Cluster Computing (2020) 23:2347–2374
https://doi.org/10.1007/s10586-019-03006-z

OLAP operators for social network analysis

Maha Ben Kraiem[1] · Mohamed Alqarni[2] · Jamel Feki[2] · Franck Ravat[3]

Received: 4 April 2019 / Revised: 12 September 2019 / Accepted: 19 October 2019 / Published online: 29 October 2019

Abstract
The multidimensional data model and implementations of social networks come with a set of specific constraints, such as missing data, reflexive relationship on fact instance. However, the conventional OLAP operators and existing models do not provide solutions for handling those specificities. Therefore, we should invest further efforts to extend these operators to take into consideration the specificities of multidimensional modeling of tweets as well as their manipulation. Face to this issue, we propose, in this paper, new OLAP operators that enhance existing solutions for OLAP analyses involving a reflexive relationship on the fact instances and dealing with missing values on dimension members. For each OLAP operator, we suggest a user-oriented definition as an algebraic formalization, along with an implementation algorithmic.

Keywords Social network analysis · OLAP operators · Null-Drilldown · Null-Rollup · Null-Select · FDrilldown · FRollup

1 Introduction

In the last decade, the data warehouse (DW) has been the backbone of decision support systems for more than 2 decades and widely accepted and used across the globe in a variety of applications. Contributions of the research community in the data-warehousing field, complimented by advancement in the relevant hardware technology, have matured these systems in managing huge volumes of data and providing their access with matchless efficiency to applications and decision-makers.

✉ Maha Ben Kraiem
Maha.BenKraiem@gmail.com

Mohamed Alqarni
alqarni@uj.edu.sa

Jamel Feki
Jfeki@uj.edu.sa

Franck Ravat
Franck.Ravat@irit.fr

[1] MIRACL Laboratory, University of Sfax, Airport Road Km 4, P.O. Box. 1088, 3018 Sfax, Tunisia

[2] University of Jeddah, CCSE, Jeddah, Saudi Arabia

[3] IRIT, University of Toulouse, 118, Route de Narbonne, 31069 Toulouse Cedex 9, France

Data warehousing has proven its success on structured data and pre-established relationships among this data, thereby achieving less performance in dealing with huge volumes of data. The OLAP technology in a data warehouse performs aggregated-oriented analyses from multiple dimensions of interest.

Social media is yet another interesting domain producing much more large data volumes that trap the attention of research and business communities. There is growing interests in gaining insights to the way social networks operate, their users behave, engage in conversations, express their opinions and influence others. This involves performing aggregations across conventional and unconventional dimensions in social media data. Furthermore, businesses can largely benefit from this new resource and market of social media. Provided underlying technology and systems of data warehousing can partially solve the challenges of data extracted from social media as their heterogeneity, semi-structured, and velocity.

In this paper, we have projected the use of conventional data warehouse systems, used in many businesses, over the data issued from social media by enabling them to store, model, operate, and consume such data. In previous work, we have proposed a multidimensional model dedicated to the OLAP of data exchanged through tweets [1]. This model takes into account the specificities of data issued from tweets. Among these specificities, we can adduce

missing data and *reflexive relationship* between tweets and their answers' tweets on fact instances. Facing to this issue, we set a twofold purpose, first increase the efficiency of analysis and, secondly, help data-analysts. For this reason, we propose a set of algebraic OLAP operators to support analyses over the proposed multidimensional model. We illustrate the execution process of each operator using an algorithm. At last, we implement the set of analysis operators in an R-OLAP framework. This framework aims at proving the feasibility of the proposed concepts and evaluating the efficiency of carrying out analyses.

We have organized this paper as follows. Section 2 studies representative works related to the OLAP operators. Section 3 introduces our motivation example and context. Section 4 describes our modeling solution for OLAP operators. For each operator, we formalize it as an algebraic definition and develop an algorithm to implement it. Section 5 provides experimental results and assessments on the efficiency of our proposed OLAP operators and finally, Sect. 6 concludes the paper.

2 Related works

The multidimensional data model and implementations of social networks come with a set of specific constraints, such as missing data, reflexive relationship between fact instances. However, the conventional OLAP operators have been defined for a classical multidimensional context assuming that data are present all the time [2] and, therefore, do not provide solutions for handling those specificities. This section provides a literature survey of the works related to the problems of missing data and reflexive relationship between fact instances.

2.1 Overview of works dealing with missing data

Data analysis of social networks is often impeded by the difficulty of missing data. Recent proposals highlight the effects of this difficulty mainly regarding the querying process. The analysis of data extracted from social networks would be severely distorted when it is limited to filled fields (i.e., not null valued fields) and ignoring missing data [3].

To solve the problems due to missing data, the literature suggests numerous ways. In [4], the authors address the problem of missing data in information cascades. A cascade is formed when information or actions spread from node to node through the social network." Given a fraction C' of the complete cascade C, they estimate the size or depth of the complete cascade C using k-tree model. The proposed model was evaluated using information propagation cascades in the Twitter network (70 million nodes). The experiments show that the k-tree model is an effective tool to study the effects of missing data in cascades [5] "proposed a data model that captures imprecise data by using the concept of data granularity. Data imprecision is handled by first testing whether the data is precise enough to answer a query. If the data is sufficiently precise, the query is evaluated and a correct, precise answer results. The authors suggested new strategies for data that lacks sufficient precision. The first strategy is to suggest an alternative query that can be answered precisely. The second strategy handles imprecision explicitly by presenting three different answers to the user. The first type of answer discards the imprecise data and uses only the known, precise data, which leads to a conservative answer. The second type, the liberal answer, includes everything that could possibly be true, which allows some imprecise data to be included. The third type of answer is a weighted answer that includes everything that might be true, but assigns heavier weights to precise data than to imprecise data. Along with the aggregate computation, a separate computation of the precision of the result is carried out.

Multiple imputation [6, 7] is a technique from statistics, where multiple values are imputed, i.e., substituted, for missing values, allowing data with some missing values to be used for analysis, while retaining the natural variance in the data.

Another imputation procedures replace missing values by plausible estimates [8]. This results in a completed dataset and gives the researcher the opportunity to proceed with the analysis using standard analysis methods and software [9]. The authors performed a simulation study to investigate the effect of non-response and missing data on the structural properties of social networks, and the ability of some simple imputation techniques to treat the missing network data.

In Ref. [10] the authors argue that new methods of gathering data issued from social networks with a large number of missing data introduce a greater degree of vagueness that require reviewing the social network analysis techniques. Hence, the authors proposed a new area in data management called "probabilistic databases", whose principal goal is to supply tools to handle and manage missing data.

Furthermore, in Ref. [11], the authors consider the problem of aggregation using an imprecise probability data model that allows representing imprecision by partial probabilities and uncertainty using probability distributions.

Other approaches are proposed for the processing of missing data. In Ref. [12], the authors have suggested methods which aim to find approximations to missing data in a dataset by using "optimization algorithms" to improve the network parameters. The optimization methods

considered are genetic algorithm, simulated annealing, particle swarm optimization, random forest and negative selection. These methods are individually used in combination with auto-associative neural networks for missing data estimation.

The concept of null values has been generalized to partial values, where one of a set of possible values is the true value. Work has been done on aggregation over partial values in relational databases [13].

According to this study, we may conclude that missing data in social networks is a long standing but the suggested solutions are relatively incomplete. Indeed, most of the works do not offer tools for the decision-maker to manipulate missing data over OLAP analyses. None of the previous work fully supports carrying out analysis when missing data are involved.

2.2 Overview of works dealing with OLAP operators on fact table

To the best of our knowledge, no solution for OLAP analysis is proposed for *Drilling Down* and *Up* on the fact in the multidimensional schema. Only few *querying* operators on fact (*Drill-Across, FRotate*) are formally proposed in Refs. [2, 14].

The *Drill-Across* operator relates information contained in two multidimensional facts having some common dimensions. According to [16]. Other authors relax this restriction, [14, 15] define the Drill-Across as replacing a currently analyzed fact F1 with another fact F2 while keeping the same analysis space (current dimensions). The authors have identified semantic relationships between dimensions and facts: Derivation, Generalization, Association and Flow to extend possibilities to Drill-across. These relationships between dimensions and facts improve the conformity between attributes and could be used to navigate or Drill-across between Star schemas, even when dimensions are not shared. The authors in [17] define the Drill-across as an extension to the natural join where the intersection of the two dimensions is aggregated at the finest grain of these dimensions. Furthermore, [18] propose extending the navigation operation Drill-across to include the non-conformed dimensions.

The *FRotate* operator defined in [2] consists in using a new fact in the multidimensional table while preserving the characteristics of the current analysis axes. The new fact must share at least the two current (i.e., displayed) dimensions with the current fact. Note that the fact rotation operation *FRotate*, is equivalent to the Drill-Across operation [15].

According to this study, we may conclude that none of these works offers tools for the decision-maker to navigate (Drilling-down, and -up) through the fact. So far, the OLAP frameworks lack the ability to cope with this problem.

To alleviate this drawback, our proposed OLAP operators namely *FDrilldown* and *FRollup* go further according to a new *Reflexive* relationship on the fact instances. These new operators allow modifying the analysis level in a fact while keeping the same analysis context (i.e., maintaining the currently dimensions for the analyzed fact). Hence, data analysts would benefit greatly from the ability to navigate and view combined multidimensional data from multiple levels of fact.

Facing to these issues, we aim at proposing a modeling solution for OLAP analysis operators. In contrast with the related work, our proposed analysis operators should facilitate decision-makers' tasks by not requiring the involved missing data. Moreover, they allow navigating down and up through the implicit hierarchical levels of the *fact*; this represents the first step to detect strong connections between fact instances and, therefore discover interesting or amazing topics and then conduct much more deep analysis of such data sets.

3 Motivation example

3.1 Multidimensional model

Referring to our multidimensional model dedicated to the OLAP of data exchanged through tweets, our motivation example relies on the '*Tweet Constellation*' proposed in [18]. This model has two facts namely *FACTIVITY-TWITTOS* and *FACTIVITY-TWEET*. The first fact corresponds to an observation on user accounts and allows the analysis of the user activity over time, whereas the *FACTIVITY-TWEET* fact is a reflexive fact. It models links between a tweet and the person concerned by the answer (answered person) and then allows participants and other readers to follow easily the exchange of tweets. Being reflexive, *FACTIVITY-TWEET* allows interconnecting instances of the same fact hierarchically. In practice, if a tweet *tr* is a reply to tweet *t*, *tr* refers *t* (it contains the ID of tweet *t*). This reflexive relationship between tweets will guarantee that every tweet response inserted to the DW corresponds to an existing tweet so that the analysis of a set of linked tweets becomes possible. Our '*Tweet Constellation*' multidimensional model is composed of five dimensions namely *DTime, DSource, DTweet-Metadata, DPlace* and *DUser*. Figure 1 depicts the extended multidimensional model for tweets. For further details on the *Tweet Constellation* model, refer to [1].

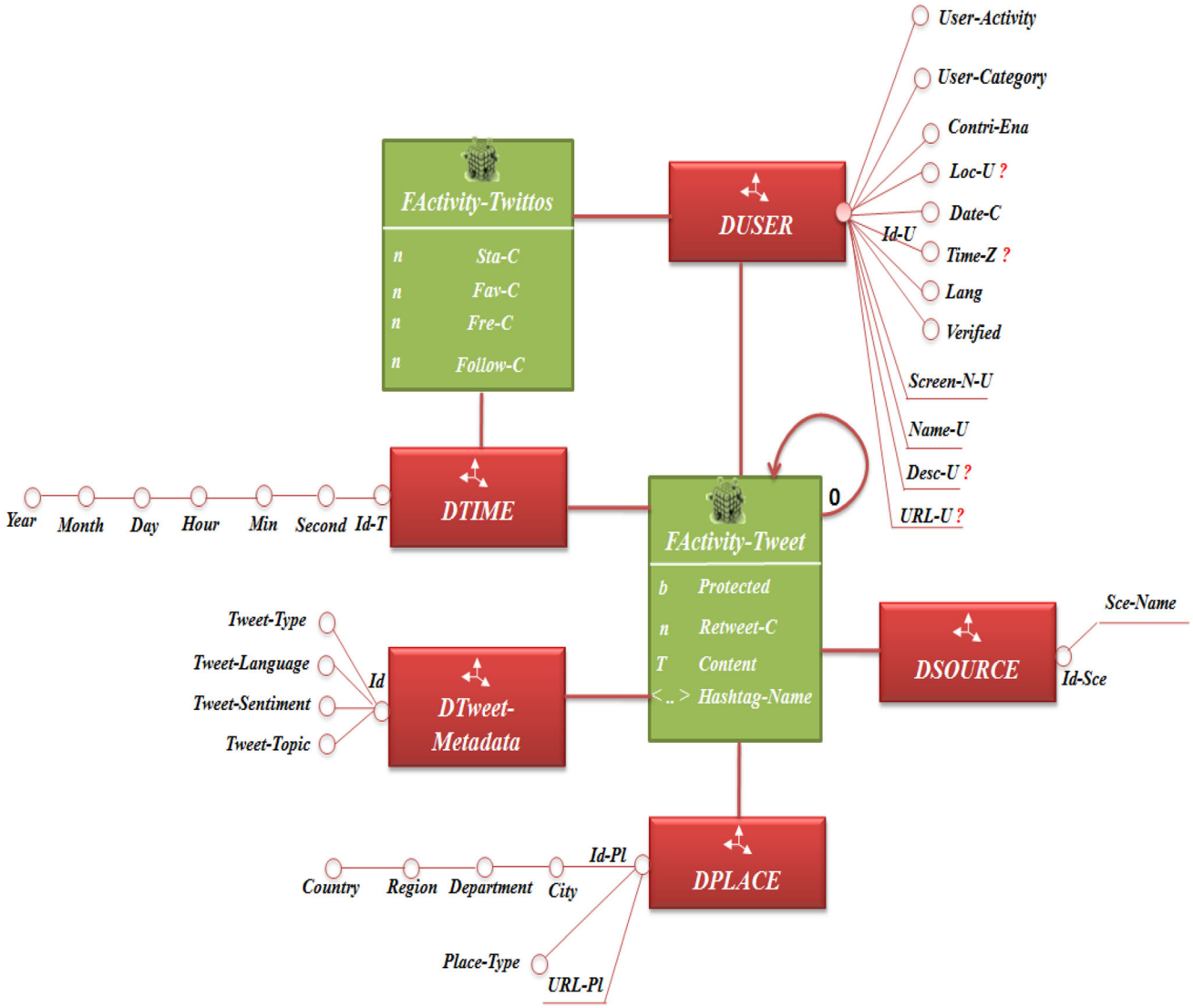


Fig. 1 Multidimensional constellation model dedicated for the OLAP of tweets

3.2 Logical modeling: R-OLAP

Once the conceptual model is defined, the logical model can be derived by applying a set of transformation rules. In this section, we define the rules to transform a constellation into R-OLAP logical model.

Although the various types of R-OLAP models, we have opted for detailing the transformation rules for the denormalized R-OLAP model. This model is the most used because it needs less join operations during queries execution. We will suggest some extensions to the denormalized R-OLAP model in order to reflect the specificities of the proposed multidimensional model (Reflexive Fact, Empty dimension, measure of the type List of elements).

We transform the proposed constellation model into R-OLAP logical model according to the following set of three rules:

Transformation Rule 1	Each dimension D will be transformed into a relational table which columns are parameters and weak attributes of all hierarchies of D. The primary key of the table is the attribute of the finest level of granularity of all hierarchies of D. For the Empty Dimension, we create one instance for which we generate a surrogate key and keep Null all its other attributes. Thus, instances of the fact may be associated with this dimension instance

Transformation Rule 2	Each fact F will be transformed into a relational table of the same name which columns are all measures of F and foreign keys referencing the dimensions connected to F. For a reflexive fact, the primary key contains an additional attribute which corresponds to the tweet identifier (Id-Twt) and a foreign key (Id-Twt-Response) which can be either null, or has a values referring an existing tweet (Id-Twt). Note that the reflexive relationship between tweets is supported by the referential integrity constraint. "It requires that a foreign key value must reference a valid, existing primary key value in the parent table". The primary key of a non-reflexive fact is the concatenation of its foreign keys.
Transformation Rule 3	Each measure of type List of elements is transformed into a relational table called T-MeasureName. The primary key of the T-MeasureName table is the concatenation of the primary key of the fact with an additional attribute (Position of a hashtag in our case).

Example: The measure *Hashtag-Name* that has the type *List of element* (in the *FACTIVITY-TWEET* Fact) becomes a relational table named *T-HashtagName*, where Id-Twt and Position of a Hasthag (in the tweet) are its primary key.

Figure 2 depicts the R-OLAP model resulted from the transformation process of the multidimensional constellation diagram of Fig. 1.

4 Extended OLAP operators for missing data

The multidimensional data model and implementations of social networks come with a set of further constraints, such as missing data. This is the case where there is simply no value provided at all. Technically, the loading process detects NULL values. Actually, the missing data problem is omnipresent in OLAP applications. Some research authors ignore null values as [3], others replace missing data by plausible estimates in the data set as in [9], etc. All the literature works related to this problem apply on the data set.

Existing OLAP operators cannot be successfully applied to handle the above-mentioned challenge. These operators have been defined in a classical context assuming that data are present all the time [2]. So, a remarkable effort should be made to extend these operators to take into consideration the specificity of multidimensional modeling of tweets (missing data).

Facing to this issue, we propose to extend three OLAP operators *Drilldown*, *Rollup* and *Select*. We call the extended versions *Null-Drilldown*, *Null-Rollup* and *Null-Select*, in order to support missing data by offering new options. The user should choose between three options, either *All*, $All_{NullLast}$, or *Flexible* option to ensure the proper performance of analysis operators. Our extended OLAP operators improved the analysis results either by reorganizing the multidimensional table moving non-significant rows to the bottom of the result table ($All_{NullLast}$), or by displaying percentages of not-null data (*Flexible* option). In fact, our approach keeps the original data set unchanged and treat the problem at the manipulation level i.e., using our proposed OLAP operators, without affecting the data warehouse content; this is very important because for some decisional-users data should not be altered and then kept as initially extracted from the social network.

For each of these OLAP operators, we propose a user oriented definition along with an algorithmic translation for its implementation.

4.1 Changing analysis granularity

We extend two operators to allow changing the granularity levels of analysis, namely Null-Drilldown and Null-Rollup, both take several arguments as input: a currently displayed *multidimensional table*, an analysis *dimension*, a *parameter* and a *Null-option argument*. During execution, these operators exploit the navigation paths defined by the currently displayed hierarchy and produce a new *multidimensional table* containing information at a different granularity levels.

4.1.1 Null-Drilldown OLAP operator

Classically, the Drilldown operator allows displaying aggregated fact's measures at a finer granularity level according to dimensions. After Drilling-down on a dimension, the decision-maker obtains a new multidimensional table where the drilled dimension displays information at a finer granularity level without changing the granularity levels of the other dimension(s).

We extend the classical Drilldown operator by taking into account the involved missing data during the analysis. The user should choose between three options, either *All,* $All_{NullLast}$, or *Flexible* option to ensure the proper performance of analysis operator. Our extended OLAP operator improved the analysis results either by reorganizing the multidimensional table moving non-significant rows at the

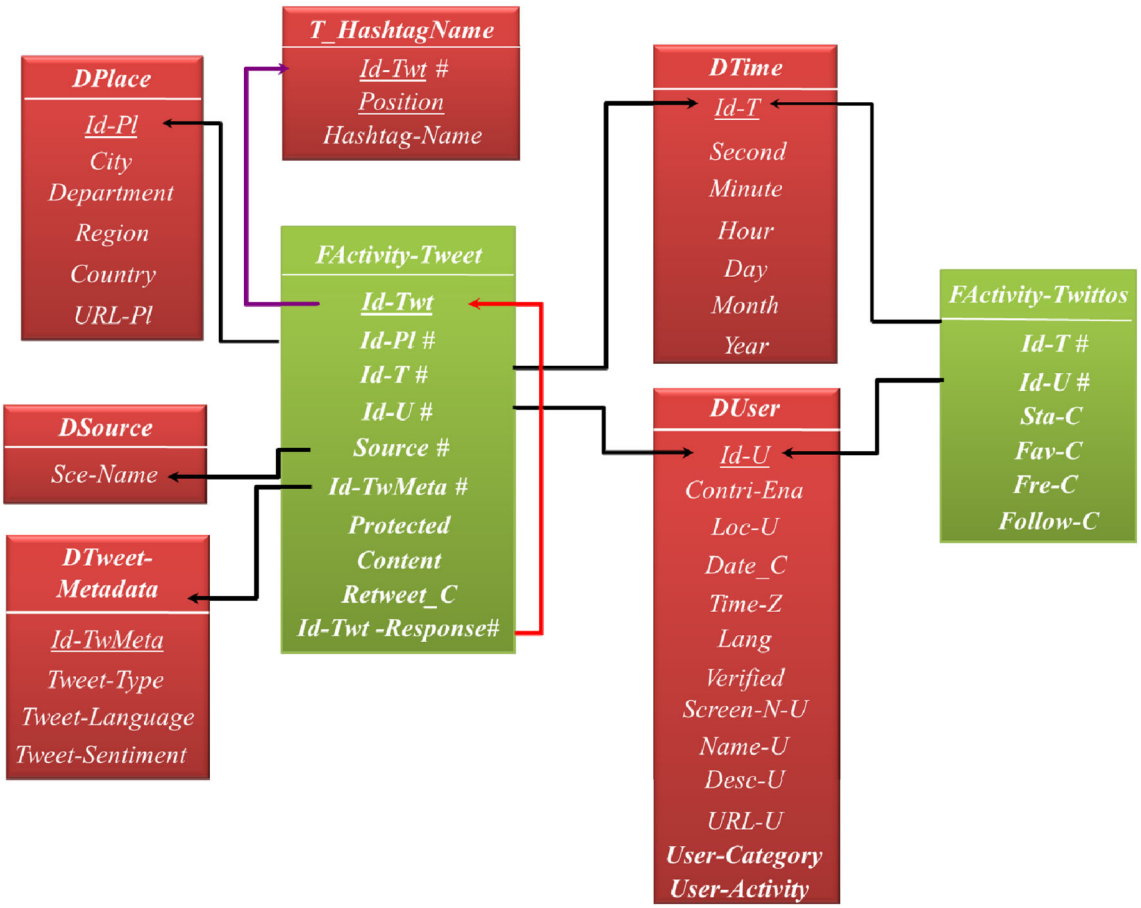


Fig. 2 R-OLAP logical model for constellation of Fig. 1

bottom of the table ($All_{NullLast}$), or by displaying percentages of not-null data (*Flexible* option).

4.1.1.1 Conceptual definition

See Table 1.

Example 1. In order to introduce, test and assess our proposed operator, we have extracted and loaded a dataset containing 25,508 tweets belonging to different geographical places. Note that, in Fig. 1, the fields of the *PLACE* dimension are not valued in all tweets.

For instance, assume that a decision-maker starts his analysis by displaying the total number of tweets according to the *Country* parameter of the *DPLACE* dimension and the *User-Activity* of the *DUSER* dimension. This displays the table MT_0 (in Fig. 3).

After executing the previous query, the decision-maker continues her/his analysis by displaying the number of tweets at a finer granularity level (*Region*) on one current displayed dimension (*DPLACE* in our running example) and without changing the granularity level *User-Activity* of the *DUSER* dimension. The analysis expression is:

$$MT_1 \leftarrow Drilldown\ (MT_0, DPLACE, Region) \quad (1)$$

Figure 4 shows the result.

Note that many *missing data* are involved in such analysis (cf. Fig. 4). Based on the results shown in MT_1, the Null-*Drilldown* operator has to adapt itself in accordance to the *null-option*. The decision-maker may receive three different versions of multidimensional tables according to the specified value for the *Null-option* in the *Null-Drilldown*.

- If the option *All* is used, we keep the analysis granularity to *Region* level. The corresponding OLAP algebraic expression is the following:

$$MT_1 \leftarrow Null - Drilldown(MT_0, DPLACE, Region, All).$$

- If the $ALL_{NullLast}$ option is used, the Drilldown returns all rows including those containing null-values of parameter P_{inf}. The rows containing null values are pushed to the end of the resulted multidimensional table. The corresponding algebraic expression is:

$$MT_2 \leftarrow Null - Drilldown(MT_0, DPLACE, Region, All_{NullLast}) \quad (2)$$

Table 1 Algebraic formalization of the *Null-Drilldown* operator

MT ← *Null-Drilldown* (MT_k , D_i , P_{inf} , *[Null-Option], [S])*	
Input	- MT_k: A multidimensional table currently displayed - D_i: One among the two dimensions displayed in MT_k - P_{inf}: A parameter of lower level than the lowest parameter displayed in the current hierarchy of D_i. - *Null-Option* ∈ {*All* | $All_{NullLast}$ | *Flexible*}: an optional parameter to indicate how null-values of parameter P_{inf} will be treated by the *Null-Drilldown*. If no option is specified, the default is *All*. - *All*: is the default option, it means that the Drilldown returns all rows / columns including those containing null values of parameter P_{inf}. - $All_{NullLast}$: The Drilldown returns all rows including those containing null-values of parameter P_{inf}. It displaces all rows / columns containing null values to the end of the result multidimensional table MT. - *Flexible*: If the percentage of null values returned for P_{inf} exceeds the threshold *S*, the operator changes the granularity level of P_{inf} in order to find a parameter *p* of lower level than P_{inf} having a percentage of null values less than *S*. A message will be posted to the user; it contains the percentage of null values for each parameter *p*. So, the user will be guided to select the adequate parameter *p* instead of P_{inf}. - *S*: An optional threshold defined by the decision-maker, it indicates the highest acceptable *percentage of null values* (*Percentage_Null_Values)* in all rows or columns in the result. *Percentage_Null_Values = Number of cells containing null values / Card (MT).*
Output	*MT* is the resulting multidimensional table.

Fig. 3 MT_0: Result multidimensional table for Example 1

FACTIVITY-TWEET|Count(Id-Twt)		DPLACE | H_Geo			
	Country	Belgium	Canada	France	Spain
DUSER | H_Act	User-Activity				
	New And Active	77	76	1266	56
	New And Passive	131	13	1230	87
	Old And Active	85	2	9177	103
	Old And Passive	-	20	13119	66
R = ∅					

Figure 5 depicts the result of expression 2.

– Using the *Flexible* option, the *Null-Drilldown* augments the granularity level of the selected parameter. Thus, an additional result displays for each parameter of level lower than *Region* the percentage of the missing values as below:

Department(Null %) : 99.16 %
City(Null %) : 9744 %

The decision-maker chooses the parameter having a percentage of missing values less than the threshold he has defined, 20% for example. Thereafter, the parameter *Region* is replaced with *City*. The corresponding analysis expression is:

Fig. 4 MT_1: Result of the *Drilldown* in expression (2)

FACTIVITY-TWEET | Count(Id-Twt)		DPLACE | H_Geo								
		Country	France						Spain	Canada	Belgium
		Region	Centre	Ile-De-France	Null	Midi-Pyrénées	Picardie	Languedoc-Roussillon	Null	Null	Null
DUSER | H_Act	User-Activity										
	New And Active		2	12	1075	38	87	57	56	76	77
	New And Passive		1	24	1241	1	2	1	87	13	131
	Old And Active		1	51	9002	77	101	35	103	2	85
	Old And Passive		7	342	12907	1	11	30	66	20	
R = Ø											

Fig. 5 MT_2: Result of the *Null-Drilldown* in expression (2)

FACTIVITY-TWEET | Count(Id-Twt)		DPLACE | H_Geo								
		Country	France						Spain	Canada	Belgium
		Region	Centre	Ile-De-France	Languedoc-Roussillon	Midi-Pyrénées	Picardie	Null	Null	Null	Null
DUSER | H_Act	User-Activity										
	New And Active		2	12	57	38	87	1075	56	76	77
	New And Passive		1	24	1	1	2	1241	87	13	131
	Old And Active		1	51	35	77	101	9002	103	2	85
	Old And Passive		7	342	30	1	11	12907	66	20	-
R = Ø											

$$MT_3 \leftarrow Null{-}Drilldown\ (MT_0,\ DPLACE, City, Flexible,\ 20\ \%) \quad (3)$$

Figure 6 is the result of expression 3. Note that the analysis of data by the *City* parameter has improved the results since most of the missing values due to two parameters *Department* and *Region* are no longer included in table MT_3. We may conclude that the extended operator has both increased the efficiency of analysis and facilitated the analysts' task by discarding more than 80% of missing data.

4.1.1.2 Logical definition The algorithm *Null-Drilldown* depicts the processing logic of the *Null-Drilldown* operator; it uses two functions defined as follows:

- *Length* (H^D, *D*): returns the number of levels in hierarchy H^D of *dimension D*
- *Level* (*p*, H^D, *D*): returns the level of parameter *p*, in hierarchy H^D of dimension *D* starting from 1 for the finest parameter.

Fig. 6 MT_3: Result of the *Null-Drilldown* in expression (3)

FACTIVITY-TWEET | Count(Id-Twt)		DPLACE | H_Geo								
		Country	France					Spain		Canada	Belgium
		City	Cergy	Nanterre	Paris	Toulouse	...	Gérone	Ripoll	St.Catharines	Tournai
DUSER | H_Act	User-Activity										
	New And Active		91	80	670	524	...	56	-	76	77
	New And Passive		110	159	567	109	...	-	87	13	131
	Old And Active		86	922	4039	362	...	103	-	2	85
	Old And Passive		98	121	6788	-	...	-	66	20	-
R = Ø											

Algorithm Null-Drilldown:

// *MT ← Null-Drilldown (MT_k , D_i , P_{inf} , [Null-Option], [S])*

Input

MT_k: Multidimensional Table

$D \in \{D_1, D_2\}$: One of the two dimensions displayed in MT_k

P_{inf}: a parameter of H^D to reach by drilling

Null-option: Indicates how null-values of parameters P_{inf} will be treated by the Drilldown

S: Optional threshold to indicate the highest acceptable percentage of null values (*Percentage_Null_Values*) in the result.

Output

MT: the result multidimensional table, with the same structure as MT_k

Begin

1. Let H^D be the actually displayed hierarchy of *D*
2. Let $Par = \{p_n, p_{n-1}, \ldots, p_c\}$ be the set of displayed parameters of H^D where *c* is the level of the finest displayed parameter of H^D, and *n* is the level of the finest parameter of H^D *(i.e., n= Length(H^D, D)), ($c \leq n$)*
3. If *Level (p_c, H^D, D,) ≤ Level (P_{inf}, H^D, D)* then
4. Impossible operation, P_{inf} has finer granularity than the displayed parameter p_c.
5. Else
6. Translate *Null-Drilldown* (MT_k; *D*; P_{inf}) into query *Q* such as
   Q = 'SELECT' || p_n, $p_{n-1}, \ldots$, P_{inf} || $f_1(m_1)$, $f_2(m_2), \ldots$|| 'FROM' || D_1, D_2, *F* || 'WHERE' || *Join conditions* || 'AND' || *Pred* || 'GROUP BY' || p_n, $p_{n-1}, \ldots$, P_{inf} || 'ORDER BY' || P_{inf}
7. *MT* = Execute (Q)
8. *Percentage_Null_Values = Number of cells containing null values of P_{inf} in MT / Card(MT)*
9. If *Percentage_Null_Values* > S then
10. If Null-option = "*Flexible*" then
11. For each parameter $p_j \in H^D$ ($1 \leq j < Level (P_{inf}, H^D, D)$)
12. ContinueDrilling = True
13. While ContinueDrilling
14. Drop table MT
15. Translate *Null-Drilldown* (MT_k; D ; p_j) into query Q such as
    Q = 'SELECT' || p_n, $p_{n-1}, \ldots$, p_j || $f_1(m_1)$, $f_2(m_2), \ldots$|| 'FROM' || D_1, D_2, *F* || 'WHERE' || *Join conditions* || ' AND ' || *Pred* || 'GROUP BY' || p_n, $p_{n-1}, \ldots$, p_j || 'ORDER BY' || p_j;
16. MT = Execute (*Q*)
17. *Percentage_Null_Values = Number of rows containing null values of p_j in MT / Card(MT)*
18. If *Percentage_Null_Values* < S then
19. Display table MT
20. ContinueDrilling = False
21. End If
22. j = j+1
23. End While
24. Else
25. Drop table MT
26. Translate *Null-Drilldown* (MT_k; *D*; P_{inf}) into query *Q* such as
    Q = 'SELECT' || p_n, $p_{n-1}, \ldots$, P_{inf} || $f_1(m_1)$, $f_2(m_2), \ldots$|| 'FROM' || D_1, D_2, *F* || 'WHERE' || *Join conditions* || 'AND' || *Pred* || 'GROUP BY' || p_n, $p_{n-1}, \ldots$, P_{inf} || 'ORDER BY' || P_{inf};
27. MT = Execute (*Q*)
28. If null-option = "$All_{NullLast}$" then
29. Translate *Null-Drilldown* (MT_k; *D*; P_{inf}, $All_{NullLast}$) into query *Q such as*
    Q = 'SELECT' || p_n, $p_{n-1}, \ldots$, P_{inf} || $f_1(m_1)$, $f_2(m_2), \ldots$|| 'FROM' || D_1, D_2, *F* || 'WHERE' || *Join conditions* || 'AND' || *Pred* || 'GROUP BY' || p_n, $p_{n-1}, \ldots$, P_{inf} || 'ORDER BY' || P_{inf} || 'DESC NULLS LAST';
30. MT = Execute (*Q*)
31. End if
32. Display MT
33. End For
34. End if
35. End if
36. End if
37. **End**

4.1.1.3 Result Now we will illustrate how the Null-Drilldown operator is transformed into SQL queries according to the chosen null-option method. The first multidimensional table MT_0 (cf. Fig. 3) is obtained by executing the following SQL code.

```
SELECT    COUNT(A.Id-Twt), U.User-Activity, P.Country
FROM      FACTIVITY-TWEET A, DUSER U, DPLACE P
WHERE     A.Id-U = U.Id-U AND A.Id-Pl = P.Id-Pl
GROUP BY  U.User-Activity,P.Country
ORDER BY  P.Country;
```

Since the chosen parameter (Region) has a higher granularity level than the displayed parameter (Country) (cf. lines 2–4), the algorithm automatically generates a query (cf. lines 6–8). In the MT, the result is aggregated according to the Region level. Then, the algorithm calculates the percentage of null values in the MT (cf. line 9). If the percentage of null values is higher than the threshold (cf. line 10), three types of queries could be generated depending on the used option during the execution of the Null-Drilldown operator.

– If the option *Flexible* is used then for each parameter having a lower level than *Region* (*Department* and *City*) the algorithm generates a multidimensional table and calculates the percentages of null values by this parameter (cf. lines 10–28). The chosen parameter *Region* is replaced by *City,* which has the minimum of missing values, to ensure that the analysis includes the minimum of null values as the analyst wants. The corresponding algebraic expression is:

$MT_3 \leftarrow Null{-}Drilldown\,(MT_0, DPLACE, City, Flexible).$

The generated query is as follows

```
SELECT   COUNT(A.Id-Twt), U.user-Activity, P.Country, P.City
FROM       FACTIVITY-TWEET A, DUSER U, DPLACE P
WHERE      A.Id-U = U.Id-U AND A.Id-Pl = P.Id-Pl
GROUP BY   U.user-Activity,P.Country, P.City
ORDER BY   P.City;
```

– Using the option *All,* the *Null-Drilldown* operator maintains the chosen granularity level to *Region*. The involved analysis expression is:

$MT_1 \leftarrow Null{-}Drilldown\,(MT_0, DPLACE, Region, All)$ (cf.lines $27-30$).

The generated query is:

```
SELECT COUNT(A.Id-Twt),U.user-Activity, P.Country, P.Region
FROM       FACTIVITY-TWEET A, DUSER U, DPLACE P
WHERE      A.Id-U = U.Id-U AND A.Id-Pl = P.Id-Pl
GROUP BY   U.User-Activity,P.Country, P.Region
ORDER BY   P.Region;
```

– According to the option $All_{null\text{-}last}$, the attribute *Region* remains unchanged. The query is transformed into the following SQL code (cf. lines 31–34).

```
SELECT COUNT(A.Id-Twt),U.User-Activity, P.country, P.Region
FROM       FACTIVITY-TWEET A, DUSER U, DPLACE P
WHERE      A.Id-U = U.Id-U AND A.Id-Pl = P.Id-Pl
GROUP BY   U.user-Activity,P.country, P.Region
ORDER BY   P.Region
DESC NULLS Last;
```

4.1.2 Null-Rollup OLAP operator

The *Null-Rollup* operator consists in moving from finer granularity data to coarser granularity data on a currently displayed dimensional hierarchy. We extend the classical *Rollup* operator by considering the missing data involved during the analysis. If there is missing data, the user should choose between three options, either *All*, $All_{NullLast}$, or

Table 2 Algebraic Formalization of the *Null-Rollup* operator

MT ← *Null-Rollup* (MT_k; D_i; P_{sup}; [*Null-Option*], [*S*]))	
Input	- MT_k: The multidimensional table currently displayed - D_i: One among the dimensions currently displayed in MT_k - P_{sup}: A parameter of higher level than the highest parameter displayed in the current hierarchy of D_i. - *Null-Option*: {*All* | $All_{NullLast}$ | *Flexible*}: An optional parameter to specify how the *Null-Rollup* will treat missing data of parameter P_{sup}. If no option is specified the default *ALL* is assumed. - *ALL*: The Rollup operator returns all the values corresponding to the parameter P_{sup} including the null-values. - $All_{NullLast}$: Idem as *ALL* with pushing to the end of the multidimensional table all rows / columns containing null values. - *Flexible*: If the percentage of null values returned for P_{sup} exceeds the threshold *S*, the Rollup operator changes the granularity level in order to find a parameter *p* of higher level than P_{sup} having a percentage of null values less than *S*. A message will be posted to the user; it contains the percentage of null values for each parameter *p*. this helps the user selecting the adequate (i.e., the minimum percentage of the null values) parameter *p* instead of P_{sup}. - *S*: Optional threshold to indicate the highest acceptable percentage of null values (*Percentage_Null_Values*) in all rows or columns in the result. *Percentage_Null_Values = Number of cells containing null values / Card (MT)*.
Output	*MT* is the resulting multidimensional table.

Fig. 7 MT_4: Result multidimensional table for Example 2

FACTIVITY-TWEET|Count(Id-Twt)		DPLACE | H_Geo					
	Region	Centre	Ile-de-France	Languedoc-Roussillon	Midi-Pyrénées	Null	Picardie
DUSER | H_Cat	User-Category						
	Friendship relationship	7	96	9	27	7436	128
	Information Seeker	3	131	11	36	6778	27
	Information Sharing	1	202	103	54	10855	46
R = ∅							

Flexible option. Table 2 shows the algebraic formalization of this extended operator.

4.1.2.1 Conceptual definition **Example 2**. The decision-maker continues her/his analysis rolling up analysis level. Suppose he intends to get the total number of tweets by *City* (*DPLACE* dimension) and *User-Category* (*DUSER* dimension). The MT_4 table in Fig. 7 shows this result.

For instance, if we continue the analysis and want to roll up from the *City* level to the *Country* level, the analysis expression is (4).

$$MT_5 \leftarrow Null{-}Rollup\,(MT_4;\ DPLACE;\ Country) \quad (4)$$

Figure 8 shows the obtained results. According to this analysis, no missing values are involved by the chosen parameter. Hence, no *null-option* method needs to be specified for *Null-Rollup* operator (cf. Algorithm Null-Rollup).

4.1.2.2 Logical definition The following algorithm describes the processing logic of the *Null-Rollup* operator.

Algorithm Null-Rollup:

// *MT* ← *Null-Rollup* (MT_k , D_i , P_{sup} , *[Null-Option], [S])*

Input

MT_k: Multidimensional table

MT: New multidimensional table, with the same structure as MT_k

$D \in \{D_1, D_2\}$ one between the two dimensions of MT_k

P_{sup}: Chosen parameter in the current H^D of dimension *D in the current* MT_k

Null-option: Indicates how null-values of parameters P_{sup} will be treated by the Rollup

S: Optional threshold to indicate the highest acceptable percentage of null values (Percentage_Null_Values) in the result.

Output

MT: the result multidimensional table, with the same structure as MT_k

Begin

1. Let H^D be the current displayed hierarchy of *D*
2. Let $Par = \{p_n, p_{n-1}, \ldots, p_c\}$ be the set of displayed parameters of H^D such as *n* and *c* are respectively the levels of the lowest and highest displayed parameter of H^D (i.e., c= Length(H^D, D)), ($c \geq n$)
3. If *Level* (p_c, H^D, D,) ≥ *Level* (P_{sup}, H^D, *D)* then
4. Impossible operation, the parameter P_{sup} is of higher granularity level than the specified parameter p_c displayed.
5. Else
6. Translate *Null-Rollup* (MT_k; *D* ; P_{sup}) into query *Q* such as

Q = 'SELECT' || $p_n, p_{n-1}, \ldots, P_{sup}$ || $f_1(m_1), f_2(m_2), \ldots$ || 'FROM' || D_1, D_2, F || 'WHERE' || *Join conditions* || ' AND ' || *Pred* || 'GROUP BY' || $p_n, p_{n-1}, \ldots, P_{sup}$ || 'ORDER BY' || P_{sup} ;

7. *MT* = Execute (*Q*).
8. *Percentage_Null_Values = Number of cells containing null values of* P_{sup} *in MT / Card(MT)*
9. If *Percentage_Null_Values* > *S* then
10. If Null-option = "*Flexible*" then
11. For each parameter $p_j \in H^D$ *(Level* (P_{sup}, H^D, *D)* < *j* ≤ *Level* (p_c, H^D, *D))*
12. ContinueDrilling = True
13. While ContinueDrilling
14. Drop table *MT*
15. Translate *Null-Rollup* (MT_k; *D* ; p_j) into query *Q* such as

Q = 'SELECT' || $p_n, p_{n-1}, \ldots, P_{sup}$ || $f_1(m_1), f_2(m_2), \ldots$ || 'FROM' || D_1, D_2, F || 'WHERE' || *Join conditions* || ' AND ' || *Pred* || 'GROUP BY' || $p_n, p_{n-1}, \ldots, P_{sup}$ || 'ORDER BY' || P_j;

16. *MT* = Execute (*Q)*
17. *Percentage_Null_Values = Number of rows containing null values of pj in MT / Card(MT)*
18. If *Percentage_Null_Values* < *S* then
19. Display table *MT*
20. ContinueDrilling = False
21. End If
22. j = j+1
23. End While
24. Else
25. Drop table MT
26. Translate *Null-Rollup* (MT_k; *D* ; P_{sup}) into query *Q* such as

Q = 'SELECT' || $p_n, p_{n-1}, \ldots, P_{sup}$ || $f_1(m_1), f_2(m_2), \ldots$ || 'FROM' || D_1, D_2, F || 'WHERE' || *Join conditions* || ' AND ' || *Pred* || 'GROUP BY' || $p_n, p_{n-1}, \ldots, P_{sup}$ || 'ORDER BY' || P_{sup} ;

27. *MT* = Execute (*Q)*
28. If null-option = "$Al_{lNullLast}$" then
29. Translate *Null-Rollup* (MT_k; *D* ; P_{sup}, $All_{NullLast}$) into query *Q such as*

Q = 'SELECT' || $p_n, p_{n-1}, \ldots, P_{sup}$ || $f_1(m_1), f_2(m_2), \ldots$ || 'FROM' || D_1, D_2, F || 'WHERE' || *Join conditions* || 'AND' || *Pred* || 'GROUP BY' || $p_n, p_{n-1}, \ldots, P_{sup}$ || 'ORDER BY' || P_{sup} || 'DESC NULLS LAST';

30. *MT* = Results of query *Q*
31. End if
32. Display MT
33. End For
34. End if
35. End if
36. End if
37. **End**

Fig. 8 MT_5: Result of the *Null-Rollup* in expression (4)

FACTIVITY-TWEET|Count(Id-Twt)		DPLACE | H_Geo			
	Country	Belgium	Canada	France	Spain
DUSER | H_Cat	User-Category				
	Friendship relationship	77	96	1885	56
	Information Seeker	131	13	3730	153
	Information Sharing	85	2	19207	103
R = ∅					

4.2 Adding restriction predicates

During a decisional process, the *Select* operator allows removing fact and dimension instances that do not satisfy some specified selection criteria. Like other analysis operators, the *Select* operator cannot avoid involving missing data. For this reason, we define a new version of the classical *Select* operator. In case of missing data are involved by the selection criteria, the user should choose between three options, either *All*, $All_{NullLast}$, or *Flexible* option to ensure good performance of the analysis.

4.2.1 Null-Select OLAP operator

Facing a large volume of missing data, we propose an extension for the conventional Select operator called *Null-Select*, which allows removing dimension instances containing null values from the result multidimensional table. Table 3 shows the algebraic formalization of the extended operator.

4.2.1.1 Conceptual definition **Example 3**. The formerly obtained MT_1 (cf. Fig. 4) shows the total number of tweets

Table 3 Algebraic Formalization of the *Null-Select* operator

MT ← Null-Select (MT_k; Pred; [Null-Option], [S])	
Input	- MT_k: A multidimensional table currently displayed - *Pred* : A set of restriction predicates to apply on the dimensions or the fact in MT_k - *Null-Option*: {*All* | $All_{NullLast}$ | *Flexible*}: an elective parameter to specify how missing data involved by the restriction predicates will be treated by the *Null-Select*, the default is *ALL*. - *ALL*: The *Select* will return all the values corresponding to the restriction predicates including the null-values. - $All_{NullLast}$: Idem as *ALL* with pushing to the end of the multidimensional table all rows / columns containing null values. - *Flexible*: If the percentage of null values involved by each restriction predicate exceeds the threshold *S*, the operator will return only the non-null values corresponding to this predicate. A message posted to the user displays the percentage of null values for each restriction predicate. However, if there is too many missing values we remove certain restriction predicates. As an extreme example, the set of restriction predicates in *MT* remains unchanged (the same as MT_k). - *S*: Optional threshold to indicate the highest acceptable percentage of null values (*Percentage_Null_Values*) in all rows or columns in the result. *Percentage_Null_Values = Number of cells containing null values / Card (MT)*.
Output	MT is the resulting multidimensional table.

Fig. 9 MT_6: Result of the *Null-Select* in expression (5)

FACTIVITY-TWEET | Count(Id-Twt)		DPLACE | H_Geo
		Region
DUSER | H_Act	User-Activity	
	New And Active	1075
	New And Passive	1241
	Old And Active	9002
	Old And Passive	12907
Pred = *PLACE. Region!= 'Ile-De-France' ∧ count(Id-Twt) > 500*		

Fig. 10 MT_7: Result of the *Null-Select* in expression (6)

FACTIVITY-TWEET | Count(Id-Twt)		DPLACE | H_Geo			
		Region	Centre	Languedoc-Roussillon	Midi-Pyrénées	Picardie
DUSER | H_Act	User-Activity					
	New And Active		2	57	38	87
	New And Passive		1	1	1	2
	Old And Active		1	35	77	101
	Old And Passive		7	30	1	11
Pred = *PLACE. Region!= 'Ile-De-France'*						

by the parameters *Region* (*DPLACE* dimension) and *User-Activity* (*DUSER* dimension). If we want to explore the number of tweets where *Region* is different from '*Ile-De-France*' (***Pred*$_1$**) and the total number of tweets by region exceeds 500 tweets (***Pred*$_2$**), thus this analysis can be performed with a *Null-Select* operator including *two* restriction predicates. The OLAP analysis is calculated according to the algebraic expression (5):

$$MT_6 \leftarrow Null{-}Select\ (MT_1; DPLACE.\ Region! = \text{'}Ile{-}De{-}France\text{'} \wedge\ count\ (Id{-}Twt) > 500,\ All \tag{5}$$

Figure 9 shows the result of expression (5).

The result table contains only null values, they are due to the selection condition ***Pred*$_2$** ('*count (Id-Twt) > 500'*).

Note that using the *Flexible* option, this analysis becomes as in expression (6).

$$MT_7 \leftarrow Null{-}Select\ (MT_1;\ PLACE.\ Region! = \text{'}Ile{-}De{-}France\text{'},\ Flexible) \tag{6}$$

Therefore, even the null values returned by ***Pred*$_1$** are removed and a message is posted to the decision-maker, the message will contain the percentage of the null values removed (cf. Fig. 10).

4.2.1.2 Logical definition The algorithm *Null-Select* describes the processing logic of the *Null-Select* operator.

Algorithm Null-Select:

// *MT* ← *Null-Select* (MT_k, *Pred*, [*Null-Option*], [*S*])

Let $Pred_k$ be the set of restriction predicates on MT_k

Input

MT_k: A multidimensional table

Pred: a set of restriction predicates to apply on dimensions.

Null-option: Indicates how null-values will be treated by the *Select*

S: Optional threshold to indicate the highest acceptable percentage of null values (Percentage_Null_Values) in the result.

Output

New multidimensional table *MT*, with the same structure as MT_k

Declare function *Generate_* Query*_Select (MT; Pred, [Null-option]*)

Q = 'SELECT' || p_n, p_{n-1},..., p_1 || $f_1(m_1)$, $f_2(m_2)$,...|| 'From' || D_1, D_2, F || ' WHERE' || Join condition || 'AND' || *Pred* || 'GROUP BY' || p_n, p_{n-1},..., p_1;

Return Q

Begin

1. For each $pred_j \in Pred$
2. *Q* = *Generate_* Query*_Select* (MT_k; $Pred_j$)
3. *MT* = Execute(Q)
4. *Percentage_Null_Values = Number of cells containing null values in MT / Card(MT)*
5. If *Percentage_Null_Values = 100* then
6. *Pred = Pred \ [predj]*
7. *Display a warning message about the deletion of* $Pred_j$
8. Else
9. If *Percentage_Null_Values > S* then
10. If *Null-option = "Flexible"* then
11. Drop table *MT*
12. *Q = Generate_* Query*_Select* (MT_k*; Pred, Flexible*)
13. *MT = Execute (Q) \ Null_Values*
14. Display table *MT*
15. Else
16. Drop table *MT*
17. *Q = Generate_*Query*_Select* (MT_k; Pred)
18. *MT* = Execute (*Q*)
19. If *null-option = "*$All_{NullLast}$*"* then
20. *Q = Generate_*Query*_Select* (MT_k; *Pred,* $All_{NullLast}$)
21. *MT* = Execute (*Q*)
22. End if
23. Display *MT*
24. End if
25. End if
26. End if
27. End For
28. End

5 OLAP operator for reflexive relationship on the fact

Previously, in ref. [1], we have proposed an extension for the classic concept of fact by adding a reflexive relationship between fact instances. This fact-to-fact reflexive relationship allows connecting an instance of the fact to one or several instances of the same fact. Based on this relationship, the fact instances are linked at many successive levels. In practice, if a tweet is a reply tweet, it contains the ID of a previous tweet. This reflexive relationship will guarantee that every tweet response inserted to the data warehouse corresponds to an existing tweet so that the analysis of linked tweets becomes possible.

Table 4 shows a set of seven reflexive tweets from the fact *FACTIVITY-TWEET*.

Table 4 Sample of seven interconnected instances of tweets from *FACTIVITY-TWEET*

N	ID-Twt	Content	Id-Twt-Response	LEVEL
1	946077853262778373	Tu sais que tu n'as rien foutus de ta journée quand ton AppleTV te demande si tu es encore là. #BingeWatching #NoLife	-	**1**
2	946078190027661312	"@lolfr C'est bien aussi :) T'as vu quoi ?"	946077853262778373	**2**
3	946084024375750657	@cegron On my list. J'ai jusqu'au 31 pour rattraper mon retard.	946078190027661312	**2**
4	946078475923935232	@cegron The Punisher. J'étais en retard. ¿	946078190027661312	**3**
5	946078699283206145	"@lolfr Moins que moi, alors. Je n'en suis qu'au 8" .	946078475923935232	**4**
6	946079260711768064	@cegron Ah mais j'ai pas fini. Épisode 6 seulement. ¿	946078699283206145	**5**
7	946080193910853633	"@lolfr I.N.E.X.C.U.S.A.B.L.E ¿ Tu as le spécial noël de Doctor Who aussi"	946079260711768064	**6**

In the example in Table 4, we distinguish six hierarchical levels. The first level corresponds to the tweet at line 1. The second level corresponds to tweets at lines 2 and 3, which are responses to the same tweet in line 1. Finally, tweets from lines 4–7 correspond, respectively, to levels 3, 4, 5 and 6 (cf. Fig. 11). Hence, we may notice that due to the reflexive relationship on fact instances, the fact is composed of hierarchical data at multiple levels and allows a decision-maker to navigate between levels. Using levels in OLAP offers further alternatives analyses since it provides users with the flexibility to view data from different perspectives.

Tweets responses can be understood both as a form of information broadcasting and as a structure where people can be part of a conversation. One interpretation of Twitter's value derives from the real-time nature of its conversations which necessarily form trending topics.

However, classical OLAP algebra does not provide solutions for handling an intuitive navigation between levels of the fact's instances based on the reflexive relationship. Hence, we define in this section two new OLAP operators called *FDrilldown* and *FRollup*. They allow navigating through the hierarchical levels of the fact, in order to analyze a measure with more or less precision. Drilling upwards (*FRollup*) consists in displaying the data

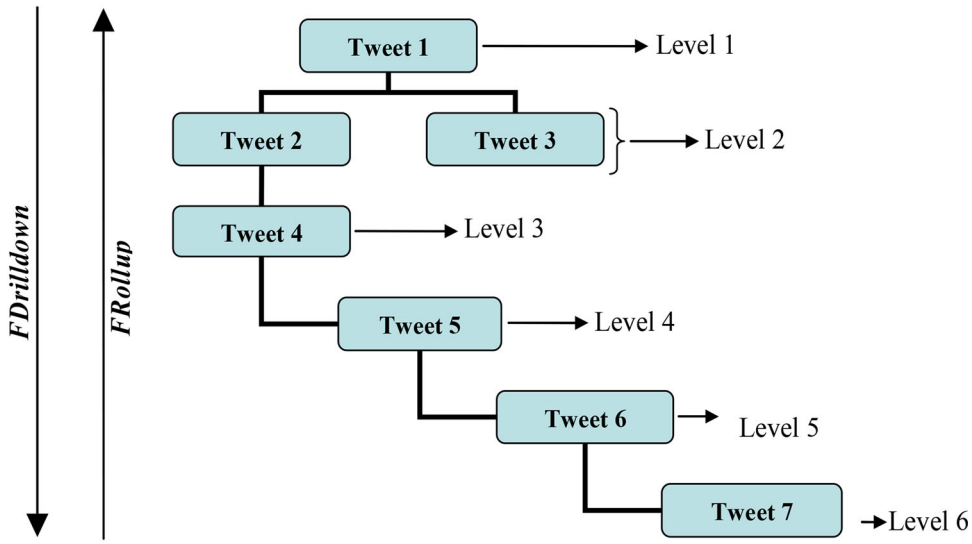


Fig. 11 Hierarchy of levels of tweets listed in Table 4

Table 5 Algebraic formalization of the *FDrilldown* operator

$MT \leftarrow FDrilldown\ (MT_k, F, Lvl_{inf})$	
Input	– MT_k: A multidimensional table currently displayed
	– F: ***Reflexive*** fact analyzed in MT_k and on which the drilling operation will be applied
	– Lvl_{inf} : A level lower than the displayed level in the current fact F
Output	MT is the result multidimensional table

with a coarser level of detail. Drilling downwards (*FDrilldown*) consists in displaying the data with a finer level.

The proposed operators are well suited to decision making applications since they can produce an output that leads to many different kinds of analyses. They allow identifying topics that have elicited a significant number of responses; these topics can be more investigated/explored using sophisticated tools based on "Text Mining" techniques. Thus, we can extract knowledge from tweets and strengthen more semantics.

5.1 FDrilldown operator

The *FDrilldown* operator applies to a reflexive fact; it consists in moving from coarser-level data to finer level data within the same analyzed fact. This drilling is possible due to the presence of the reflexive relationship on fact instances. Next, we give the *FDrilldown* algebraic formalization.

5.1.1 Conceptual definition

See Table 5

5.1.2 Logical definition

The *FDrilldown* algorithm develops the logic of the *FDrilldown* operator. Note that "For each row r in the result set, the keyword LEVEL returns the depth in the hierarchy (hierarchical level) of the node represented by row r. The LEVEL of the root node is 1, the LEVEL of an immediate child of the root node is 2, and so on".[1]

Algorithm FDrilldown:
// $MT \leftarrow FDrilldown\ (MT_k, F, Lvl_{inf})$

Input
MT_k: Multidimensional table
F: Reflexive fact analyzed in MT_k, on which the drilling down operation is applied.
Lvl_{inf} *: A* level of F, to be reached by the *FDrilldown*

Output
New multidimensional table MT, with the same structure as MT_k

Begin
1. Let Levels = $\{Lvl_n, Lvl_{n-1},\ldots,Lvl_c\}$ be the set of displayed levels of F with Lvl_c is the finest level, and Lvl_n is the highest level $(c \leq n)$
2. Let NB be the number of levels in the reflexive fact F in MT_k
3. Query-Level = **'SELECT MAX (LEVEL) FROM'** || F || **'CONNECT BY PRIOR'** || child_expr = parent_expr (i.e., *Id-Twt = Id-Twt-Response*);
4. NB = Result of *Query-Level*
5. If $Lvl_c \leq Lvl_{inf}$ OR $Lvl_{inf} \geq NB$ then
6. // Impossible Drilling operation
7. Else
8. Translate *FDrilldown* $(MT_k; F, Lvl_{inf})$ into query Q such as
Q = 'SELECT LEVEL' || $p_n, p_{n-1},\ldots, p_1$ || $f_1(m_1), f_2(m_2), \ldots.$ || 'FROM' || D_1, D_2, F || ' WHERE' || Pred || 'AND' || D_1.primary key = F.foreign key-D_1 || 'AND' || D_2.primary key = F.foreign key-D_2 || ' AND LEVEL = ' || Lvl_{inf} || 'CONNECT BY PRIOR' || child_expr = parent_expr (i.e., *Id-Twt = Id-Twt-Response*) || 'GROUP BY' || LEVEL, $p_n, p_{n-1},\ldots, p_1$;
9. MT = Execute (Q).
10. Display MT
11. End if

End

[1] https://www.enterprisedb.com/docs/en/9.5/eeguide/EDB_Postgres_Enterprise_Guide.1.036.html.

Fig. 12 MT_8: Result multidimensional table for Example 4

FACTIVITY-TWEET | Count(Id-Twt)		DPLACE | H_Geo				
		Country	Belgium	Canada	France	Spain
DTWEET-METADATA | H_Sen	Tweet-Sentiment					
	Positive		167	46	10885	172
	Negative		41	28	6279	97
	Neutral		85	17	7658	43
R = Ø						

Fig. 13 MT_9: Result of the expression 7

FACTIVITY-TWEET | Count(Id-Twt)		DPLACE | H_Geo		
		Country	Belgium	France
DTWEET-METADATA | H_Sen	Tweet-Sentiment			
	Positive		37	985
	Negative		9	392
R = Ø				

Table 6 Algebraic formalization of the FRollup operator

$MT \leftarrow FRollup\ (MT_k, F, Lvl_{sup})$	
Input	– MT_k: A multidimensional table currently displayed
	– F: is a reflexive fact, on which the FRollup operation is applied
	– Lvl_{sup} is a coarser-graduation level on the current fact
Output	MT is the resulting multidimensional table

Example 4. We illustrate how the *FDrilldown* executes on an example of analysis. Suppose that we wish to count the number of tweets (*Count (Id-Twt)*) by *Tweet-Sentiment (of the DTWEET-METADATA* dimension) and by *Country* (of the *PLACE* dimension). As a result for this requirement, we obtain the multidimensional table MT_8 shown in Fig. 12. Each cell in MT_1 gives the number of tweets for each combination of *Country* and *Tweet-Sentiment.*

To the extent that this sample is representative, most conversations that occur in Twitter appear to be dyadic exchanges of three to six messages. For this reason, based on the results depicted in MT_8 the decision-maker intends to restrict the analysis to tweets that tie in *level 6*. In fact, his aim is to move deeper into a chain of data, from high-level information to more detailed information. Hence, data pertaining to fact can then be pre-summarized and then be available for more analyses (number of intense conversation…). This OLAP analysis is calculated using the algebraic expression (7) that produces the multidimensional table in Fig. 13.

$$MT_9 \leftarrow FDrilldown(MT_8, FACTIVITY - TWEET, 6) \quad (7)$$

This result will allow the analyst to get interesting values from topics that have elicited more responses, performing specialization if more details are needed and, finally gleaning valuable insights about the way of propagation of data within each level. It also allows identifying where relevant tweets originate from.

5.2 FRollup operator

The *FRollup* operator is the reverse of *FDrilldown*, it consists in moving from a finer level to a coarser level on a currently displayed fact based on fact instances linked through the reflexive relationship (Tweet Response–Tweet). Each tweet may be connected to n $(n \geq 0)$ tweets responses within the same fact.

5.2.1 Conceptual definition

See Table 6

5.2.2 Logical definition

The algorithm *FRollup* develops the logic of the *FRollup* operator.

Algorithm FRollup:
// $MT \leftarrow FRollup((MT_k, F, Lvl_{sup})$

Input
MT_k: A multidimensional table
F: The fact actually analyzed in MT_k, on which the rolling up operation will apply. The relationship between the instances of the analyzed fact must be reflexive.
Lvl_{sup}: A level of F to reach by the *FRollup*.
Output
MT: Result multidimensional table having the same structure as MT_k
Begin
1. Let Levels = {Lvl_n, Lvl_{n-1},…,Lvl_c} be the set of levels displayed for F; *n* and *c* are respectively the lowest and highest levels ($c \geq n$)
2. Let *NB* be the number of levels in the fact F analyzed in MT_k
3. Query-Level = **'SELECT' MAX (LEVEL) FROM'** || F || **'CONNECT BYPRIOR '** || child_expr = parent_expr (i.e., *Id-Twt = Id-Twt-Response*;
4. *NB* = Result of *Query-Level*
5. If $Lvl_c \geq Lvl_{sup}$ OR $Lvl_{sup} \geq NB$ then
6. // Impossible *FRollup* operation
7. Else
8. Translate *FRollup* (MT_k; *F*, Lvl_{sup}) into query *Q* such as
Q = 'SELECT LEVEL,' || p_n, p_{n-1},…, p_1 || $f_1(m_1)$, $f_2(m_2)$, …. || ' **FROM '** || D_1, D_2, F || 'WHERE' || Pred || 'AND' || D_1.primary key = F.foreign key-D_1 || 'AND' || D_2.primary key = F.foreign key-D_2 || 'AND LEVEL =' || **Lvl_{sup}** || ' CONNECT BY PRIOR' || child_expr = parent_expr (i.e., *Id-Twt = Id-Twt-Response* || 'GROUP BY' || LEVEL, p_n, p_{n-1},…, p_1;
9. MT = Execute (Q).
10. Display MT
11. End if
End

Example 5: Assume the decision-maker starts his analysis by displaying the number of tweets at *level 3* by *Sce-Name* (*source Name* on the *DSource* dimension) and by *User-Category* (on *DUSER* dimension). He obtains a multidimensional table as in Fig. 14. Each cell represents the number of tweets of level 3 for a given *Source Name* and a given *User Category*.

Suppose the decision-maker carries out the same analysis described in the example 5, but he puts less emphasis on the depth of involved level (*Level 3*) (cf. Fig. 14), the decision-maker continues by rolling up analysis level. This time he expects to get the number of tweets at *level 2*. The corresponding analysis expression is (8):

$$MT_{11} \leftarrow FRollup(MT_{10}, FACTIVITY - TWEET, 2) \quad (8)$$

Figure 15 shows the obtained result within level 2.

According to this result, we may conclude that the number of tweets for Information-Sharing category and Twitter for iPhone as well as Twitter for Android is important. In fact, Information-sharing users post news and tend to have a large base of "followers" and answers about that news.

6 Experimental results

In order to evaluate our proposals, we have developed a software prototype called *OLAP4Tweet* [1], developed using Java and ORACLE 10 g.

The architecture of *OLAP4Tweet* is composed of four modules namely: Data source, ETL, Data warehouse, and Analysis. Figure 16 depicts the architecture of the proposed prototype.

- Data source: "The data source is represented by the available Twitter APIs [19] for data streaming. In fact, the Streaming API provides real-time access to tweets in sampled and filtered form. This API is HTTP based

Fig. 14 MT_{10}: Result multidimensional table for Example 5

FACTIVITY-TWEET| Count(Id-Twt)		DUSER | H_Cat			
		User-Category	Friendship relationship	Information Seeker	Information Sharing
DSOURCE | H_Sce	Sce-Name				
	Tendances France		-	-	58
	Tweetbot for ioS		2	-	4
	Tweetbot for Mac		-	2	1
	Twitter for Android		37	33	65
	Twitter for iPad		-	-	2
	Twitter for iPhone		-	75	172
	Twitter Web Client		25	13	36
R = Ø					

Fig. 15 MT_4: Result of expression 8

FACTIVITY-TWEET| Count(Id-Twt)		DUSER | H_Cat			
		User-Category	Friendship relationship	Information Seeker	Information Sharing
DSOURCE | H_Sce	Sce-Name				
	Tendances France		3	17	103
	Tweetbot for ioS		4	2	12
	Tweetbot for Mac		5	3	3
	Twitter for Android		157	176	208
	Twitter for iPad		1	4	8
	Twitter for iPhone		256	250	481
	Twitter Web Client		93	66	97
R = Ø					

GET, POST, and DELETE requests can be used to access the data. In Twitter terminology, individual messages describe the "status" of a user. The Streaming API allows near real-time access to subsets of public status descriptions, including replies and mentions created by public accounts. The dataset delivered by the Twitter Streaming API is a semi-structured data file conform to JSON (JavaScript Object Notation) output format. Each tweet is streamed as an object containing 67 data fields. This module may include additional external sources, such as AlchemyAPI, language recognition systems for enriching the metadata and the contents of the streamed tweet.

- Extract-Transform-Load (ETL): "This module encompasses processes required to extract data from the Twitter platform, transforms them according to the target schema and then uploads them into the data warehouse. The ETL works as the staging area for the extracted data where data are harmonized according to the data warehouse's standards. Issues of data inconsistency, noise, incompleteness, format incoherence, etc., are addressed in this module and data is integrated in a unified and consistent format, ready to be supplied to the data warehouse and OLAP applications. In fact, mapping semi-structured data to multidimensional cubes is generally a challenging task since the original format admits heterogeneity while the target one enforces a rigid structure. In case of the Twitter stream, the degree of heterogeneity is rather low and affects only few data fields. Therefore, we have transformed semi-structured data into a fixed schema before loading it into the data warehouse by extracting specific attributes from the semi-structured data at the loading time. The ETL module remains transparent to the end-user and applications."
- Data Warehouse: Preprocessed and transformed data are stored in "the Data Warehouse. The Twitter data model results in a multidimensional constellation

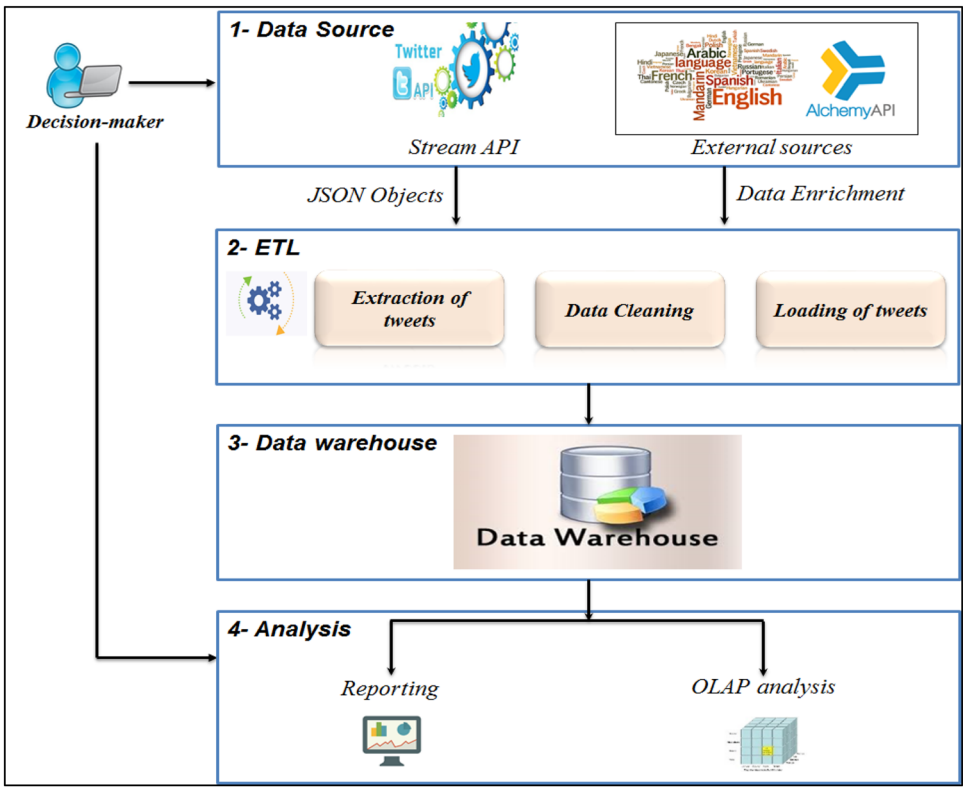


Fig. 16 Architecture of the *OLAP4Tweet* prototype

schema "Tweet Constellation" having two facts and four dimensions."

- Analysis: "This module enables frontend users to query, analyze the data and present the analyses in an adequate form for better interpretation and decision making. In fact, once the multidimensional model is generated and loaded with data, the decision-maker can generate reports and perform OLAP analyses on tweets using our proposed OLAP operators. We have developed a user friendly graphical interface to facilitate the interactions between a decision-maker and the analysis framework. It allows expressing an analytical query and visualize its result in tabular and graphical forms."

We have loaded a dataset containing 71,739 tweets collected by crawling 2 h of public tweets (from Fri Dec 22 10:48:50 UTC 2017 to Fri Dec 22 12:48:50 UTC 2017). These tweets are written in different languages. Once we load the "Tweet Constellation" multidimensional model with data, we can express and execute OLAP queries. For this purpose, we include a user-friendly decisional process in our analysis framework. A decision-maker starts his/her analysis by exploring the proposed model through an interface. (S)he selects measures and attributes related to their analysis needs by clicking and then an SQL code is generated. Queries involved in the experimental assessments aggregate the measure through the *COUNT* aggregation function. Finally, the interface provides the decision-maker with a dashboard interface representing the analysis result in tabular and graphical forms (cf. Fig. 17).

In order to illustrate how our OLAP operators perform, we carry out a set of analyses on our previous example through the graphical interface of the framework.

First analysis: *Number of tweets by User-Activity and Country.*

The user starts an analysis by displaying the *Number of tweets* per *User-Activity* and *Country*. (S)he chooses the fact that contains a measure standing for the *Number of tweets* as well as the attributes *User-Activity* and *Country* via the interface. At the end of the analysis, the prototype produces the interface shown in Fig. 18.

Based on this analysis, the decision-maker continues to analyze the *number of tweets* at a finer granularity level (*Region*) on one currently displayed dimension (*PLACE* dimension in our case) to get more detailed information and with keeping the granularity level *User-Activity* of the *USER tweet* dimension. A list of radio buttons allow the decision-maker to choose one or several OLAP operators. (S)he firstly selects the corresponding operator, *Drilldown* in our case. However, the analysis framework detects many *missing data.* To deal with the missing data, it displays a warning message and proposes to the decision-maker to choose a *null-option* method (cf. Fig. 19).

Once the analyst click on "*OK*", a new interface is displayed and containing three null-option methods (*All*,

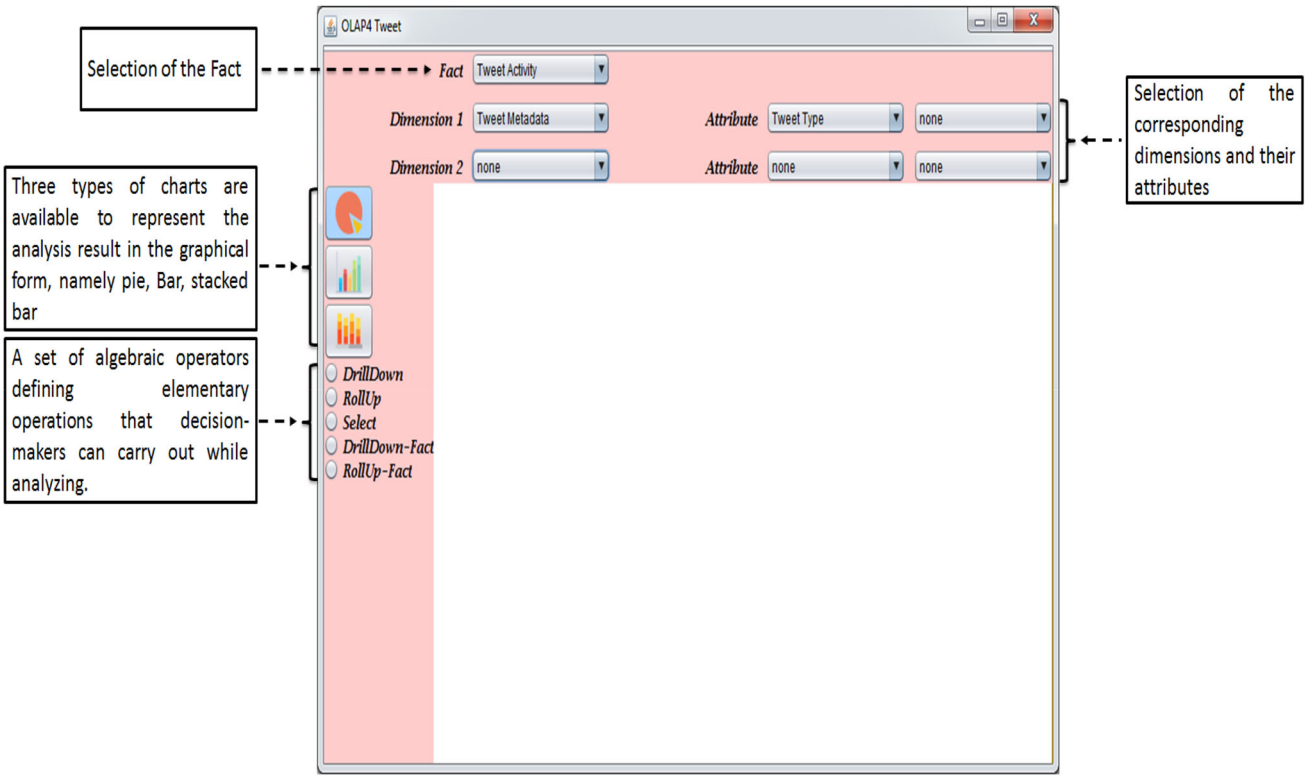
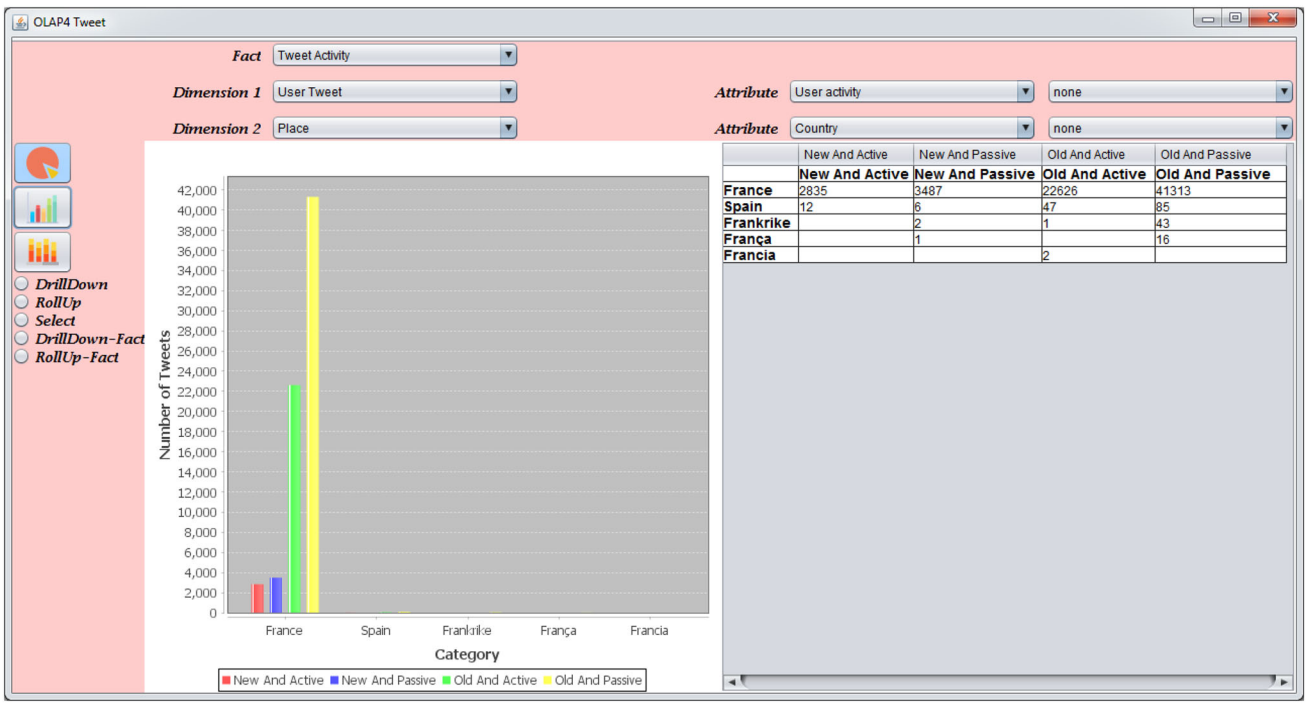


Fig. 17 Graphical interface of *OLAP4Tweet*



	New And Active	New And Passive	Old And Active	Old And Passive
France	2835	3487	22626	41313
Spain	12	6	47	85
Frankrike		2	1	43
França		1		16
Francia			2	

Fig. 18 Number of tweets by user-activity and by country

$All_{NullLast}$ and *Flexible*). Selecting the option *All*, the framework continues the analysis with the chosen attribute *Region* level. The result is directly displayed as table and graphics (cf. Fig. 20).

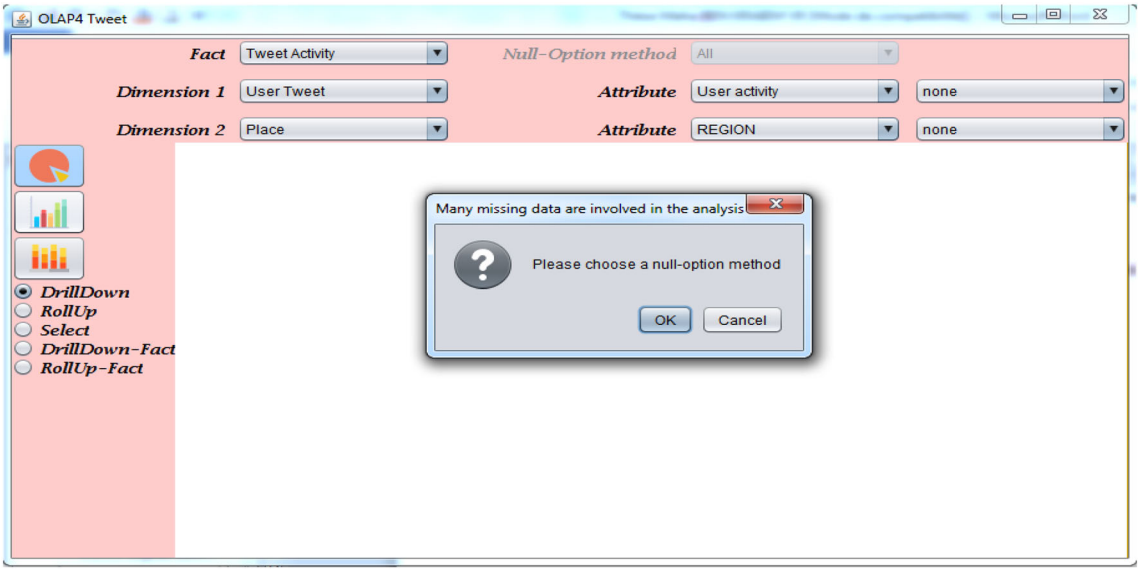


Fig. 19 Choice a Null-option method to deal with missing data

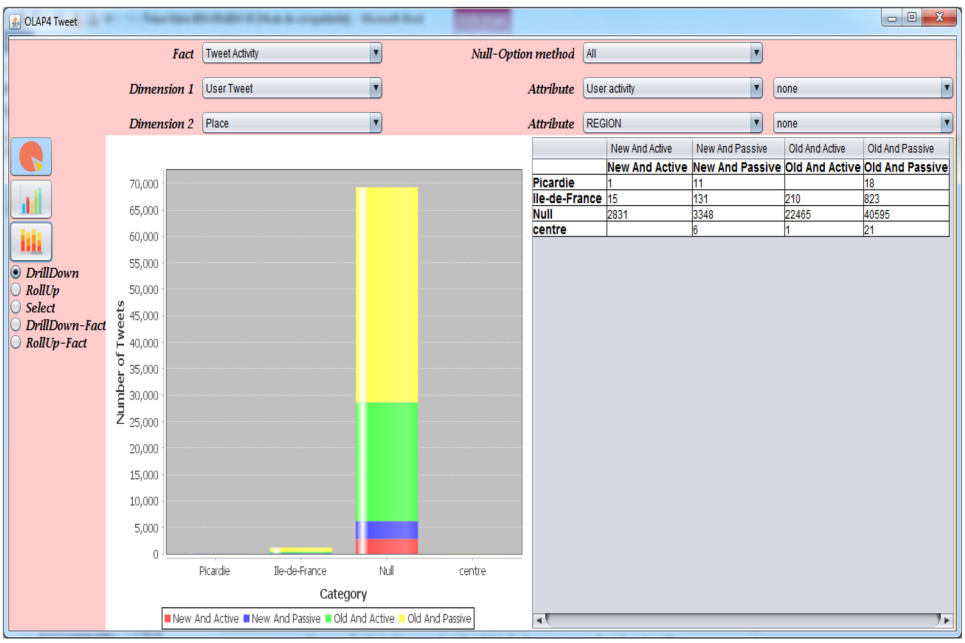


Fig. 20 Result obtained using the *All* option

If $All_{NullLast}$ option is used, the rows containing null values are pushed to the end of the result table (cf. Fig. 21).

Finally, using the *Flexible* option, a message containing a list of all parameters of lower level than *Region* with the percentage of the missing values for each one will be posted to the analyst (cf. Fig. 22). The analyst selects the parameter having a percentage of missing values less than the threshold. In our running example, we arbitrarily set the threshold to 20%. By clicking *Yes*, the selected attribute *Region* is replaced with the *City* attribute (cf. Fig. 23).

Second analysis: *Number of tweets by User-Category and City.*

This time the user intends to see the *Number of tweets* by *City* and by *User-Category*. (S)he clicks on measures and attributes related to her/his analysis. *OLAP4Tweets* displays an interface representing the analysis result in tabular and graphical formats (cf. Fig. 24).

Afterwards, the user wants to roll up analysis level from *City* to *Country*. According to this analysis, no missing values are involved by the *Country* attribute. Hence, no null-option method needs to be specified. By choosing the Bar chart, (s)he obtains the interface in Fig. 25.

Third analysis: *Number of Tweets by Tweet-Sentiment and by Country.*

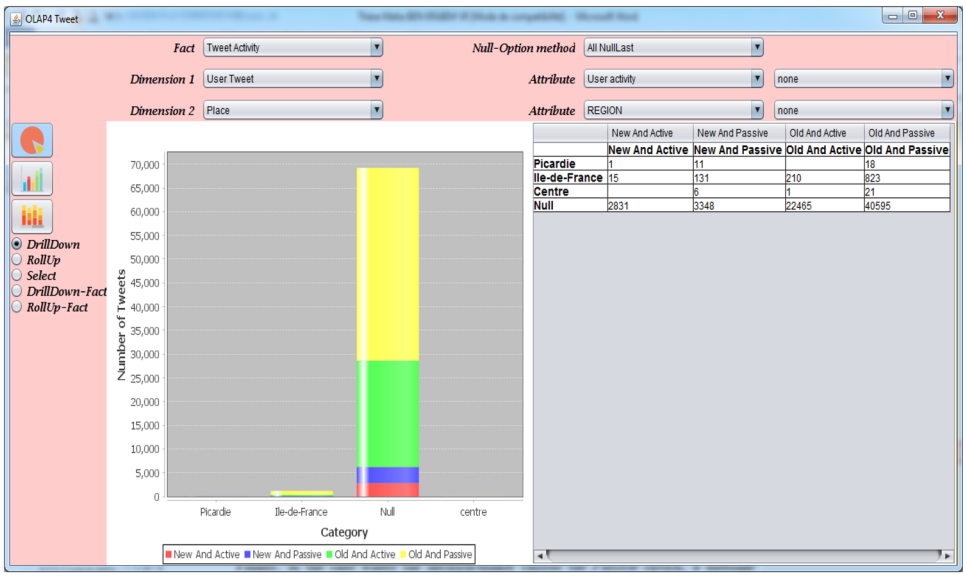


Fig. 21 Analysis result obtained using the $All_{NullLast}$ option

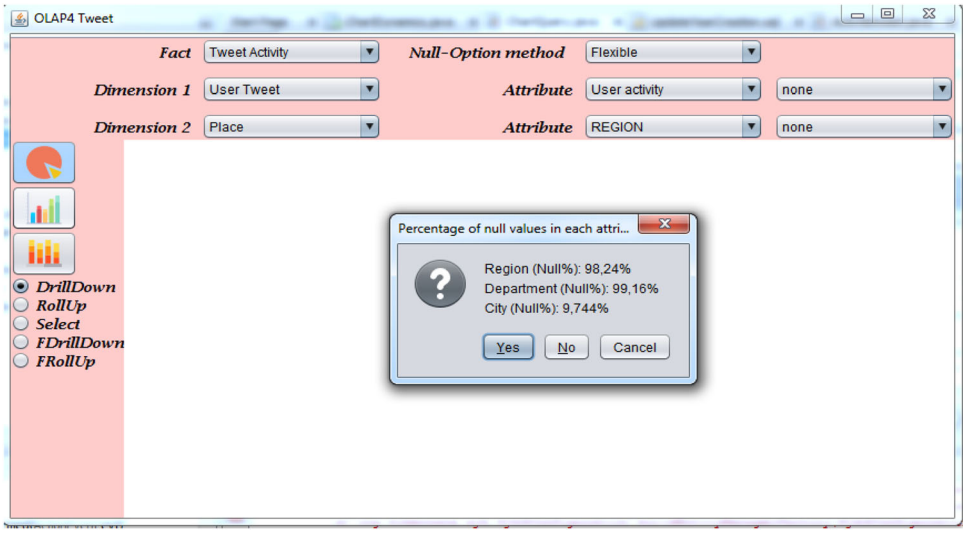


Fig. 22 Percentage of null values

This scenario illustrates how the *FDrilldown* and *FRollup* operators perform on the fact. Assume we want to analyze the *Number of tweets* by *Tweet-Sentiment* (*of the TWEET-METADATA* dimension) and by *Country* (of the *PLACE* dimension). The decision-maker chooses the Bar chart to display the result (cf. Fig. 26).

Based on the analysis result depicted in Fig. 26 the user intends to restrict the analysis to tweets of *level 6*. This level can represent a valuable source of information that could help him get a full picture of topics. Hence, statistical data pertaining to fact can then be pre-summarized, and then be available for large analyses. A slider (on the right top of the interface in Fig. 26) allows changing the fact level. Figure 27 shows the obtained result.

7 Conclusion

This paper has twofold objective; first increase the efficiency of analysis and, secondly, facilitate the analysts' task. To reach this objective, we have proposed a solution based on a set of five OLAP operators for supporting analyses considering the specificities of our proposed multidimensional model 'Tweet Constellation'.

First, facing large volumes of data among which a great amount of missing data is involved, we have proposed an extended version for each of the three conventional OLAP operators Drilldown, Rollup and Select. We have called the extended versions *Null-Drilldown*, *Null-Rollup* and *Null-Select*; they are to process OLAP queries on datasets having missing data. If missing data are found in analyses

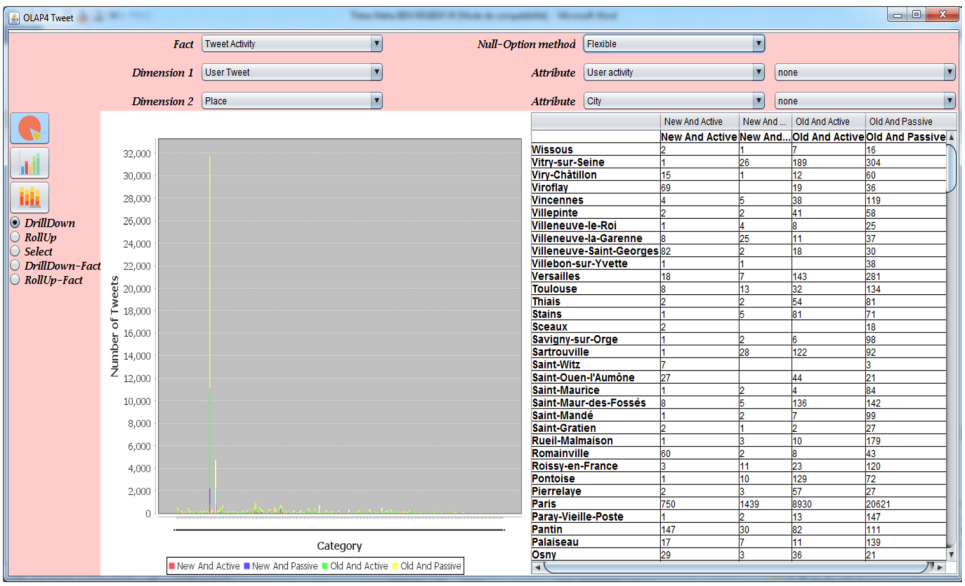


Fig. 23 Interface obtained using the *Flexible* option

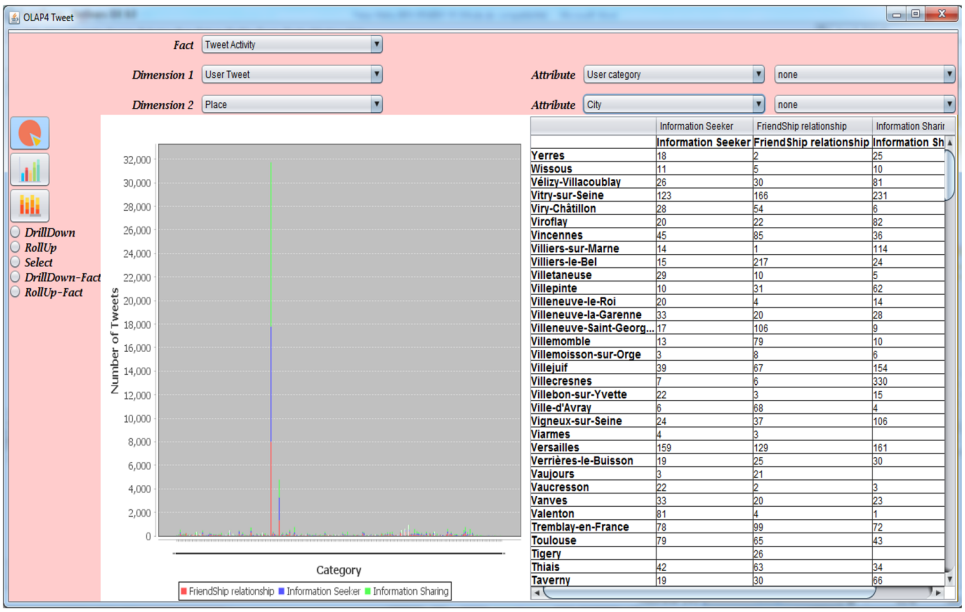


Fig. 24 Second analysis result: *Number of tweets* by *User-Category* and *City*

results, the decision-maker can choose between three options, either *All*, *AllNullLast*, or *Flexible*. Our extended OLAP operators improved the analysis results either by reorganizing the multidimensional table moving non-significant rows at the bottom of the table (*AllNullLast*), or by displaying percentages of not-null data (*Flexible* option).

Secondly, in order to exploit the *reflexive relationship* on fact instances, we have proposed two specific OLAP operators namely *FDrilldown* and *FRollup*. They provide solutions for handling an intuitive navigation between different levels within the fact. The proposed operators are well suited to decision making applications since they can produce an output that leads to many different kinds of analyses. They highlight the importance of tweets responses to show how information propagates through each tweet. They allow identifying topics that have elicited a significant number of responses; these topics can be more investigated/explored using sophisticated tools based on "Text Mining" techniques; thus, we can extract knowledge from tweets and strengthen more semantics.

For each of these operators, we have presented an algebraic formalization, and a logic definition as a pseudo code algorithm.

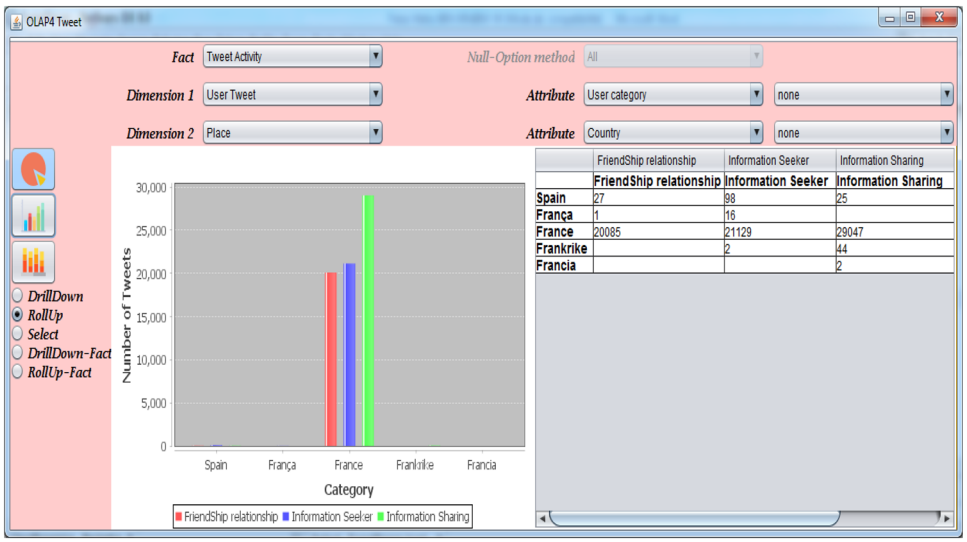


Fig. 25 Interface obtained using the Null-*Rollup* operator

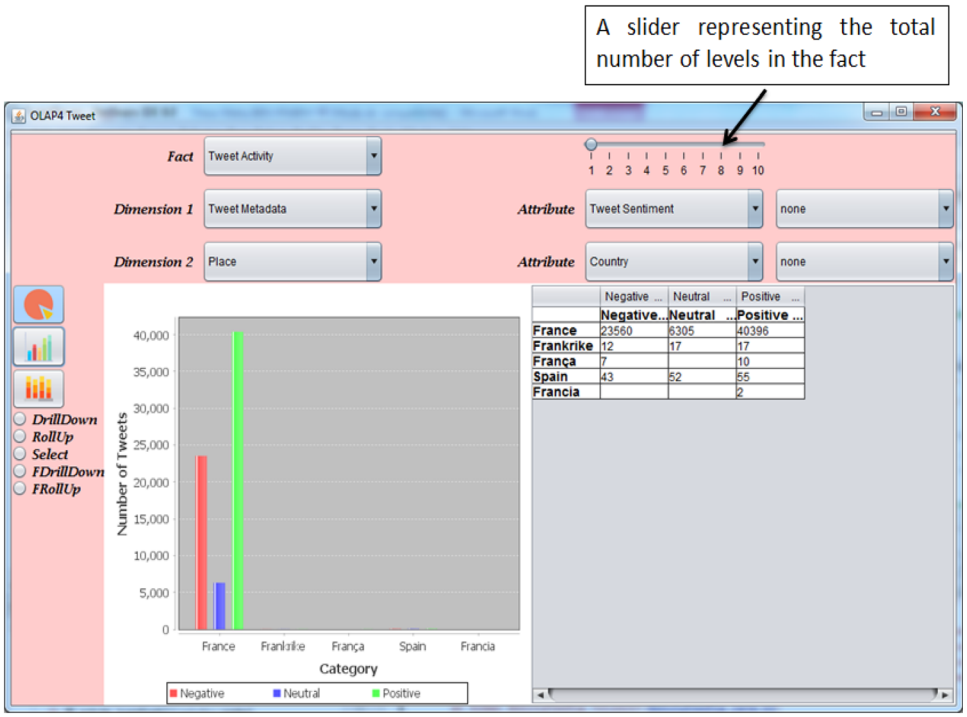


Fig. 26 Third analysis result: *Number of tweets by Tweet-Sentiment and by Country*

To the best of our knowledge, this is the first contribution that has proposed an extension of classical operators for handling missing data during OLAP analysis and for drilling down and up within a fact by exploiting the reflexive relationship.

To show the feasibility of our extended OLAP operators, we have carried out some sample analyses using our prototype *OLAP4Tweet*. The results show that the framework is enough efficient to detect and propose Null-option methods when missing data are involved during the decision making process, and is enough reliable to handle analyses using the proposed operators.

As perspective work, we intend to integrate more analysis operators that take into consideration the specificities of our multidimensional model, as dynamic Data. These operators will help the interpretation of the results of multidimensional analyses on tweets and their metadata. It is also important to use OLAP mining, which integrates online analytical processing (OLAP) with data mining so that mining can be performed in different portions of data

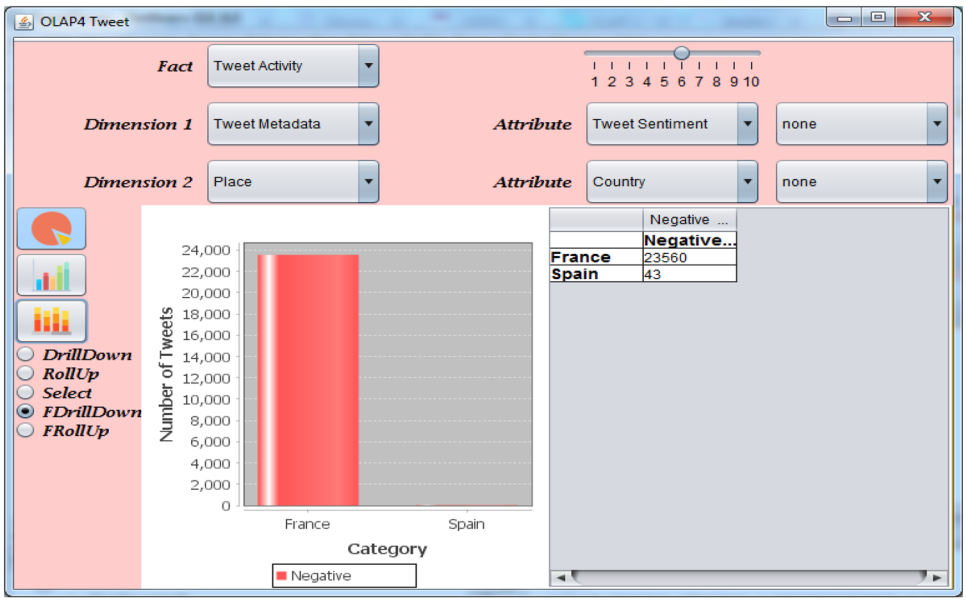


Fig. 27 Result of the *FDrilldown* on TWEET-ACTIVITY

warehouses and at different levels of abstraction at user's fingertips. Moreover, we plan to conduct an evaluation of the prototype on a real case in order to evaluate the response time of queries and the degree of satisfaction of decision-makers with our contributions. Actually, we are in the step of looking for a real case study.

References

1. Ben Kraiem, M., Feki, J., Khrouf, K., Ravat, F., Teste, O.: Modeling and OLAPing social media: the case of twitter. Soc. Netw. Anal. Min. **5**(1), 1–15 (2015)
2. Ravat, F., Teste, O., Tournier, R., Zurfluh, G.: Algebraic and graphic languages for OLAP manipulations. Int. J. Data Warehous. Min. **4**(1), 17–46 (2008)
3. J. Hess, Dealing with missing values in the data warehouse. A Report of Stonebridge Technologies, Inc. (1998)
4. Sadikov, E., Medina, M Leskovec, J,. Garcia-Molina, H.: Correcting for missing data in information cascades. Proceedings of 4th ACM International Conference on Web Search and Data Mining, WSDM'11, Hong Kong, pp. 55–64 (2011)
5. Pedersen, T.B., Jensena, C.S., Dyreson, C.E.: A foundation for capturing and querying complex multidimensional data. Inf. Syst. **26**, 383–423 (2001)
6. Buuren, S.v., Mulligen, E.V.v., Brand, J.P.L., Routine multiple imputation in statistical databases, Proceedings of the Seventh International Conference on Scientific and Statistical Database Management, pp. 74–78 (1994)
7. Rubin, D.B.: Multiple Imputation for Nonresponse in Surveys. Wiley, New York (1987)
8. Graham, JW., Schafer, JL.: On the performance of multiple imputation for multivariate data with small sample size. Statistical Strategies for Small Sample Research, ed. R. Hoyle, pp. 1–29 (1999)
9. Huisman, M.: Imputation of missing network data: some simple procedures. J. Soc. Struct. **10**(1), 1129 (2009)
10. Adar, E., Ré, C.: Managing uncertainty in social networks. IEEE Comput. Soc. Techn. Comm. Data Eng. **30**, 15–22 (2007)
11. Shapcott, M., McClean, S., Scotney, B.: Aggregation of imprecise and uncertain information in databases. IEEE Trans. Knowl. Data Eng. **13**(6), 902–912 (2001)
12. Collins Leke, C., Twala, B., Marwala, T.: Missing data prediction and classification: the use of auto-associative neural networks and optimization algorithms, computer science. Neural and Evolutionary Computing (2014)
13. Chen, A.L.P., Chiu, J.-S., Tseng, F.S.C.: Evaluating aggregate operations over imprecise data. IEEE Trans. Knowl. Data Eng. **8**(2), 273–284 (1996)
14. Abelló, A., Samos, J., Saltora,F.: On relationships offering new drill-across possibilities. Proceedings of the 5th ACM international Workshop on Data Warehousing and OLAP, McLean, Virginia, ACM Press, pp. 7–13 (2002)
15. Abelló, A., Samos, J., Saltora, F.: Implementing Operations to Navigate Semantic Star Schemas. Proceedings of the 6th ACM International Workshop on Data Warehousing and OLAP, New Orleans, Louisiana, ACM Press, pp. 56–62 (2002)
16. Kimball, R.: The data warehouse toolkit: practical techniques for building dimensional data warehouses. Wiley, New York (1996)
17. Cabibbo, L., Torlone.R.: Dimension compatibility for data mart integration. In Proceedings of the Italian Symposium on Advanced Database Systems, Cagliari, pp. 6–17 (2004)
18. Ben Kraiem, M., Feki,J., Khrouf, K., Ravat, F., Teste, O.: OLAP of the tweets from modeling toward exploitation, 8th International Conference on Research Challenges in Information Science (IEEE RCIS'2014), pp. 45–55 (2014)
19. Kevin, M.: Twitter API: Up and running: Learn how to build applications with the twitter API, O'Reilly Media, Inc., pp. 30–31 (2009)

Maha Ben Kraiem I received my Bachelor's degree in computer science applied to management. (2009) from the Faculty of Economics and Management, University of Sfax (Tunisia), a Master degree (2012) and a Ph.D. from the university of Sfax. I am a research member at the IRIT research center at the University of Toulouse, France and a member at the Mir@cl research Laboratory, University of Sfax (Tunisia). My research interests span many aspects of data management and the business intelligence. My publications appear in major national and international journals and conference proceeding.

Dr. Mohammed Alqarni obtained a bachelor's degree in computer science from King Khalid University, Saudi Arabia in 2008, and received an MSc in Computational Sciences from Laurentian University, Sudbury, Canada in 2012. He was then awarded a Ph.D. in Computer Science from McMaster University, Hamilton, Canada in 2016. He is currently an assistant professor at, and the dean of, the College of Computer Science and Engineering at the University of Jeddah, Saudi Arabia. He enjoys research in a wide variety of topics.

Jamel Feki received his BS in CS (1980) from the University of Sfax (Tunisia), a Master degree (1981) and a Ph.D. (1984) in CS from the University Paul SABATIER (France). He joined the University of Sfax (Tunisia) in 1986 where he is a full professor and member of the Mir@cl research laboratory. Now, he is a full professor at the University of Jeddah (Saudi Arabia) since 2015. He has supervised several Ph.D. theses and has published research papers in refereed journals and conferences; he is a co-author of three book chapters. His research interests include Decision Support Systems and Business intelligence: analytical requirements specification, Data Warehouse design methods, DW Integration, Knowledge Warehouses and Big Data. He is steering committee member in conferences and workshops; he is PC member in international conferences and reviewer in journals.

Franck Ravat is currently a full professor of computer science at the University of Toulouse, France. He also is a research staff member at the IRIT research centre (CNRS UMR 5505). His research interests span many aspects of data management and the business intelligence. In the field of data management, he focusses on data integration of various data types (structured, semi-structured and unstructured data) and modelling data to be implemented and analyzed in different frameworks (distributed data files, relational DBMS, NOSQL or NewSQL systems…) In the field of Business Intelligence, he focusses on data warehousing based on OLAP (On-Line Analytical Processing) analyses and data lakes. His publications appear in major national and international journals and conference proceedings, for which he also serves as reviewer.