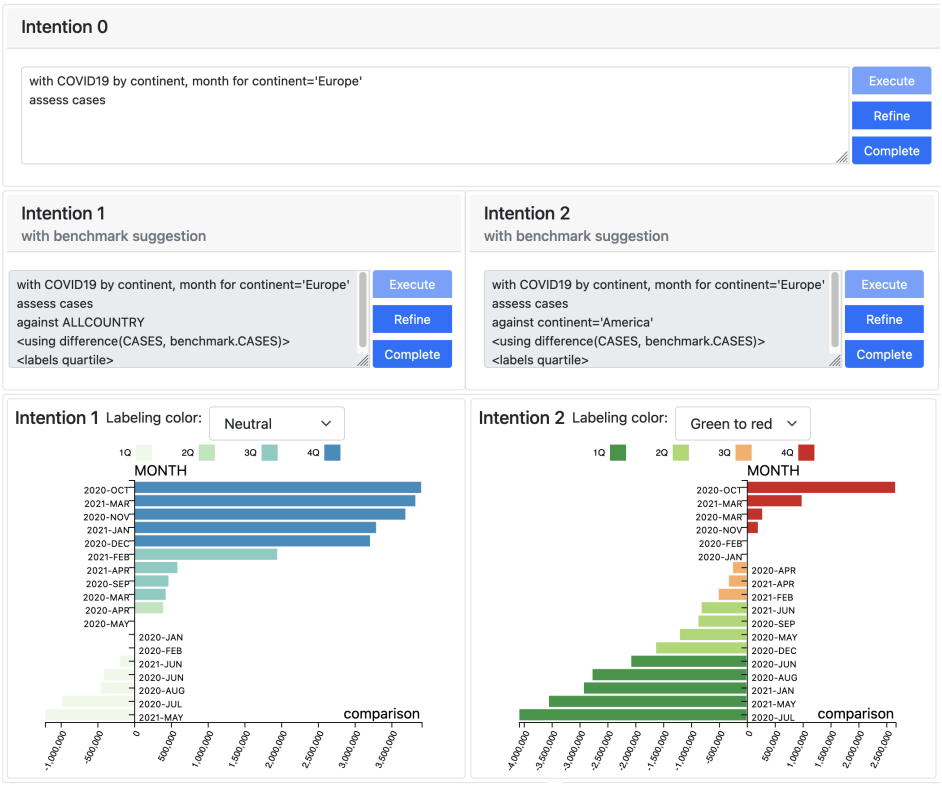

Fig. 6. An example of visualization after the first step of progressive refinement (the picture only shows two suggestions for better readability)

5 *Comparison diversification* (Section 5.2, (ii)) is applied to obtain one representative comparison $r \in R$.
6 *Selection of candidate labelings*. The set of candidate labelings T is determined (Section 5.1, (iii)).
7 *Labeling diversification* (Section 5.2, (iii)) is applied to obtain one representative labeling $\lambda \in T$.
8 *Assessment visualization*. The user previews the results of the assessment based on B, r, and λ (Section 5.3).

Again, the most expensive step is 2, which reads the target and benchmark data from the DBMS.

6 Experiments

The source code for the prototype implementation we used for the tests is available at https://github.com/big-unibo/assess; the web-based interface can be accessed at http://big.csr.unibo.it/projects/assess/. The prototype uses the simple multidimensional engine described by [16], which in turn relies on the Oracle 11g 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 mining models are imported from the Scikit-Learn Python library.

6.1 Scalability

These tests aim at evaluating the querying performance by measuring the time required to return the assessment under different conditions. To this end we used the Star Schema Benchmark (SSB) cube, described by four hierarchies; please refer to [17] for the logical schema of SSB. Specifically, we generated four base SSB cubes, namely

TABLE 1
Times for computing an assessment with parent benchmarks (in seconds); $|G|$ denotes the number of levels in the by clause

	Partial refinement				Auto-complete			
$\|G\|$	SSB_1	SSB_5	SSB_{10}	SSB_{15}	SSB_1	SSB_5	SSB_{10}	SSB_{15}
1	11	37	84	97	11	40	95	116
2	24	100	223	293	28	107	242	340
3	70	332	690	1045	73	331	746	1145

SSB_1, SSB_5, SSB_{10}, and SSB_{15}, with different scale factors resulting in the following cardinalities:

$$|SSB_1| = 6 \cdot 10^6, |SSB_5| = 3 \cdot 10^7$$
$$|SSB_{10}| = 6 \cdot 10^7, |SSB_{15}| = 9 \cdot 10^7$$

Note that the cardinality of each cube is equal to the number of tuples in the corresponding fact table. As commonly done in OLAP settings, primary and foreign keys were indexed using B-Trees. Each test has been repeated multiple times and the average results are reported. The tests were run on an Intel(R) Core(TM)i7-6700 CPU@3.40GHz with 8GB RAM.

Tables 1 and 2 show the times required to compute different assessments based on parent and sibling benchmarks, respectively, with the four SSB cubes and in either partial refinement or auto-complete mode. Our first comment is that the performance of the refinement and auto-complete modes are roughly the same (for a fair comparison, in partial refinement mode the times taken to execute each step are summed up); thus, in the following we will focus on the auto-complete mode only.

TABLE 2
Times for computing an assessment with sibling benchmarks (in seconds); $|P|$ denotes the number of predicates in the for clause

	Partial refinement				Auto-complete			
$\|P\|$	SSB_1	SSB_5	SSB_{10}	SSB_{15}	SSB_1	SSB_5	SSB_{10}	SSB_{15}
1	14	42	99	109	13	44	98	124
2	21	60	146	160	23	62	146	164
3	26	74	194	204	30	79	192	204

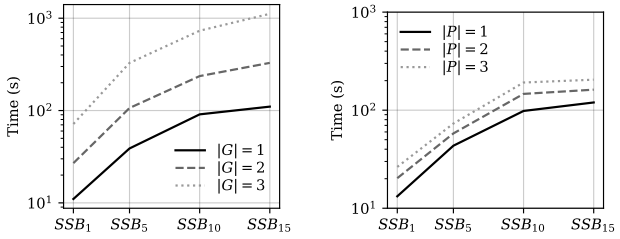

Fig. 7. Times for computing an assessment with parent (left) and sibling (right) benchmarks in auto-complete mode

Figure 7 shows how performance scales when the cardinality of the cube and the assessment complexity are increased, for parent and sibling benchmarks. For parent benchmarks, the simplest (partial) statement is with SSB by year assess quantity, which is made more complex by adding first category, then nation to the by clause. The cardinalities of the target cubes for the three resulting assessments are 7, 175, and 4375, respectively. For sibling benchmarks, the simplest (partial) statement is with SSB for year = '1992' by year assess quantity, which is made more complex by adding first a predicate on category, then one of nation to the for clause (and the corresponding levels to the by clause). The cardinality of the target cubes here is always 1. The number of siblings involved is 7 for year, 25 for category, and 5 for nation.

When increasing the cardinality of the SSB cube, the execution times scale linearly for the parent benchmark and less than linearly for the sibling benchmark. In the first case, as confirmed by Table 1, the relative increase in time is the same as the one in cardinality, since each intention accesses all the cube tuples (i.e., no selection predicate is applied). For the sibling benchmark (Table 2), the selection predicates restrict the access to fewer cube tuples, so the time increase is smaller.

We also note that, expectedly, the execution time increases proportionally to the number of parents and siblings since (i) adding parents and siblings increases the number of joins necessary to build the benchmark, and (ii) adding parents increases the cardinality of such cube. In particular, in sibling benchmarks the increase in time is smaller than in parent benchmarks due to the selectivity of the predicates involved in the intention; intuitively, the computation of a parent benchmark typically requires a large portion of the cube to be accessed (e.g., to assess the cases for Italy, all the cells of Europe must be accessed), while the computation of a sibling benchmark only requires accessing a smaller cube slice (e.g., to assess the cases in Italy, the corresponding cells for France must be accessed).

TABLE 3
Breakdown of the execution time for SSB_{15} (in seconds)

		$\|C\|$	Querying	Comparison	Labeling
Sibling	$\|P\| = 1$	1	124	0.007	0.02
	$\|P\| = 2$	1	164	0.007	0.02
	$\|P\| = 3$	1	204	0.007	0.02
Parent	$\|G\| = 1$	7	116	0.007	0.02
	$\|G\| = 2$	175	340	0.007	0.1
	$\|G\| = 3$	4375	1137	0.02	8

Finally, Table 3 shows the execution time for the auto-complete mode, broken down into (i) querying (selection of candidate benchmarks, OLAP query execution, and diversification), (ii) comparison (selection of candidate comparisons and diversification), and, (iii) labeling (selection of candidate labelings and diversification). For both sibling and parent benchmarks, the largest amount of time is spent (by far) in reading the target and benchmark data from the DBMS; this time could be improved by leveraging workload-specific optimization strategies. While the diversification time necessary to select comparison functions is almost negligible, diversifying labeling schemes takes slightly longer due to the computation of the Kendall Tau distance [13].

6.2 User experience

In this section we describe two experiments carried out to put our approach to the test with real users. The first experiment, called **Exp1** from now on, concerns a cube measuring the electric consumption in France[4] by IRIS (an IRIS is a subdivision of a French town), year, consumption category (residential, professional, enterprise), and commercial sector (e.g., telecommunications and catering). The second experiment, **Exp2**, has been conducted on the COVID19 cube used as a working example throughout the paper[5]. The test had three main goals: (i) evaluate the user satisfaction with the interaction paradigm and the assess operator, (ii) compare the auto-complete and progressive refinement modes in terms of user satisfaction and formulation effort, and (iii) evaluate the saving in formulation effort of the two modes as compared to manual writing of assess statements. See the supplemental material for more details about the data used.

In all our tests we set $k = 3$ to avoid burdening users with too much information.

6.2.1 Description

The relevant figures for the two cubes used in the experiments are summarized in Table 4. In **Exp1**, 5 volunteers were involved. All of them were data journalists or data enthusiasts with little or no skill in databases and OLAP but a good knowledge of the business domain (electric consumption in France). In **Exp2**, 10 PhD students and postdocs with good database and OLAP skills and some generic

4. https://data.enedis.fr/explore/dataset/consommation-electrique-par-secteur-dactivite-iris/export/
5. https://www.ecdc.europa.eu/en/publications-data/data-national-14-day-notification-rate-covid-19

TABLE 4
Main figures for the cubes used in the experiments

Cube name	ELEC_CONS	COVID19
♯ dimensions	4	2
♯ measures	1	2
cube cardinality	2,968,697	14,704
dimension cardinality	$45,268 \times 10 \times 3 \times 97$	77×214
total number of levels	12	6

knowledge of the business domain (COVID-19 infections) were involved.

In both cases, the users were first shown a 10 minutes tutorial video explaining the meaning of assessment, the operator syntax, and the test goal[6]; then they were assigned two tasks:

Task 1: Assess a specific cube measure for a specific slice starting from a given partial statement. This had to be done in three steps: (i) use auto-completion to obtain a fully-specified statement; (ii) manually edit this fully-specified statement to try to obtain a more interesting result; (iii) starting again from the initial partial statement, use progressive refinement to obtain a fully-specified statement.

Task 2: Given a generic assessment goal, take 20 minutes to assess the cube measures by writing one or more partial statements and freely using one or the other mode.

Eventually, we collected the feedback of the users by means of a questionnaire. The participants answered a set of questions aimed at assessing their satisfaction from different points of view. For the first 14 questions we adopted a 5-point Likert scale, which allows a neutral midpoint and two nuances for positive and negative answers. These questions are:

1) Satisfaction with the assessment obtained by auto-completion (Task 1.i)
2) Satisfaction with the assessment obtained by editing the previous one (Task 1.ii)
3) Experience with the auto-complete mode in terms of quality and interest
4) Experience with the auto-complete mode in terms of effort
5) Satisfaction with the assessment obtained by progressive refinement (Task 1.iii)
6) Experience with the progressive refinement mode in terms of quality and interest
7) Experience with the progressive refinement mode in terms of effort
8) Satisfaction with the benchmarks proposed (Task 2)
9) Satisfaction with the comparisons proposed (Task 2)
10) Satisfaction with the labeling schemes proposed (Task 2)
11) Satisfaction with the label coloring schemes proposed (Task 2)
12) Complexity of assess statements
13) Understandability of assess statements
14) Preference of refinement over auto-completion

The last question was in open form and aimed at collecting general suggestions.

6.2.2 Results

We start by discussing **Exp1**. As shown in Figure 8, the progressive refinement mode is preferred to the auto-complete mode in terms of effort (questions 4 and 7) by data journalists, while it is considered mostly equivalent in terms of satisfaction with the assessment obtained (questions 1 and 5) and overall experience (questions 3 and 6). Manual editing after auto-completion brings some marginal improvement (questions 1 and 2). Among the single components of the approach that were evaluated, comparison functions were liked the least (question 9) while coloring schemes were liked the most (question 11). Users found the syntax of complete assess statements (those suggested by the system) complex and not very understandable (questions 12 and 13). At the end of the test, no user expressed a preference for auto-completion over progressive refinement (question 14).

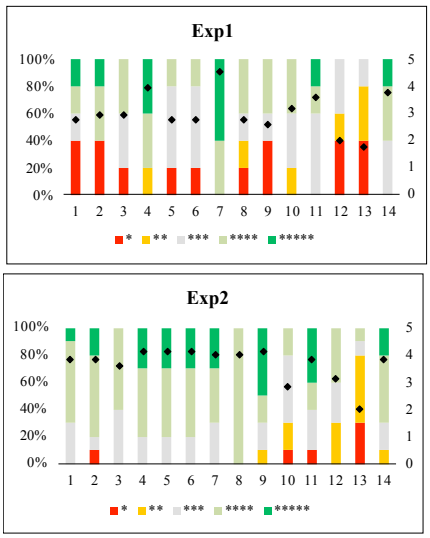

Fig. 8. Questionnaire results for the user experiments

As to **Exp2**, progressive refinement is preferred by students in terms of effort but also of satisfaction and overall experience. Manual editing after auto-completion seems to bring no real improvement. Among the single components of the approach, labeling schemes were liked the least while comparison functions were liked the most. The judgement of understandability and complexity is better than the one by journalists. Again, at the end of the test, no user expressed a preference for auto-completion over progressive refinement.

The suggestions given in open form from both user groups were mainly focused on auto-completing the names of members to facilitate writing the for clause, on the difficulty in interpreting the charts at first glance, and on the possibility of having explanations of the selections made by the system.

Table 5 shows, for both experiments and both modes, the average number of assess statements issued, the average percentage of characters saved in formulating an assessment using our approach, the average percentage of edited statements, the average time taken by the user to interpret the charts obtained and take a further action, and the average time taken by the system to compute an assessment. As an example of computation of the formulation saving, consider again Figure 6: Intention 0 (the partial intention initially formulated by the user) and Intention 1 (a complete intention proposed via progressive refinement) take 68 and 144 characters, respectively, so the formulation saving is $(144 - 68)/144 = 53\%$.

6.2.3 Lessons learnt

With reference to the three goals of these experiments, we can conclude that:

6. https://tinyurl.com/4tnddrp8 (in French)

TABLE 5
Average number of statements per task, formulation saving, editing effort, and timing for the user experiments

	Exp1		Exp2	
	Progr. ref.	Auto-compl.	Progr. ref.	Auto-compl.
♯ statements	3	1	3	1
Form. saving	42%	40%	54%	51%
Edit effort	66%	50%	39%	48%
Form. time	80s	15s	34s	21s
Exec. time	7s	7s	6s	6s

(i) Overall, the user satisfaction with the approach is good. Some users found the syntax of complete assess statements overly complex, which confirms that giving suggestions to complete partial statements is important to make the assess operator more usable.
(ii) Progressive refinement is preferred to auto-completion from all points of view.
(iii) Both interaction modes significantly help users in the assessment process by saving, on average, about half the formulation effort.

Interestingly, the user groups involved in **Exp1** and **Exp2** behaved differently. Being domain experts, data journalists do deeper and more focused analyses. This is shown by the high percentage of edited statements in **Exp1** (Table 5): users have a clear understanding of the domain and look for specific parents and siblings; thus, they spend more time in getting insights from the comparisons suggested. Conversely, students in **Exp2** tend to perform more explorative analyses, which entail fewer edits and a "coarser" reading of the charts (i.e., users are focused more on general trends rather than on the details).

7 Related work

Our approach lies at the intersection of three active research areas: *OLAP operators*, *interactive data exploration*, and *visualization*.

OLAP comes with a large number of proposals on its foundations and operators; we refer the interested reader to an excellent survey [18]. Over the years, several additional operators have been proposed to complement the fundamental ones and have been recently classified depending on their purpose [19]: *coverage* (return patterns that cover tuples with certain values; e.g., [20]), *information* (return patterns providing information about the distribution of measure values; e.g., [21]), and *contrast* (return patterns occurring with some values but not the others). Noticeably, while coverage and information operators are focused on returning tuples representative of different parts of a cube, contrast operators entail the comparison of cube tuples — thus, they are closely connected to our approach. Specifically, among contrast operators, some variants of a *DIFF* operator have been introduced either to (i) return the set of tuples that most successfully describe the difference of values between two given cube cells [22]; (ii) group and highlight commonalities among data points [23]; or (iii) pinpoint differences between two datasets that share the same discrete attributes [24]. The EKISO algorithm [25] returns the best insights derived from aggregating a cube; insights are scored with respect to how much the returned value impacts on the aggregate and how unexpected the insight is. The *RELAX* operator verifies whether a pattern observed at a certain level of detail is present at a coarser level of detail too [26]. In the same direction of RELAX, in [27], the authors evaluate and confirm the accuracy of user insights on a given query result. Alternative operators have also been proposed in the Cinecubes approach [28] to compare the result of a given query to results over sibling values or drill-downs. The Cinecubes can be seen as a form of assessment, although neither tunable nor explicitly invoked by the user. Finally, Siddiqui et al. [29] define the Compare operator to give a clear semantics and logical foundations to general comparisons of two series of data. While assess is expressed in terms of a cube algebra and implemented as a web application, Compare is expressed in SQL and implemented within a RDBMS.

Similarly to OLAP operators, interactive data exploration aims at producing sequences of queries that can give users interesting insights, and is often complemented by visualization techniques capable of highlighting hidden patterns. Many studies focus on learning users' preferences to suggest personalized search and give recommendations [30]; this is somehow complementary to our approach, since assess could be one of the potential patterns to be learned and recommended to the users. Similarly, in [31], the authors profile the explorations of a user and use the maximum entropy principle to recommend parts of the cube can be the most surprising in a subsequent query. NextiaJD [32] aims at finding interesting and accurate joins (in other terms, at suggesting benchmarks) to bring together datasets with heterogeneous schemata. Other approaches, that can be profitably used for interactive data exploration, aim at creating *data descriptions*, i.e., explanations that make large dataset more understandable at a glance by a user. For instance, in [33] this description is generated in the form of predicates that apply to the target dataset.

As to visualization, query result comparison is achieved by showing multiple visualizations juxtaposed and highlighting the difference between them [4]. In [5], the authors return visualization sequences by grouping subsets of visualizations with shared properties (e.g., a common measure or time period) so as to minimize the cognitive cost (or perceived amount of difference) across adjacent views. Similarly, Voyager2 [34] allows users to specify multiple charts in parallel by authoring partial specifications; Voyager2 returns a chart gallery, showing all charts that satisfy the specified constraints, and uses the global minimum and maximum values of a data field to aid comparison across charts. Finally, Zenvisage [35] introduces an algebra to compose, filter, compare, and sort multiple visualizations.

Overall, the novelty of the approach based on the assess operator lies in (i) the fundamental problem it addresses, i.e., the support to a *complete* assessment process consisting of getting the benchmark, computing the comparison, and labeling the result; (ii) the adoption of a declarative syntax to hide the complexity of the assessment; and (iii) the automatic/interactive refinement of each part of this process.

We close this section by observing that, although the work presented in this paper may seem to be related to the recommendation of OLAP queries (e.g., [36]) or sessions (e.g., [37]), the goal of recommendation is quite different:

given interestingness and similarity functions and former analytic sessions, recommendation helps a user navigating data by recommending queries/sessions that are either similar to what interested her in the past or that interested similar users.

8 Conclusions

In this paper we proposed and evaluated two interaction modes aimed at supporting the assessment process by suggesting completions for partially-specified assess statements. Both modes have performances compatible with the interactivity requirement of analysis sessions. The tests with users showed a good level of user satisfaction, with a clear preference of progressive refinement over auto-completion.

Our future work on this topic will be mainly aimed at addressing the issues raised by the users during the experiments; specifically, we will investigate how to provide an effective explanation of the system's suggestions.

As a final remark, we emphasize that the assessment suggested by the auto-complete mode is typically not included in those proposed by partial refinement. Indeed, this is a predictable consequence of the approach adopted for the two modes: although the same clustering algorithm is used, auto-completion sets $k = 1$ since it aims at providing a representative assessment, while refinement sets $k > 1$ since it aims at diversification. Thus, the two modes can be eventually seen as complementary rather than competing.

References

[1] V. L. O'Day and R. Jeffries, "Orienteering in an information landscape: how information seekers get from here to there," in *Proc. INTERACT*, 1993, pp. 438–445.

[2] S. Kandel, A. Paepcke, J. M. Hellerstein, and J. Heer, "Enterprise data analysis and visualization: An interview study," *IEEE Trans. Vis. Comput. Graph.*, vol. 18, no. 12, pp. 2917–2926, 2012.

[3] K. Wongsuphasawat, Y. Liu, and J. Heer, "Goals, process, and challenges of exploratory data analysis: An interview study," *CoRR*, vol. abs/1911.00568, 2019.

[4] T. Blount, L. Koesten, Y. Zhao, and E. Simperl, "Understanding the use of narrative patterns by novice data storytellers," in *Proc. CHIRA*, 2020, pp. 128–138.

[5] J. Hullman, R. Kosara, and H. Lam, "Finding a clear path: Structuring strategies for visualization sequences," *Comput. Graph. Forum*, vol. 36, no. 3, pp. 365–375, 2017.

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

[7] M. Francia, M. Golfarelli, P. Marcel, S. Rizzi, and P. Vassiliadis, "Assess queries for interactive analysis of data cubes," in *Proc. EDBT*, 2021, pp. 121–132.

[8] M. R. Vieira, H. L. Razente, M. C. N. Barioni, M. Hadjieleftheriou, D. Srivastava, C. T. Jr., and V. J. Tsotras, "On query result diversification," in *Proc. ICDE*, 2011, pp. 1163–1174.

[9] O. B. El, T. Milo, and A. Somech, "Automatically generating data exploration sessions using deep reinforcement learning," in *Proc. SIGMOD*, 2020, pp. 1527–1537.

[10] X. Jin and J. Han, "K-medoids clustering," in *Encyclopedia of Machine Learning*, C. Sammut and G. I. Webb, Eds. Springer US, 2010, pp. 564–565.

[11] M. Francia, P. Marcel, V. Peralta, and S. Rizzi, "Enhancing cubes with models to describe multidimensional data," *Inf. Syst. Front.*, 2022, to appear.

[12] T. Haslwanter, *An Introduction to Statistics with Python: With applications in the life sciences*. Springer, 2016.

[13] R. Fagin, R. Kumar, M. Mahdian, D. Sivakumar, and E. Vee, "Comparing and aggregating rankings with ties," in *Proc. PODS*, 2004, pp. 47–58.

[14] K. Börner, *Atlas of knowledge: anyone can map*. MIT Press, 2015.

[15] E. Tufte, *The Visual Display of Quantitative Information*. Graphics Press Cheshire, 1983, vol. 2.

[16] M. Francia, E. Gallinucci, and M. Golfarelli, "COOL: A framework for conversational OLAP," *Inf. Syst.*, vol. 104, p. 101752, 2022.

[17] P. E. O'Neil, E. J. O'Neil, X. Chen, and S. Revilak, "The star schema benchmark and augmented fact table indexing," in *Proc. TPCTC*, 2009, pp. 237–252.

[18] O. Romero and A. Abelló, "On the need of a reference algebra for OLAP," in *Proc. DaWaK*, 2007, pp. 99–110.

[19] L. Golab and D. Srivastava, "Exploring data using patterns: A survey and open problems," in *Proc. DOLAP@EDBT/ICDT*, 2021, pp. 116–120.

[20] L. Golab, H. J. Karloff, F. Korn, and D. Srivastava, "Data auditor: Exploring data quality and semantics using pattern tableaux," *Proc. VLDB Endow.*, vol. 3, no. 2, pp. 1641–1644, 2010.

[21] A. Chédin, M. Francia, P. Marcel, V. Peralta, and S. Rizzi, "The tell-tale cube," in *Proc. ADBIS*, 2020, pp. 204–218.

[22] S. Sarawagi, "Explaining differences in multidimensional aggregates," in *Proc. VLDB*, 1999, pp. 42–53.

[23] F. Abuzaid *et al.*, "DIFF: a relational interface for large-scale data explanation," *VLDB J.*, vol. 30, no. 1, pp. 45–70, 2021.

[24] R. Subramonian, "Defining *diff* as a data mining primitive," in *Proc. KDD*, 1998, pp. 334–338.

[25] B. Tang, S. Han, M. L. Yiu, R. Ding, and D. Zhang, "Extracting top-k insights from multi-dimensional data," in *Proc. SIGMOD*, 2017, pp. 1509–1524.

[26] G. Sathe and S. Sarawagi, "Intelligent rollups in multidimensional OLAP data," in *Proc. VLDB*, 2001, pp. 531–540.

[27] E. Zgraggen, Z. Zhao, R. C. Zeleznik, and T. Kraska, "Investigating the effect of the multiple comparisons problem in visual analysis," in *Proc. CHI*, 2018, p. 479.

[28] D. Gkesoulis, P. Vassiliadis, and P. Manousis, "CineCubes: Aiding data workers gain insights from OLAP queries," *Inf. Syst.*, vol. 53, pp. 60–86, 2015.

[29] T. Siddiqui, S. Chaudhuri, and V. R. Narasayya, "COMPARE: accelerating groupwise comparison in relational databases for data analytics," *Proc. VLDB Endow.*, vol. 14, no. 11, pp. 2419–2431, 2021.

[30] L. Song, J. Gan, Z. Bao, B. Ruan, H. V. Jagadish, and T. Sellis, "Incremental preference adjustment: a graph-theoretical approach," *VLDB J.*, vol. 29, no. 6, pp. 1475–1500, 2020.

[31] S. Sarawagi, "User-adaptive exploration of multidimensional data," in *Proc. VLDB*, 2000, pp. 307–316.

[32] J. Flores, S. Nadal, and O. Romero, "Towards scalable data discovery," in *Proc. EDBT*, 2021, pp. 433–438.

[33] M. Paganelli, P. Sottovia, A. Maccioni, M. Interlandi, and F. Guerra, "Explaining data with descriptions," *Inf. Syst.*, vol. 92, p. 101549, 2020.

[34] K. Wongsuphasawat *et al.*, "Voyager 2: Augmenting visual analysis with partial view specifications," in *Proc. CHI*, 2017, pp. 2648–2659.

[35] T. Siddiqui, A. Kim, J. Lee, K. Karahalios, and A. G. Parameswaran, "Effortless data exploration with zenvisage: An expressive and interactive visual analytics system," *Proc. VLDB Endow.*, vol. 10, no. 4, pp. 457–468, 2016.

[36] M. Francia, M. Golfarelli, and S. Rizzi, "A-BI$^+$: A framework for augmented business intelligence," *Inf. Syst.*, vol. 92, p. 101520, 2020.

[37] A. Giacometti, P. Marcel, E. Negre, and A. Soulet, "Query recommendations for OLAP discovery-driven analysis," *Int. J. of Data Warehousing and Mining*, vol. 7, no. 2, pp. 1–25, 2011.

Matteo Francia received his Ph.D. in Computer Science and Engineering from The University of Bologna, Italy. He is an adjunct professor and a post-doc research fellow at The University of Bologna. His research focuses on advanced analytics and unconventional data, with particular reference to trajectory, social, and sensory data.

Matteo Golfarelli is full professor at the University of Bologna. He is author of over 130 publications in international journals and conferences mainly in the areas of database systems and business intelligence. His research interests include Big Data Analytics, Machine Learning, NoSQL. He is member of the steering committee of DOLAP and associate editor for DKE and Electronics journals.

Patrick Marcel is an Associate Professor at the University of Tours, France. His current research focuses on OLAP and data warehousing, recommender systems, exploratory data analysis and data narration. He authored numerous publications in international conferences and journals on these subjects. He is a member of the steering committee of DOLAP and a member of the regular editorial board of DKE.

Stefano Rizzi is a Full Professor at the University of Bologna, Italy. He has authored more than 150 papers in international journals and conferences mainly in the field of business intelligence. He is member of the steering committee of DOLAP and of the editorial board of DKE. His research interests include data warehouse design and business intelligence.

Panos Vassiliadis is a professor at the University of Ioannina, Greece. His research focuses on the rigorous modeling of data, software, and their interdependence. Currently he works in the areas of business intelligence and schema evolution. He is a senior member of the IEEE. More information is available at http://www.cs.uoi.gr/ pvassil.