Facilitate effective decision-making by warehousing reduced data

Faten Atigui, Franck Ravat, Jiefu Song, Olivier Teste, Gilles Zurfluh

Facilitate effective decision-making by warehousing reduced data: is it feasible?

Faten Atigui[1], Franck Ravat[2], Jiefu Song[2], Olivier Teste[3], Gilles Zurfluh[2]

[1] CEDRIC- Conservatoire National des Arts et Métiers, 292 Rue Saint Martin
75141 Paris Cedex 03, France
[2] IRIT - Université Toulouse I Capitole, 2 Rue du Doyen Gabriel Marty
F-31042 Toulouse Cedex 09, France
[3] IRIT –Université. Toulouse II Jean Jaurès, 1 Place Georges Brassens
F-31703 Blagnac Cedex, France
[1] faten.atigui@cnam.fr
[2] [3] {ravat|song|teste|zurfluh}@irit.fr

ABSTRACT

Our aim is to provide a solution for multidimensional data warehouse's reduction based on analysts' needs which will specify aggregated schema applicable over a period of time as well as retain only useful data for decision support. Firstly, we describe a conceptual modeling for multidimensional data warehouse. A multidimensional data warehouse's schema is composed of a set of states. Each state is defined as a star schema composed of one fact and its related dimensions. The derivation between states is carried out through combination of reduction operators. Secondly, we present a meta-model which allows managing different states of multidimensional data warehouse. The definition of reduced and unreduced multidimensional data warehouse schema can be carried out by instantiating the meta-model. Finally, we describe our experimental assessments and discuss their results. Evaluating our solution implies executing different queries in various contexts: unreduced single fact table, unreduced relational star schema, reduced star schema and reduced snowflake schema. We show that queries are more efficiently calculated within a reduced star schema.

***Keywords**:* multidimensional design, data reduction, experimental assessment

INTRODUCTION

Nowadays, decision support systems are mostly based on Multidimensional Data Warehouse (MDW). A MDW schema is based on facts (analysis subjects) and dimensions (analysis axis). A fact includes analysis indicators while dimensions contain analysis parameters. The analysis parameters are organized according to their hierarchical level in order to classify the parameters from the lowest granularity to the highest granularity.

In a MDW, data is stored permanently and new data is steadily added. Hence, a MDW stores a huge volume of data in which the analyst may well get lost during her/his analyses. On the other hand, the relevance of MDW data decreases with time: detailed information is generally considered relevant for recent data (Skyt, Jensen, & Pedersen, 2008), while more aggregated information can usually satisfy the needs of analysis with older data. For instance, a decision-maker may have interest in analyzing published news by subthemes for the last four years, but

this level of granularity may be proved inappropriate if the analysis was carried out over an older period because most of today's subthemes did not exist before: it is impossible to carry out analyses over published news by subthemes but by a higher granularity level which stays stable over time, such as news' theme.

Facing large volumes of both relevant and irrelevant data, our aim is to increase the performance of query treatment and especially to facilitate the analysts' task by providing only pertinent data over time. Our aim is to provide a multidimensional analysis framework adapted to analysts' needs, allowing them to remove the temporal granularity levels which are irrelevant for their analyses. As detailed data value decreases with time, we implement selective deletions at low levels of granularity. This reduction is achieved mainly through progressive data aggregation: older data is synthesized.

This paper is composed of the following sections: Section 2 describes a state of the art of data reduction. Section 3 defines our conceptual model of multidimensional data based on reductions. Section 4 presents a meta-model for managing MDW composed of states. Section 5 provides experimental assessments to evaluate our solution in various implementation environments.

RELATED WORK

Reducing data allows us both to decrease the quantity of irrelevant data in decision making and to increase future analysis quality (Udo & Afolabi, 2011). In the context of decision support, data reduction is a technique originally used in the field of data mining (Okun & Priisalu, 2007; Udo & Afolabi, 2011).

In the DW context, (Garcia-Molina, Labio, & Yang, 1998) were the first to define solutions for data deletion. More precisely, they study data expiration in materialized views so that they are not affected and can be maintained after updates with the help of a set of standard predefined views.

In the multidimensional area, (Chen et al., 2002) propose an architecture allowing the integration of data streams into a MDW and reduce the size. The size reducing is predefined and automatically executed by partially aggregating the data cube; it makes sure the detailed information is only available during a time interval. Nevertheless, this work only focuses on the fact table. (Skyt et al., 2008) presents a technique for progressive data aggregation of a fact. This study intends to specify data aggregation criteria of a fact due to higher levels of dimensions. The authors also provide techniques to query reduced multidimensional objects. As mentioned in (Iftikhar & Pedersen, 2011), this work is highly theoretical but it fails to provide us a concrete example of implementation strategy. In (Iftikhar & Pedersen, 2011), a gradual data aggregation solution based on conception, implementation and evaluation is proposed. This solution is based on a table containing different temporal granularities: second, minute, hour, month and year.

This previous work only focuses on the fact table. (Iftikhar & Pedersen, 2010, 2011) use a temporal table for gradual data reduction. None of the previous work takes into account analysts' needs. Our goal is more ambitious as it aims to study data reduction of the complete multidimensional schema that depends only on the users' needs. We intend to provide a consistent analysis environment and thus facilitate the analyst's task by limiting the analysis to semantically consistent data.

CONCEPTUAL MODELING

We aim at specifying aggregated schemata over time in order to keep only useful data for decision support. All dimensions and facts may be reduced to different granularity levels according to analyst's needs. The useless information is deleted from MDW in order to provide only relevant data for analyses.

Case Study

The case study shows a multidimensional schema progression that fulfills the analyst's needs. A MDW fed by RSS streams allows decision-makers to analyze the number of published news from her/his favorite websites. A decision-maker expresses her/his needs as followed: (a) during the last four years, news analysis is carried out with reference to lowest levels of granularity: subtheme, city and publication date; (b) in the previous period from 2010 to 2000, analyses are summarized according to news' theme, country mentioned in the news and month of publication, because no daily analysis referring to subtheme and city is required; (c) before 2000, only aggregated information about published news by quarter and by continent makes sense.

The following three figures represent the conceptual multidimensional schemas fulfilling user's needs. Each schema is based on the graphic notation called star schema introduced in (M. Golfarelli, Maio, & Rizzi, 1998). A star schema is based on a subject of analysis (fact) related to different dimensions. Each fact contains of one or more indicators. For instance, in figure 1 the fact named "*FNews*" is composed of two indicators: number of published news (*NBN*) and a set of key words appeared in news (*KeyWords*). A dimension models an analysis axis; it reflects information according to which subjects of analysis are to be dealt with. For instance, the "*FNews*" fact is connected to 3 dimensions: *DTheme*, *DGeography* and *DTimes*. Dimension attributes (also called parameters or levels) are organized according to one or more hierarchies. Hierarchies represent a particular vision (perspective) of a dimension.

Figure 1. MDW schema valid from 2010 to 2014

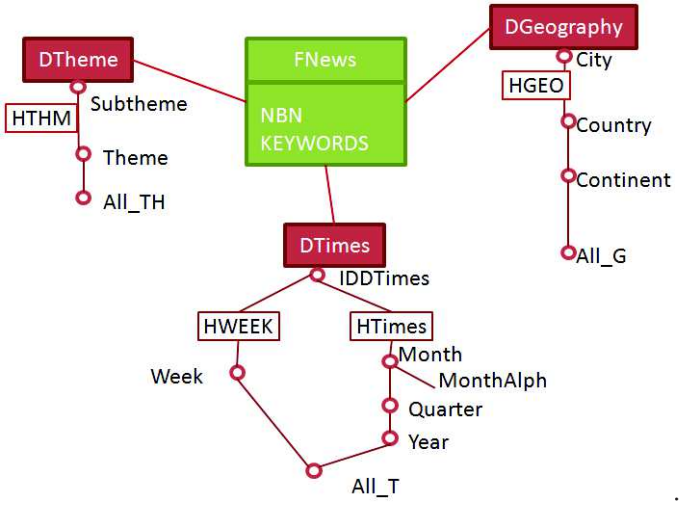


Figure 2. MDW schema valid from 2000 to 2010

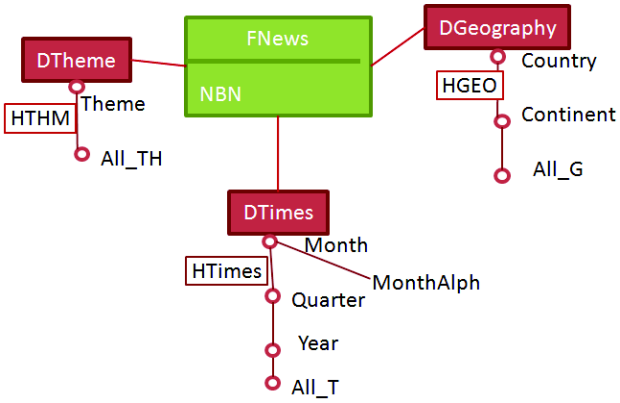
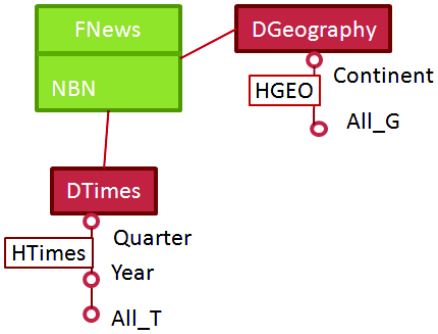


Figure 3. MDW schema valid from 1990 to 2000



Concepts

We define a reduced MDW as a set of star schemas called states. Each state is valid for a certain period and is modeled with a star schema composed of one fact and several dimensions. The current state is the valid state of the MDW at present. Past states correspond to a succession of reduced states over time. Each past state is defined from a previous one with a reduction function.

Let us define $\mathcal{N}$ such as $\mathcal{N} = \{n^i,..., n^k\}$ is a finite set of names, $\mathcal{N} \neq \varnothing$.

Definition 1. A MDW is defined by $S = (n^S ; \mathcal{E} ; \text{Map})$ where

- $n^S \in \mathcal{N}$ is the name of the MDW
- $\mathcal{E} = \{E_1 ; \ldots ; E_n\}$ is a set of states composing the MDW
- Map: $\mathcal{E} \rightarrow \mathcal{E} \mid \text{Map}(E_k) = E_{k+1}$ is a reduction function defining the state named E_{k+1} obtained by the reduction of E_k.

Let us define F and D such as $F = \{F_1,..., F_n\}$ is a finite set of facts, $n \geq 1$ and $D = \{D_1,..., D_m\}$ is a finite set of dimensions, $m \geq 2$.

Definition 2. A state is a star schema defined for a temporal period such as $E_i = (F_i ; \mathcal{D}_i ; T_i)$ where

- $F_i \in F$ is a fact representing a subject of analysis
- $\mathcal{D}_i = \{D_{TIMES} ; D_1 ; \ldots ; D_m\} \subseteq D$ is a set of dimensions associated to the fact with necessarily a temporal dimension denoted D_{TIMES}
- $T_i = [t^i_{start} ; t^i_{end} [$ is a temporal interval defined on the D_{TIMES} dimension and associated to the state E_i.

To define T_i, we adopt a linear and discrete time model approaching time in granular way through time observation units (Wang, Bettini, Brodsky, & Jajodia, 1997; Ravat & Teste, 2000). A temporal grain is an integer relative to a time unit; we adopt the standard time units

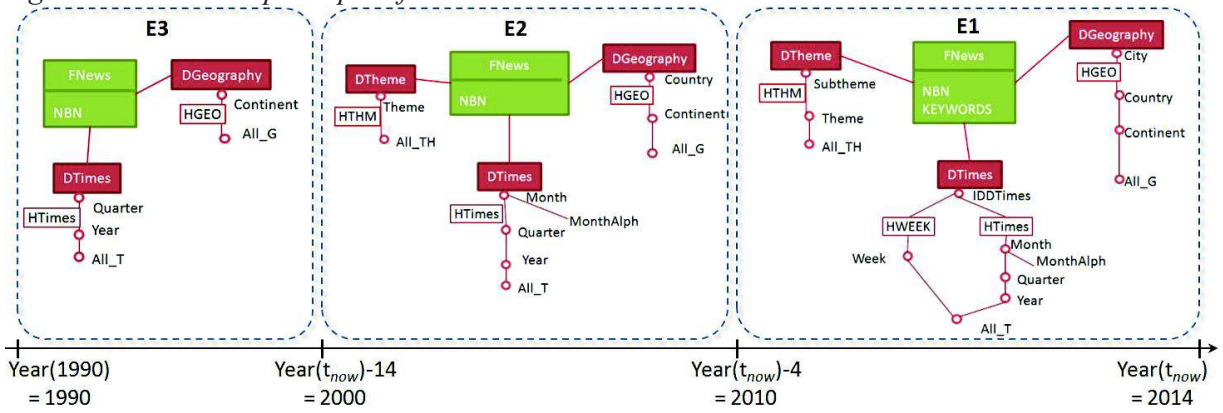
manipulated through functions: Year, Quarter, Month, Day... For instance, Year (1990) defines the instant "1990" for the year time unit.

An instant is a temporal grain. We note T_{now} the current instant which is characterized by its dynamic nature, ie. T_{now} changes constantly as time goes by. A time interval is defined by a couple of instants noted "t_{start}" and "t_{end}". These instants can be fixed (temporal grains) or dynamic (defined with the instant "T_{now}").

Example. The following figure represents three states of our case study. It illustrates the principle of states derived by the reduction. This MDW is defined as follows: $\mathcal{E} = \{E_1 ; E_2 ; E_3\}$ with Map = { (E_1, E_2) ; (E_2, E_3) } where

- $E_1 = (F_{NEWS} ; \{D_{THEME} ; D_{TIMES} ; D_{GEOGRAPHY}\} ; [Year(T_{now})-4 ; Year(T_{now})[)$
- $E_2 = (F_{NEWS} ; \{D_{THEME} ; D_{TIMES} ; D_{GEOGRAPHY}\} ; [Year(T_{now})-14 ; Year(T_{now})-4[)$
- $E_3 = (F_{NEWS} ; \{D_{TIMES} ; D_{THEME}\} ; [Year(1990); Year(T_{now})-14[).$

Figure 4. Reduction principle of multidimensional schemas



The current state denoted E_1 is associated to the validity interval [$Year(t_{now})$-4; $Year(T_{now})$[corresponding to [2010; 2014[. The instances of this state correspond to news published between 2010 and 2014 only, according to the D_{TIMES} dimension. In the same way, the state named E_2 stores data related to news published between 2000 and 2010, whereas the state denoted E_3 stores data related to news published prior to 2000.

In figure 4, Year(1990) is a fixed instant representing the date when the database was created. In this figure, we can also find time-variant intervals (moving over time) defined by the following instants: $Year(T_{now})$-14, $Year(T_{now})$-4 and $Year(T_{now})$. So, next year, $Year(T_{now})$ = 2015, $Year(T_{now})$-4 = 2011 and $Year(T_{now})$-14 = 2001. At each change of year, the states denoted E_1, E_2 and E_3 will be instantly updated.

Definition 3. A *fact*, denoted F_i, $\forall i \in [1..n]$, is defined by (n^{Fi}, M^{Fi}) where

- $n^{Fi} \in \mathcal{N}$ is the fact name
- $M^{Fi} = \{m_1,..., m_{pi}\}$ is a set of *measures* or *indicators*.

Definition 4. A *dimension*, denoted D_i, $\forall i \in [1..m]$, is defined by (n^{Di}, A^{Di}, H^{Di}), where

- $n^{Di} \in \mathcal{N}$ is the dimension name

- $A^{Di} = \{ a_1^{D_i}, ..., a_{r_i}^{D_i} \}$ is the set of the *attributes of the dimension*
- $H^{Di} = \{ H_1^{D_i}, ..., H_{h_i}^{D_i} \}$ is a set of *hierarchies*.

Hierarchies organize the attributes of a dimension, from the finest graduation (root parameter, ID_{Di}) to the most general graduation (extremity parameter, All_{Di}). Thus, a hierarchy defines the valid navigation paths on an analysis axis.

Definition 5. A *hierarchy*, denoted H_j (abusive notation of $\boldsymbol{H_j^{D_i}}$, $\forall i \in [1..m]$, $\forall j \in [1..h_i]$) is defined by (n^{Hj}, P^{Hj}, $\prec^{Hj}$, $Weak^{Hj}$), where

- $n^{Hj} \in \mathcal{N}$ is the hierarchy name;
- $P^{Hj} = \{ p_1^{H_j}, ..., p_{q_j}^{H_j} \}$ is a set of attributes called *parameters*, $P^{Hj} \subseteq A^{Di}$;
- $\prec^{Hj} = \{(p^{Hj}_x, p^{Hj}_y) \mid p^{Hj}_x \in P^{Hj} \wedge p^{Hj}_y \in P^{Hj} \}$ is an antisymmetric and transitive binary relation between parameters. Remember that the antisymmetry means that $(p^{Hj}_{k1} \prec^{Hj} p^{Hj}_{k2}) \wedge (p^{Hj}_{k2} \prec^{Hj} p^{Hj}_{k1}) \Rightarrow p^{Hj}_{k1} = p^{Hj}_{k2}$ while the transitivity means that $(p^{Hj}_{k1} \prec^{Hj} p^{Hj}_{k2}) \wedge (p^{Hj}_{k2} \prec^{Hj} p^{Hj}_{k3}) \Rightarrow p^{Hj}_{k1} \prec^{Hj} p^{Hj}_{k3}$.
- $Weak^{Hj} : P^{Hj} \rightarrow 2^{A^{D_i} \setminus P^{H_j}}$ is an application that associates to each parameter a set of dimension attributes, called *weak attributes* (2^N represents the power set of N).

Example. The E_3 state of the figure 3 is composed of one fact and two dimensions and it is valid from 1990 to 2000. The fact table named *FNews* contains a measure notated *NBN*. The dimension *DGeography* contains the hierarchy *HGeo* on which the parameters are organized according to their granularity level: from the lowest level *Continent* to the highest level *ALL_G*. The other dimension is named D_{TIMES}, it is graduated by the attributes *Quarter*, *Year* and *ALL_T* on the hierarchy *HTimes*.

The formal representation of state E_3 is as follows:

$E_3 = (F_{NEWS} ; \{ D_{GEOGRAPHY} ; D_{TIMES} \} ; [t_{1990} ; t_{2000}[)$ where

- F_{NEWS} = (FNews; { NBN })
- $D_{GEOGRAPHY}$ = (DGEOGRAPHY; {Continent, ALL_G }; {HGeo})
- D_{TIMES} = (DTIMES; {Quarter, Year, ALL_T}; {HTime}).

We take the hierarchy *HTimes* as an example to illustrate the abstract representation for a hierarchy.

HTimes = (n^{HTimes}, P^{HTimes}, $\prec^{HTimes}$, $Weak^{HTimes}$) where

- n^{HTimes} = HTimes
- P^{HTimes} = {Quarter, Year, ALL_T}
- $\prec^{HTimes}$ = {(Quarter, Year); (Year, ALL_T)}
- WeakHTimes = ∅.

Reduction Operators

Deriving the reduced schema denoted E_{k+1} from a schema denoted E_k is performed through the composition of reduction operators. We define the set of these operators as $\mho = \{RollUp^{reduce}; Drop^{reduce}; Add^{reduce}; Slice^{reduce}\}$ as the minimum core of elementary operators to define a new state. Two categories of operators are available:

- Schema reduction operators (*cf.* Table 1) allow to delete attributes (parameters, weak attributes and measures), and
- Instance reduction operators (*cf.* Table 2) reducing value domains of the dimension without modifying its schema.

The $RollUp^{reduce}$ operator provides a new state in which the specified dimension is reduced by removing all the attributes under the parameter that is specified in the operator. If the specified parameter is an extremity parameter like $ALL_{D_{rollup}}$, the dimension is completely removed in the reduced state; if the specified parameter is at the lowest granularity level like $P_1^{H_j}$, no parameter will be removed from the dimension.

The $Drop^{reduce}$ operator provides a new state in which the fact is reduced by the deletion of a specified measure. On the contrary , the Add^{reduce} operator permits to add a new measure in the fact of a new state.

Table 1. Reduction operators on schemata.

Operators	
$RollUp^{reduce}(E_k ; D_{rollup} ; p_{rollup} ; T_{k+1}) = E_{k+1}$	
Inputs	$E_k = (F_k ; \mathcal{D}_k ; T_k)$: initial state; $D_{rollup} \in \mathcal{D}_k$: dimension dedicated to a reduction; $p_{rollup} \in A^{Drollup}$: reduction parameter of the D_{rollup} dimension; $T_{k+1} = [t_{start}^{k+1} ; t_{end}^{k+1} [$ is the temporal interval of validation for state E_{k+1}.
Output	$E_{k+1} = (F_{k+1} ; \mathcal{D}_{k+1} ; T_{k+1})$ reduced state such as $F_{k+1} = F_k$; $\mathcal{D}_{k+1} = \mathcal{D}_k \setminus \{ D_{rollup} \} \cup \{ D_{new} \}$ (*) with $D_{new} = (n^{Dnew} ; A^{Dnew} ; H^{Dnew})$ $n^{Dnew} = n^{Dold}$ $A^{Dnew} = \{ a_x \in A^{Drollup} \mid a_x = p_{rollup} \vee \forall H_j \in H^{Drollup}, p_{rollup} \prec^{Hj} a_x \}$ $H^{Dnew} = \{ H_x \in H^{Drollup} \mid n^{Hx} = n^{Hj} \wedge P^{Hx} = \{ p_y \in P^{Hj} \mid p_y = p_{rollup} \vee p_{rollup} \prec^{Hj} p_y \} \wedge \prec^{Hx} = \{ (p^{Hj}_{x1}, p^{Hj}_{x2}) \in \prec^{Hj} \mid p^{Hj}_{x1} = p_{rollup} \vee p_{rollup} \prec^{Hj} p^{Hj}_{x1} \} \wedge Weak^{Hx} := \{ (p_{x1}, A^{Hx}_{x1}) \in Weak^{Hj} \mid p_y \in P^{Hj} \}$. $T_{k+1} = [t_{start}^{k+1} ; t_{end}^{k+1} [\wedge t_{start}^{k+1} = t_{end}^{k} - 1$
$Drop^{reduce}(E_k ; m_{drop} ; T_{k+1}) = E_{k+1}$	

Inputs	$E_k = (F_k ; \mathcal{D}_k ; T_k)$: initial state ; $m_{drop} \in M_k$ is a measure of F_k. $T_{k+1} = [t_{start}^{k+1} ; t_{end}^{k+1} [$ is the temporal interval of validation for state E_{k+1}.
Output	$E_{k+1} = (F_{k+1} ; \mathcal{D}_{k+1} ; T_{k+1})$ reduced state such as $F_{k+1} = (n^{Fk}, M^{Fk} \setminus \{ m_{drop} \})$; $\mathcal{D}_{k+1} = \mathcal{D}_k$. $T_{k+1} = [t_{start}^{k+1} ; t_{end}^{k+1} [\wedge t_{start}^{k+1} = t_{end}^{k}$-1
$Add^{reduce}(E_k ; m_{add} ; T_{k+1}) = E_{k+1}$	
Inputs	$E_k = (F_k ; \mathcal{D}_k ; T_k)$: initial state ; $m_{add} \in M_k$ is a measure not presented in F_k. $T_{k+1} = [t_{start}^{k+1} ; t_{end}^{k+1} [$ is the temporal interval of validation for state E_{k+1}.
Output	$E_{k+1} = (F_{k+1} ; \mathcal{D}_{k+1} ; T_{k+1})$ reduced state such as $F_{k+1} = (n^{Fk}, M^{Fk} \wedge \{ m_{add} \})$; $\mathcal{D}_{k+1} = \mathcal{D}_k$. $T_{k+1} = [t_{start}^{k+1} ; t_{end}^{k+1} [\wedge t_{start}^{k+1} = t_{end}^{k}$-1

(*) If $A^{Dnew} = \{ALL_{D_{rollup}}\}$ then $\mathcal{D}_{k+1} = \mathcal{D}_k \setminus \{ D_{rollup} \}$

The $Slice^{reduce}$ operator provides a reduced state in which the instances of the specified dimension denoted D_{Slice} is reduced. The dimension instances that satisfy the predicate denoted $pred_{slice}$ are kept in the new state.

Table 2. Reduction operators on instances.

$Slice^{reduce}(E_k ; D_{slice} ; pred_{slice} ; T_{k+1}) = E_{k+1}$	
Inputs	$E_k = (F_k ; \mathcal{D}_k ; T_k)$: initial state ; $D_{slice} \in \mathcal{D}_k$: dimension dedicated to a reduction; $pred_{slice}$: selection predicate on a domain denoted $dom(D_{slice})$ of D_{slice}. $T_{k+1} = [t_{start}^{k+1} ; t_{end}^{k+1} [$ is the temporal interval of validation for state E_{k+1}.
Output	$E_{k+1} = (F_{k+1} ; \mathcal{D}_{k+1} ; T_{k+1})$ reduced state such as $F_{k+1} = F_k$; $\mathcal{D}_{k+1} = \mathcal{D}_k$ with $dom(D_{slice}) = \{ v_i \in dom(D_{slice}) \mid pred_{slice}(v_i) = \text{TRUE} \}$. $T_{k+1} = [t_{start}^{k+1} ; t_{end}^{k+1} [\wedge t_{start}^{k+1} = t_{end}^{k}$-1

Example. In the previous example, we defined two reduced states. Each of them is defined by a reduction function. These functions are defined bellow. The first Map function, composed of

three $RollUp^{reduce}$ operators and one $Drop^{reduce}$ operator, permits to define the E_2 state. The second Map function composed of two $RollUp^{reduce}$ operators permits to define the E_3 state.

From 2000 to 2010, the analyst would like keep only number of published news (*NBN*) as measure and summarize his analyses according to news' theme, country and publication month. So firstly we delete the undesired measure *KEYWORDS*; then we remove all the attributes under the parameter P_{THEME}; next the parameters under $P_{COUNTRY}$ on dimension $D_{GEOGRAPHY}$ are also removed from the schema; at last we reduce the granularity level of the dimension D_{TIMES} until to the level of parameter P_{MONTH}. We obtain a combination of reduction operators permitting to get the E_2 state as followed:

- $RollUp^{reduce}$ ($RollUp^{reduce}$ ($RollUp^{reduce}$ ($Drop^{reduce}$ (E_1; KeyWords; [Year(T_{now})-14 ; Year(t_{Tow})-4[); D_{THEME} ; P_{THEME} ; [Year(T_{now})-14 ; Year(T_{now})-4[) ; $D_{GEOGRAPHY}$; $P_{COUNTRY}$; [Year(T_{now})-14 ; Year(t_{Tow})-4[);D_{TIMES} ; P_{MONTH} ; [Year(T_{now})-14 ; Year(t_{Tow})-4 [)= E_2 ;

For the period before 2000, the analyst want to keep only some aggregated information about number of published news by quarter and by continent. Therefore first of all we remove the whole dimension D_{THEME} by specifying a removal of granularity until to the extremity parameter *ALL_TH*; then the dimension $D_{GEOGRAPHY}$ is summarized until to the level of parameter $P_{CONTINENT}$, while for the other dimension D_{TIMES} we remove all the granularities lower than $P_{QUARTER}$. We can apply the following reduction function in order to obtain the E_3 state:

- $RollUp^{reduce}$ ($RollUp^{reduce}$ ($RollUp^{reduce}$ (E_2; D_{THEME} ; ALL_TH ; [Year(1990) ; Year(T_{now})-14[) ; $D_{GEOGRAPHY}$; $P_{CONTINENT}$; [Year(1990) ; Year(T_{now})-14[); D_{TIMES} ; $P_{QUARTER}$; [Year(1990) ; Year(T_{now})-14[) = E_3.

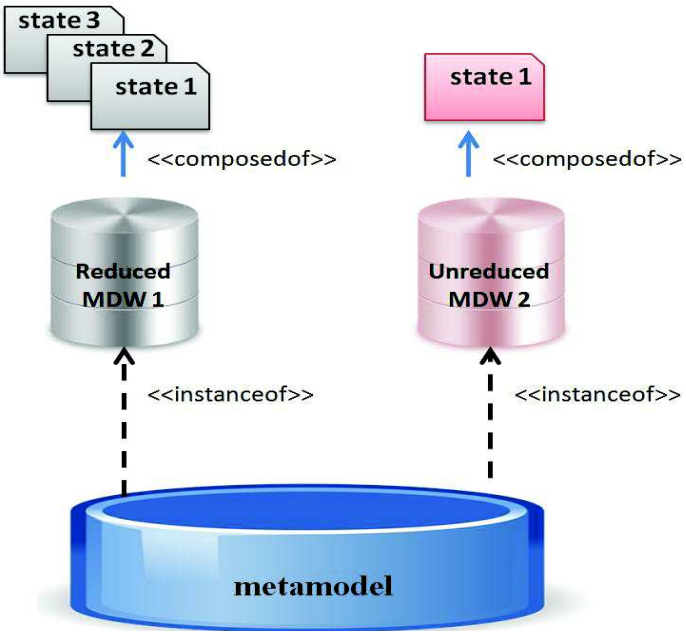
IMPLEMENTATION IN R-OLAP ENVIRONMENT

Based on the conceptual presentation of MDW and its definition, we implement our solution in a relational framework. Firstly we present our implementation architecture in order to visualize the relationships between implementation components. Then we define a conceptual modeling of metamodel permitting to manage MDW composed of states. At last we implement and instantiate our metamodel in DBMS Oracle to obtain a relational modeling.

Architecture of implementation

We have defined a metamodel which aims to concretizing and implementing the concepts previously defined. Our metamodel allows decision-makers to manage and query MDW. As showed in figure 5, both reduced and unreduced MDW can be defined through instantiation of metamodel. Reduced MDW is composed of several states, while unreduced MDW is defined as a particular case of MDW composed of only one state.

Figure 5. Implementation architecture



Conceptual modeling of metamodel

The graphical notation of our conceptual metamodel below is based on static class diagram that contains only static elements (the attributes) without dynamic components (the methods).

Figure 6. UML class diagram of the metamodel

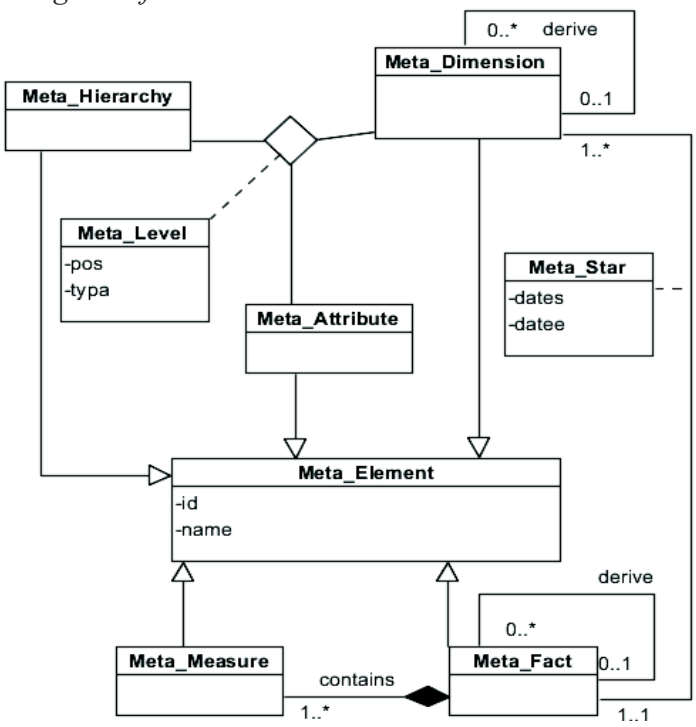


Table 3. Data dictionary of the metamodel

Code	Description	Type	Constraints
datee	ending date of a star schema	DATE	
dates	starting date of a star schema	DATE	
id	identifier of a meta-element	INTEGER	>0
ida	identifier of a attribute	INTEGER	>0
idd	identifier of a dimension	INTEGER	>0
idf	identifier of a fact	INTEGER	>0
idh	identifier of a hierarchy	INTEGER	>0
idm	identifier of a measure	INTEGER	>0
name	name of a meta-element	VARCHAR	
pos	position of a parameter or a weak attribute on a hierarchy of a dimension	INTEGER	>0
typa	type of attribute at a level	VARCHAR	'Parameter', 'Weak Attribute'

As we can see from the figure 6, the metamodel embodies all the concepts discussed above. Firstly, as all of the elements possess a name regardless of its type, the name of element along with its identifier are centralized and managed by the class notated *Meta_Element*. This class is the base of our metamodel, all the rest are considered as specialized classes of *Meta_Element*.

Secondly the notion of state $E_i = (F_i ; \mathcal{D}_i ; T_i)$ is represented by the association class *Meta_star*. This association class possesses a temporal interval between a start date (*DateS*) and an end date (*DateE*).

Thirdly, fact and the dimension are embodied respectively by the classes *Meta_Fact* and *Meta_Dimension*. Each of these classes possesses a recursive association denoted *Derive* pointing to itself. This link permits to connect a derived fact or dimension in a state to the original fact or dimension, in this way we implement the reduction function permitting to define a derivate state obtained after the reduction of original MDW.

Fourthly, by definition a fact contains a set of measures while a measure belongs to one and only one fact. This rule is expressed by the relationship notated *Contain* between the class *Meta_mesure* and the class *Meta_fact*.

Last but not the least, the attributes are organized on hierarchies of dimension according to their granularity level: from the finest graduation (root parameter, ID_{Di}) to the most general graduation on hierarchy (ALL_{Di}). The ternary association in our metamodel permits us to materialize the relationships among dimension, hierarchy, level and attributes. What's more, the antisymmetric and transitive binary relation between attributes on a hierarchy is represented by the attributes *Pos* and *Typa* in the association class *Meta_Level*. The position of an attribute at a level shows its granularity level in comparison to the others, while the type shows if the attribute belong to parameter or weak attribute on hierarchy.

Relational modeling of metamodel

We implement the conceptual metamodel in DBMS Oracle and then instantiate it with the MDW proposed in our case study. To better illustrate modeling principles, the following figures and explanations take data instances related to the state E_1. In order to distinguish state E_1 with the

original MDW before reduction, during model implementation we add "*_E1"* as suffix to the name of dimensions and fact in state E_1, while the name in original MDW before reduction is not suffixed.

First of all, as we mentioned in the previous section, the class *Meta_Element* includes all the elements in a MDW. Since each element possessed a unique identifier and a name, these two common attributes are centralized and managed by *Meta_Element* (*cf.* the following figure).

Figure 7. Content of Meta_element

ID	NAME
a1	IDDTIMES
a2	MONTH
a3	MONTHALPH
a4	QUARTER
a5	YEAR
a6	WEEK
a7	ALL_T
d1	DGEOGRAPHY
d2	DTIMES
d3	DTHEME
d4	DGEOGRAPHY_E1
d5	DTIMES_E1
d6	DTHEME_E1
f1	FNEWS
f2	FNEWS_E1
h1	HTIMES
h2	HWEEK
m1	NBN
m2	KEYWORDS

Then, the concept of state is implemented with the help of association *Meta_star* which connects a dimension to a fact with a defined period of validation. As we can see from the figure below, the fact *FNEWS_E1* is connected with three dimensions, namely *DTIMES_E1*, *DGEOGRAPHY_E1* and *DTHEME_E1*. The validation period between fact and dimension defined by a beginning and an ending date corresponds to the temporal interval associated to the state E_1.

Figure 8. Content of Meta_star

IDF	NAME	IDD	NAME	DATES	DATEE
f2	FNEWS_E1	d5	DTIMES_E1	2010	2014
f2	FNEWS_E1	d4	DGEOGRAPHY_E1	2010	2014
f2	FNEWS_E1	d6	DTHEME_E1	2010	2014

Next, the association denoted *Derive* is translated to a foreign key named *IDP* (shorted for ***ID-P****arent*) in the relational model. As we can see from the following two figures, the foreign key *IDP* permits to associate respectively the dimensions and the fact in state E_1 to the parent dimensions and parent fact in the original MDW before reduction. For instance, son dimension *DGEOGRAPHIY_E1* is associated to parent dimension named *DGEOGRAPHY*. In this way the reduction function denoted $Map(E_k) = E_{k+1}$ in a MDW is embodied in our metamodel.

Figure 9. Content of Meta_Dimension

ID	NAME	IDP
d1	DGEOGRAPHY	
d2	DTIMES	
d3	DTHEME	
d4	DGEOGRAPHY_E1	d1
d5	DTIMES_E1	d2
d6	DTHEME_E1	d3

Figure 10. Content of Meta_Fact

ID	NAME	IDP
f1	FNEWS	
f2	FNEWS_E1	f1

Moreover, as a fact contains one or several measures and a measure belongs to one and only on fact, a foreign key *IDF* (shorted for ***ID-Fact***) pointed to *Meta_Fact* is added in *Meta_Measure* after translating the conceptual model to the relational model (*cf.* figure 11).

Figure 11. Content of Meta_Measure

IDM	NAME	IDF	NAME
m1	NBN	f2	FNEWS_E1
m2	KEYWORDS	f2	FNEWS_E1

At last, the following figure shows the content of association *Meta_Level* which connects an attribute to a hierarchy on a dimension. As we can see from this figure, the attributes on dimension *DTIEMS_E1* are organized on two hierarchies: *HTIMES* and *HWEEK*. Each hierarchy organizes the attributes according to their granularity level. For instance, hierarchy *HTIMES* starts from attribute ID_{DATE} at position 1 and ends up with extremity attribute *ALL_T* at position 5. By definition, a parameter may be associated with one or several descriptive attributes (weak attributes) on the same level. That explains why two attributes of different types *MONTH* (parameter) and *MONTHALPH* (weak attribute) share the same position on hierarchy *HTIMES*.

Figure 12. Content of Meta_Level

IDD	NAME	IDH	NAME	IDA	NAME	POS	TYPA
d5	DTIMES_E1	h1	HTIMES	a1	IDDTIMES	1	PARAMETER
d5	DTIMES_E1	h1	HTIMES	a2	MONTH	2	PARAMETER
d5	DTIMES_E1	h1	HTIMES	a3	MONTHALPH	2	WEAK-ATTRIBUTE
d5	DTIMES_E1	h1	HTIMES	a4	QUARTER	3	PARAMETER
d5	DTIMES_E1	h1	HTIMES	a5	YEAR	4	PARAMETER
d5	DTIMES_E1	h1	HTIMES	a7	ALL_T	5	PARAMETER
d5	DTIMES_E1	h2	HWEEK	a1	IDDTIMES	1	PARAMETER
d5	DTIMES_E1	h2	HWEEK	a6	WEEK	2	PARAMETER
d5	DTIMES_E1	h2	HWEEK	a7	ALL_T	3	PARAMETER

EXPERIMENTAL ASSESSMENTS

Open issues

In the previous sections, we have defined a conceptual modeling supporting data reduction. By implementing our conceptual model in R-OLAP environment along with a generic meta-model, we have shown the feasibility of complete data reduction in MDW. However, the benefits of data reduction for the query execution efficiency still need to be proved through experimental assessments.

Objective

Since one objective of MDW is to accelerate data restitution in the OLAP context (Matteo Golfarelli & Rizzi, 2009), the *execution time* is undoubtedly a primordial indicator to evaluate the query execution efficiency. Therefore, our experimental assessments aim at studying if data reduction can improve query execution time, and if it does, to what extent the query *execution time* can be improved in reduced MDW. The query *execution time* can be presented in intrinsic format (e.g. *elapsed time* of each query) or in ratio (e.g. *relative gain of execution time*). As we aim at measuring relative amelioration or degradation levels of query execution times in reduced MDW, intrinsic indicators are of little use. Thus, the ratio *relative gain of execution time* is used as the first indicator of our experimental assessments.

Meanwhile, we also compare the *execution cost* for each query computed in unreduced and reduced MDW. The *execution cost* is a unitless index provided by *EXPLAIN PLAN* command of Oracle 11g. It is closely related to the query *execution time*: the higher the *execution cost*, the more important the theoretical *execution time* becomes in a given system. Yet, there is still a different between these two indicators: the *execution time* is less sensitive than *execution cost* when it comes to some relatively small datasets. For instance, accessing to a dataset with 100 tuples and another dataset with 5000 tuples could be simultaneous with today's highly powerful machines. As a result the *execution time* of queries concerning these two datasets will both be 1 which is the minimum value returned by *EXPLAIN PLAN* command. But the *execution cost* for accessing to the dataset with 5000 tuples will be far more important than to the dataset with 100 tuples, because the CPU and I/O costs for computing more tuples are higher than computing less tuples. Therefore, the *execution cost* may be in proportion to *execution time* for queries computed in relatively large datasets. But for small datasets, it is entirely possible that we obtain the same *execution time* with very different *execution cost*s. Due to this reason, we use *execution cost* as a complementary indicator.

Many factors should be taken into account while we compare the *execution time* and the *execution cost*.

- The first objective is to study if queries are more efficiently calculated within unreduced than reduced MDW. To do so we use two types of MDW: unreduced MDW and reduced MDW.
- The second objective is to study if the scale factor of MDW brings amelioration or degradation on query execution efficiency. We vary the number of tuples within non-temporal analysis axis. By consequence, different scale factors are defined for each type of implementation mentioned above.

- The third objective aims at understanding influences on query execution efficiency brought about by different implementation strategies. Thus, for each type of MDW we define two different implementations:
    - For unreduced R-OLAP databases, an implementation based on one single table and another implementation based on a fact table and different dimension tables are proposed.
    - For reduced R-OLAP databases, a denormalized R-OLAP implementation and a normalized R-OLAP implementation are proposed.
- The fourth objective consists in finding out if queries of different types can all benefit from data reduction. To this end, we propose 2 types of queries widely used:
    - Queries manipulating all the data of reduced database states (containing only joins and no selection criteria on non-temporal dimensions).
    - Queries manipulating a part of data in certain states (containing conditions restrictions on the data).
- The fifth objective lies in finding out if different aggregation and regrouping functions improve or decrease the query execution efficiency in unreduced and reduced MDW. We include five frequently used aggregation functions, namely *SUM*, *MAX, MIN, COUNT* and *AVG*. Meanwhile, influences of classical regrouping function (*GROUP BY*) and its extensions (*GROUP BY ROLLUP* and *GROUP BY CUBE*) are also taken into account during our experimental assessments.

Protocol

The experimental assessment consists in comparing some key indicators of query execution efficiency while applying different types of queries to different types of R-OLAP databases in a given implementation framework. The test protocol details our implementation framework, the choice of key indicators, the data collection for unreduced and reduced MDW and the list of queries to be executed.

Implementation framework

We carry out the experimental assessments with DBMS Oracle 11g in the following framework: OS: Red Hat Enterprise Linux Server release 5.9 (Tikanga), 2 x Intel(R) Xeon(R) E5410 @ 2.33GHz with 4 cores , 5GB of RAM and SAS 10K as disks.

Even though our framework does not provide the best DBMS tuning nowadays, it will not affect the results of experimental assessments neither. That is because our objective is not to see how fast a query can be computed in a particular machine: it aims to find out if different implementations of MDW with and without reduction have influence on query execution efficiency; and if it does, to what extent the efficiency of query execution can be improved within a certain implementation under a given implementation framework. In order to reduce the influence of irrelevant variables, different databases are implemented with the same DBMS (Oracle 11g) and the same DBMS tuning (OS, CPU, RAM, disk drive). To further minimize the influence brought by the implementation framework, we carry out several times the same experimental assessments and then calculate the average of experimental results.

Data Collection

There are several benchmarks designed for MDW nowadays, such as TPC-DS[1]. Based on a single predefined MDW, these benchmarks execute a set of queries in order to measure the performance of machine that hosts a MDW (Darmont, Bentayeb, & Boussaid, 2007). As part of our experimentations whose aim is to demonstrate the efficiency of reduced multidimensional schema rather than the capacity of a particular machine, a set of multidimensional schema would be more appropriate. Since the existing benchmarks do not permit to evaluate the impact of different modeling solutions to a given system (Darmont et al., 2007), we have decided to propose our own experiment environment by using synthetic data which turn out to be more adequate to our demonstration.

In order to make experimental assessments, we implement two types of R-OLAP databases with the Oracle DBMS and each type has two different implementations.

Unreduced R-OLAP databases

The first type of MDW corresponds to databases without reduction. Its first implementation is called *Global Star,* consists in an unreduced R-OLAP implementation based on 4 tables (*DTheme*, *Dgeography*, D*times* and *Fnews*). The second implementation is called *Global Table* in which we merge the three analysis axis (dimensions *Dtheme*, *Dgeography* and *Dtimes*) with the fact table (*Fnews*); consequently this implementation is composed of a single table that encompasses both fact and dimensions.

The population of analysis axes was done as follows:

- The dimension *Dtimes* contains all dates from 01/01/1990 to 31/12/2013.
- The two other dimensions contain random data defined by generation of synthetic data. In order to avoid the bias, allocation of random data was made so that father attribute of a hierarchy does not have the same number of sons while respecting the integrity constraints of strict hierarchies: any son attribute of a hierarchy has a single father attribute (*cf.* Table 4).

We have defined various scale factors of non-reduced databases by ranging the tuple numbers of the dimensions *Dtheme* and *Dgeography* from 10 to 40 tuples.

- |*Dgeography*| = 10, 20, 30, 40 tuples
- |*Dtheme*| = 10, 20, 30, 40 tuples
- |*Dtime*| = 8401 tuples (from 01/01/1990 to 31/12/2013)
- |*Fnews*| = | *Dgeography* | x | *Dtheme* | x | *Dtime* | = 840 100 to 13 441 600 tuples.

The following table describes different values associated to the attributes of non-temporal dimension:

[1] http://www.tpc.org/tpcds/

Table 4. Implementation details of the dimensions in Global Star and Global Table.

|*Dgeography*| x |*Dtheme*|	Contents of the dimension *Dgeography*	Contents of the dimension *Dtheme*
10x10	2 *Cities*, 2 *Countries* , 1 *Continent*	2 *Subthemes*, 2 *Themes*
20x20	4 *Cities*, 3 *Countries*, 2 *Continent*	4 *Subthemes*, 3 *Themes*
30x30	6 *Cities*, 4 *Countries*, 2 *Continent*	6 *Subthemes*, 4 *Themes*
40x40	8 *Cities*, 5 *Countries*, 3 *Continent*	8 *Subthemes*, 5 *Themes*

Even though the dimensions *Dtheme*, *Dgeography* and *Dtimes* are integrated in fact table of *Global Table*, the implementation details of MDW *Global Table* are the same as MDW *Global Star*.

Reduced R-OLAP databases

The second type of MDW corresponds to reduced databases. This type consists of three states according to the case study presented in this article (*cf.* figure 4). We have defined two implementations of reduced databases: a denormalized implementation and a normalized implementation (*cf* figure 13). The operations permitting to get the different states of MDW were implemented with the help of triggers in Oracle DBMS.

Figure 13. R-OLAP schemata of reduced MDW

E1

FNews(NBN_filtered, NBN, IDDTheme#, IDDTimes#, IDDGeography#)
DGeography(IDDGeography, City, Country, Continent)
DTheme(IDDTheme, Subtheme, Theme)
DTimes(IDDTime, Month, NameMonth, Quarter, Week, Year)

E2

FNews(NBN_filtered, NBN, Theme#, IDDTimes#, Country#)
DGeography(Country, Continent)
DTheme(Theme)
DTimes(Month, NameMonth, Quarter, Year)

E3

FNews(NBN, Continent#, Quarter#)
DGeography(Continent)
DTimes(Quarter, Year)

(a) Denormalized R-OLAP star schema

E1

FNews(NBN_filtered, NBN, Subtheme#, IDDTimes#, City#)
DGeography(IDDGeography, City#)
City(City, Country#)
Country(Country, Continent#)
Continent(Continent)
DTheme(IDDTheme, Subtheme#)
Subtheme(Subtheme, Theme#)
Theme(Theme)
DTimes(IDDTimes, Month#, Week#)
Month(Month, Quarter#)
Quarter (Quarter, Year#)
Week (Week, Year#)
Year(Year)

E2

FNews(NBN_filtered, NBN, Theme#, IDDTimes#, Country#)
DGeography(Country, Continent#)
Continent(Continent)
DTheme(Theme)
DTimes(Month, NameMonth, Quarter#)
Quarter(Quarter, Year#)
Year(Year)

E3

FNews(NBN, Continent#, Quarter#)
DGeography(Continent)
DTimes(Quarter, Year#)
YEAR(Year)

(b) Normalized R-OLAP snowflake schema

Query lists

We have defined two types of SQL queries manipulating different tables and different states. Table 5 contains queries with only joins, while table 6 shows queries containing both joins and restriction predicates on non-temporal dimensions.

The following queries are divided into 3 subgroups: the first subgroup contains queries from Q_1 to Q_8 that manipulate 1, 2 or 3 dimensions in one state; the second subgroup contains queries from Q_9 to Q_{12} manipulating 1, 2 or 3 dimensions in two states; the last subgroup contains queries Q_{13} and Q_{14}. The last two queries manipulate only one or two dimensions in three states because it is impossible to define a query manipulating 3 dimensions in 3 states, by the reason that the state denoted E_3 is only composed of 2 dimensions.

Table 5. Queries without restriction predicates on non-temporal dimensions.

	Queries (Aggregation function *SUM*)	**States**	**Dimension**
Q_1	Number of news for the last three years	E_1	DTime
Q_2	Number of filtered and un-filtered news in 2008	E_2	DTime
Q_3	Number of news before 2000	E_3	DTime
Q_4	Number of news by city from 2010 to 2012	E_1	DTime, DGeography
Q_5	Number of news by theme in each quarter from 2000 to 2005	E_2	DTime, DTheme
Q_6	Number of news by continent in each year before 2000	E_3	DTime, DGeography
Q_7	Number of news by city, country, subtheme and month in 2012	E_1	DTime, DGeography, DTheme
Q_8	Number of news by theme, country and continent from 2000 to 2005	E_2	DTime, DGeography, DTheme
Q_9	Monthly number of news since 2000	E_1; E_2	DTime
Q_{10}	Annual number of news per theme from 2002 to 2012	E_1; E_2	DTime, DTheme
Q_{11}	Number of news per year and continent from 1990 to 2009	E_2; E_3	DTime, DGeography
Q_{12}	Number of news by country, continent and theme from 2002 to 2012	E_1; E_2	DTime, DGeography, DTheme
Q_{13}	Number of news per year	E_1; E_2; E_3	DTime
Q_{14}	Number of news per year by continent	E_1; E_2; E_3	DTime, DGeography,

Three subgroups are proposed for queries with restriction predicates: the first subgroup includes queries from Q_1 to Q_3 that manipulate one dimension in one state; the second subgroup contains queries from Q_4 to Q_6 manipulating two dimension in two states; the third subgroup concerns queries from Q_7 to Q_9 manipulating one dimension in three states.

Table 6. Queries with restriction predicates on non-temporal dimensions.

	Queries (Aggregation function *SUM*)	**States**	**Dimension**
Q_1	Number of news about the subtheme X from 2010 to 2012	E_1	DTheme
Q_2	Number of news about the theme X from 2010 to 2012	E_1	DTheme
Q_3	Number of news about all the themes from 2010 to 2012	E_1	DTheme
Q_4	Number of news per month about the theme X in the country Y since 2000	E_1, E_2	DGeography, DTheme
Q_5	Number of news per month about the theme X in the continent Y since 2000	E_1, E_2	DGeography, DTheme
Q_6	Number of news per month about all the themes in the continent Y since 2000	E_1, E_2	DGeography, DTheme
Q_7	Number of news per year on the continent X	E_1; E_2; E_3	DGeography
Q_8	Number of news per year on the continent Y (the news of continent	E_1; E_2; E_3	DGeography

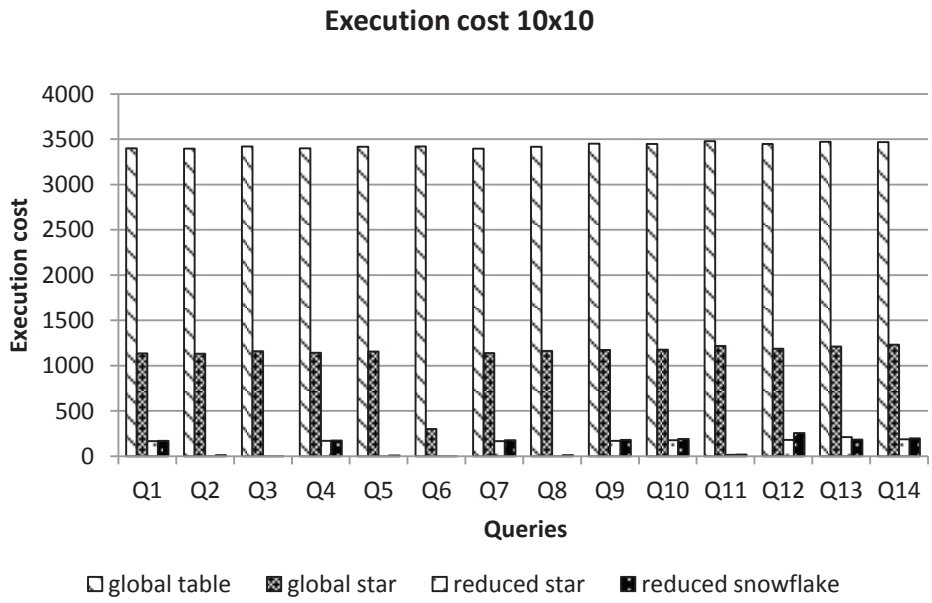
	X is three times more than those of continent Y)		
Q_9	Number of news per year on all the continents	E_1; E_2; E_3	DGeography

Results and discussions

Influence of data reduction

The first objective is to find out if reduced MDW can improve query execution efficiency. We execute queries without restriction predicates in 10x10 MDW. When it comes to a relatively small dataset, the execution cost is a more sensitive indicator than the execution time indicator. Therefore, in this section we compare only the execution cost of each query computed in the 10x10 MDW (the least voluminous MDW).

Figure 14. Execution cost of queries without restriction predicate in 10x10MDW

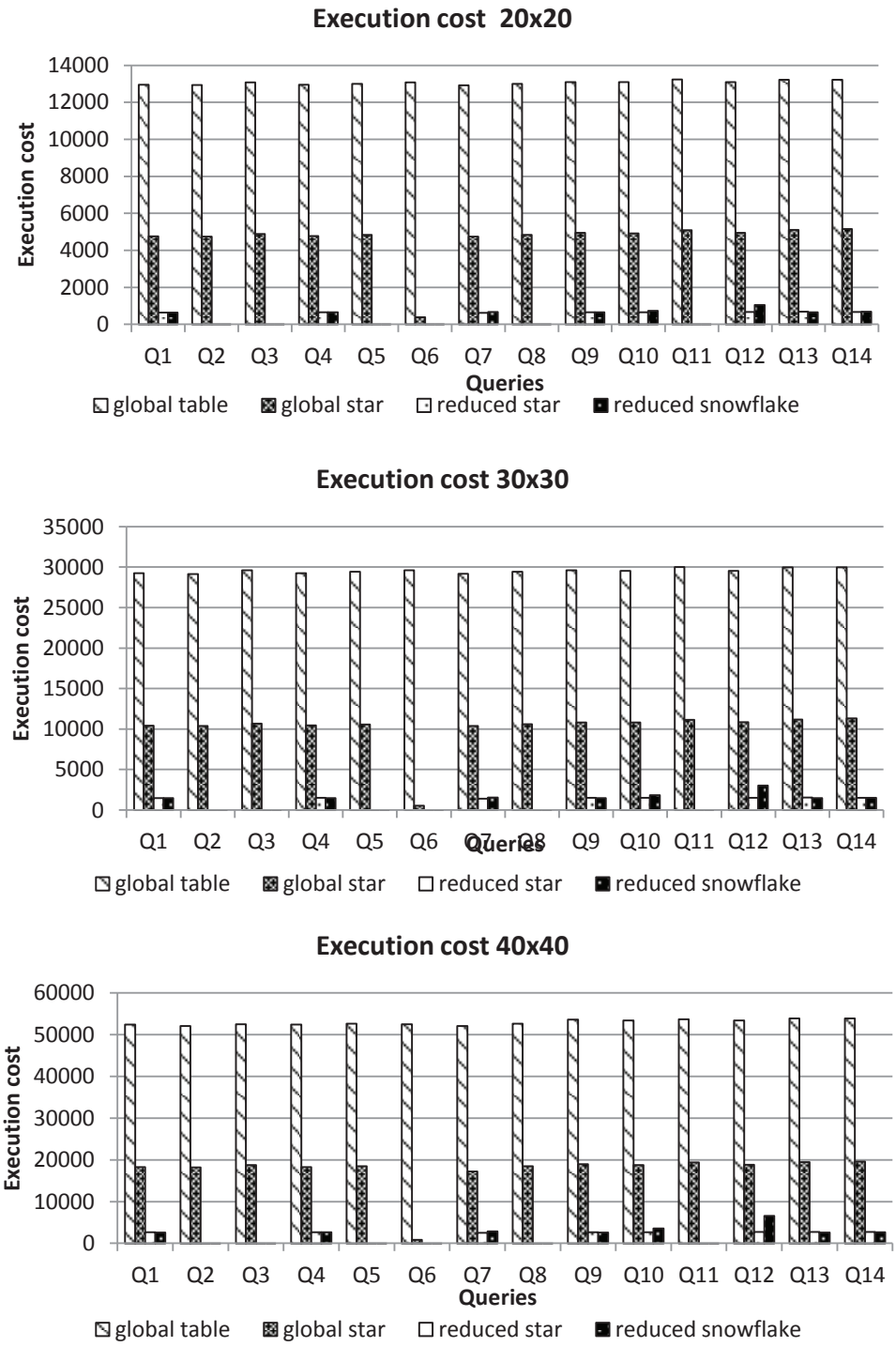


As we can see from the figure 14, regardless of different implementation strategies, the query execution cost in unreduced MDW (the column with stripe and the gray column in the figures above) is more important than in reduced MDW (the white and black columns in the figures above). We can safely conclude that data reduction does help improving query execution efficiency. However, the problem of querying performance is not so obvious in a small dataset, like the previously used 10x10 MDW. In the next section we increase the volume of MDW to see influence of MDW's scale factor on query execution efficiency.

Influence of MDW's scale factor

We vary the size of the MDW from 10x10 to 40x40. We execute the same 14 queries containing only joins and without restriction predicate. The following figures show the execution cost of each query computed in 20x20, 30x30 and 40x40 MDW.

Figure 15. Execution cost in 20x20, 30x30 and 40x40MDW



As we can see from the figure 15, queries are always more efficiently computed in reduced MDW than unreduced MDW. Within each version of MDW, the average gain of cost is between 90.86% in the MDW 10x10 and 91.68% in the MDW 40x40.

In the figures 16 and 17, we analyze the cardinalities of query results in order to verify if the cardinalities of results is related to query execution cost. Even though we have obtained diverse result cardinalities (*cf*. ordinate axis), the proportion stays similar from one MDW version to another.

Figure 16. Query results' cardinalities in 10x10 MDW

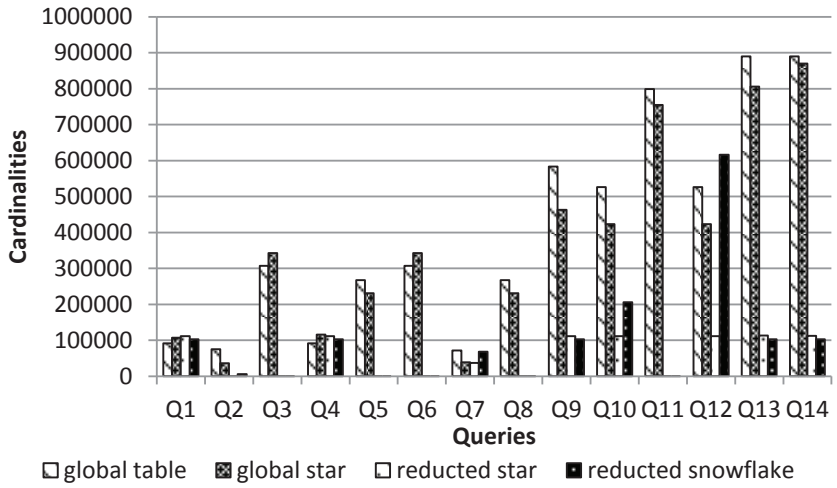
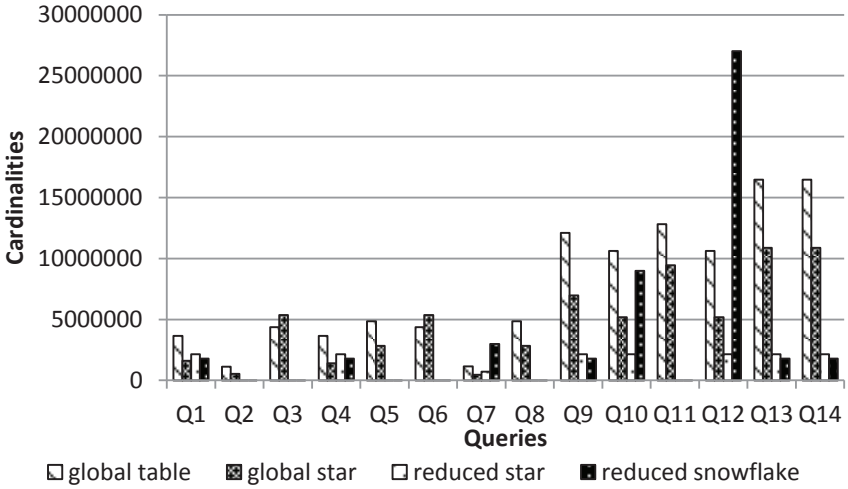
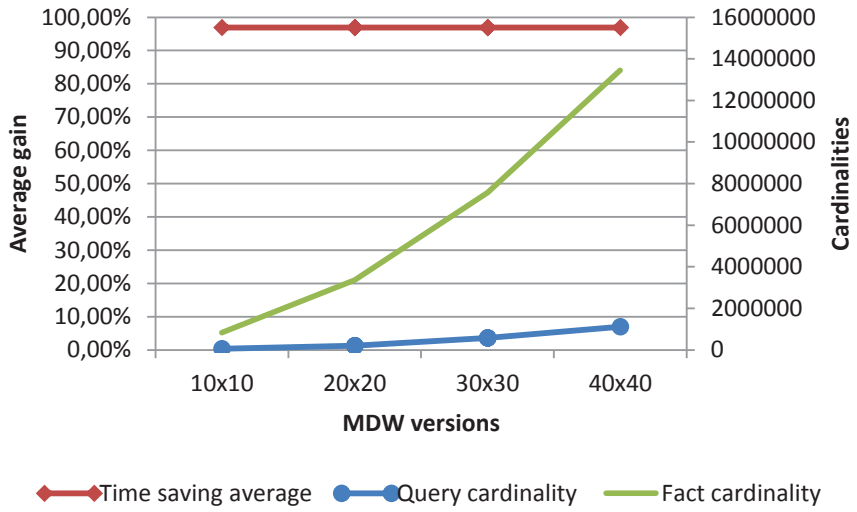


Figure 17. Query results' cardinalities in 40x40 MDW



In the figure 18, we can see whatever the database volume, the execution time gain is significant: over 97% (*cf*. red curve with diamond). We should also notice that in spite of the important augmentation of volume of fact tables (*cf*. smooth green curve) which leads to a slight augmentation of query results' cardinalities (*cf*. blue curve with dot), the average gains stay practically stable along with the cardinality augmentation: the average gain increases from 96.91% for the size 10 X 10 to 97.09% in size 40 X 40. Thus we can conclude that the more the database volume increases, the more the execution time gain in a reduced MDW is important.

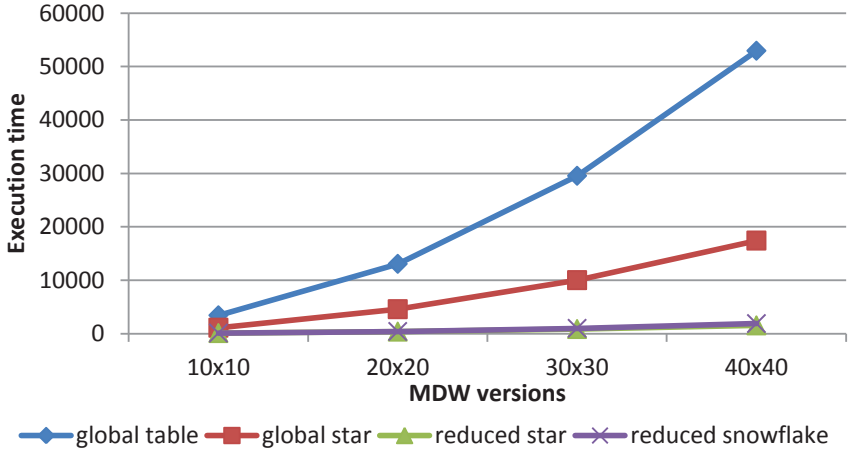
Figure 18. Average gain of execution time for the 4 versions of MDW



Influence of implementation strategy

Figure 19 shows the average execution time of queries computed in each implementation of unreduced and reduced MDW. As we can see from this figure, regardless of the MDW versions, the lowest execution times are always performed in reduced MDW called *Reduced Star*: MDW's content is reduced and the table number is limited. The highest average execution time for unreduced MDW ranges from 3432 (*Global Table* database 10 x 10) to 52941 (*Global Table* database 40 x40) and this average execution time has increased by 1443%. As for *Reduced Star* MDW, the average execution time ranges from 106 to 1554, and it has increased by 1366%, lower than 77% compared to the unreduced *Global Table* MDW.

Figure 19. Average execution time for the different versions of MDW



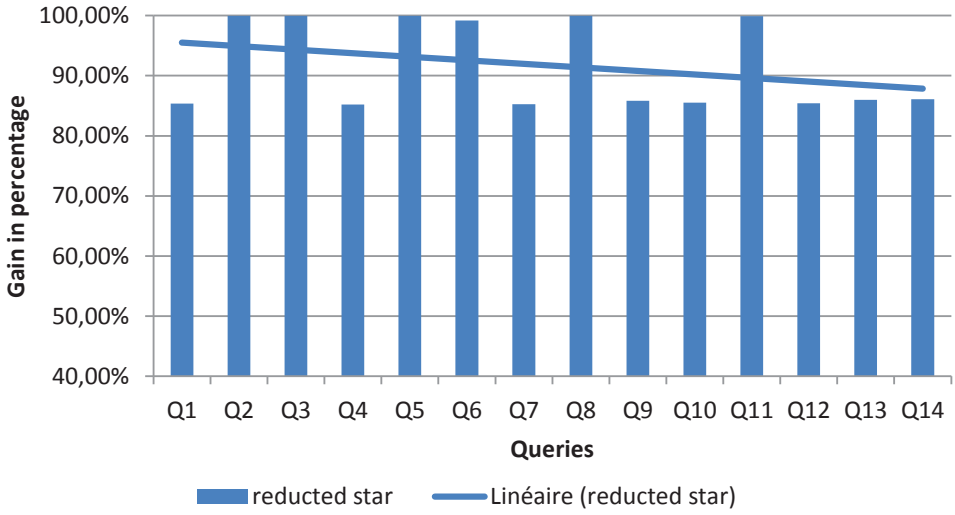
Comparison between Reduced Star and Global Table

Now we focus our study on the most voluminous version of MDW (40x40) in order to analyze the relative gain of execution time between *Reduced Star* and *Global Table* (*cf.* figure 20). As is

shown by the linear line which estimates the gain of execution time, we can notice that the more tables the query manipulates, the more important the gain of execution time becomes, all the while staying proportionally similar. However, by refining our study, we can find the more states the query manipulates, the less important the time earning becomes but it still stays in the same order of magnitude:

- For the queries manipulating a single state (Q1 to Q8), the gains varies from 95.12% (Q7) to 99.99% (Q3) with an average of 98.10%;
- For the queries manipulating two states (Q9 to Q12), the gains varies from 94.90% (Q10) to 99.96% (Q11) with an average of 96.61%;
- For the queries manipulating three states (Q13, Q14), the gains varies from 94.93% (Q7) to 94.94% (Q3) with an average of 94.935%.

Figure 20. Relative gain of execution time in percentage between Reduced Star and Global Table in 40x40 MDW

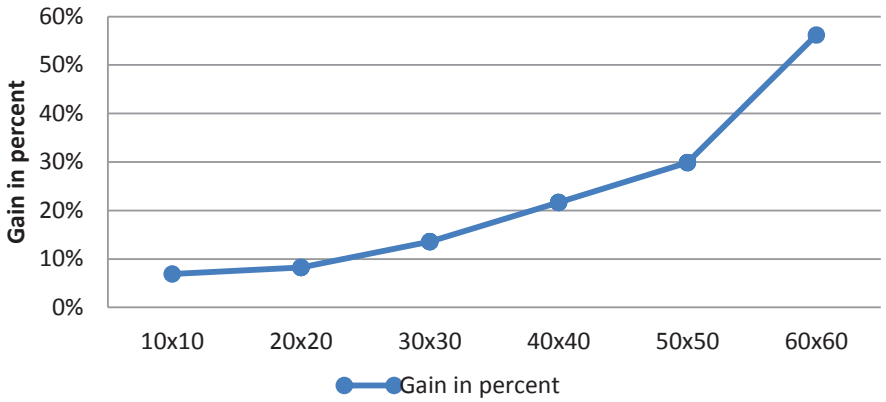


Comparison between Reduced Star and Reduced Snowflake

Within a relatively small scale of database volume (i.e. from 10x10 to 40x40 in figures 14 and 15), the difference of execution time between two implementations of reduced MDW (*Reduced Star* and *Reduced Snowflake*) is quite low. But when we increase MDW's volume until to 60x60, we find the gain of execution time in *Reduced Star* becomes more significant: about 56.2% (*cf.* figure 21). Notice that the only difference between these two implementations is in *Reduced Star* each dimension forms a unique table while in *Reduced Snowflake* dimensions are normalized so that each hierarchical level forms an independent table. The normalization in *Reduced Snowflake* regroups data in high granularity levels into new tables in order to avoid information redundancy. By consequence the space required for data storage is reduced in *Reduced Snowflake*, but the number of joins needed between tables while executing a given query becomes greater. Performing joins is less time-efficient when the database research certain scale factors. That explains why the difference of execution time between *Reduced Star* and *Reduced*

*Snowflak*e becomes more and more important with the augmentation of MDW's volume (from 7% in 10x10 to 56% in 60x60).

Figure 21. Relative gain of execution time in percentage between Reduced Star and Reduced Snowflake



Influence of query's type

In this section we aim at analyzing the impact of restriction criteria in queries. We focus only on 40 x 40 MDW which is the most voluminous. The queries of this second experimental assessment are defined in the table 6.

The following figures show execution costs and cardinalities of results for the four implementations. Contrary to our expectations, the gains between unreduced and reduced MDW remain in the same proportions whether we apply restriction or not. Indeed, this gain ranges from 95.11% (Q9) to 98.95% (Q4) with an average of 95.87% while the average gain of the queries without restriction predicates was over 97%. Moreover, whatever the scope of the restriction predicates (primary key, attribute containing different values or not), the standard deviation is not very high (0.1).

Figure 22. Execution cost of 9 queries containing restriction predicates

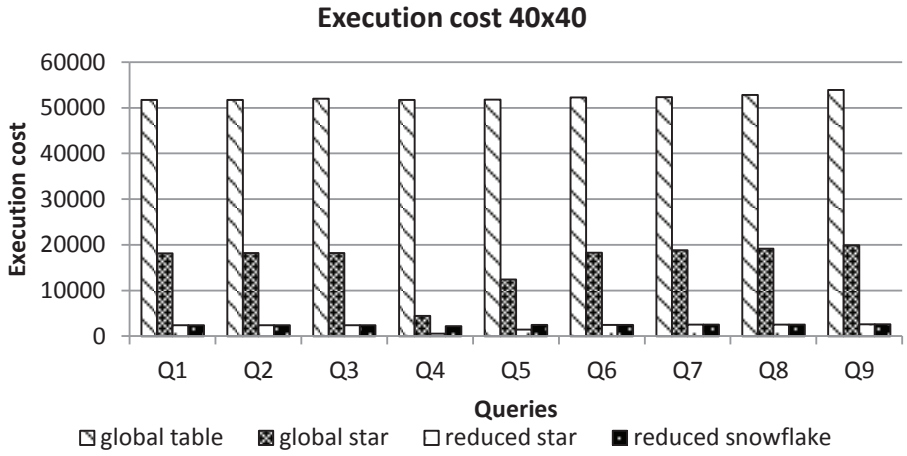
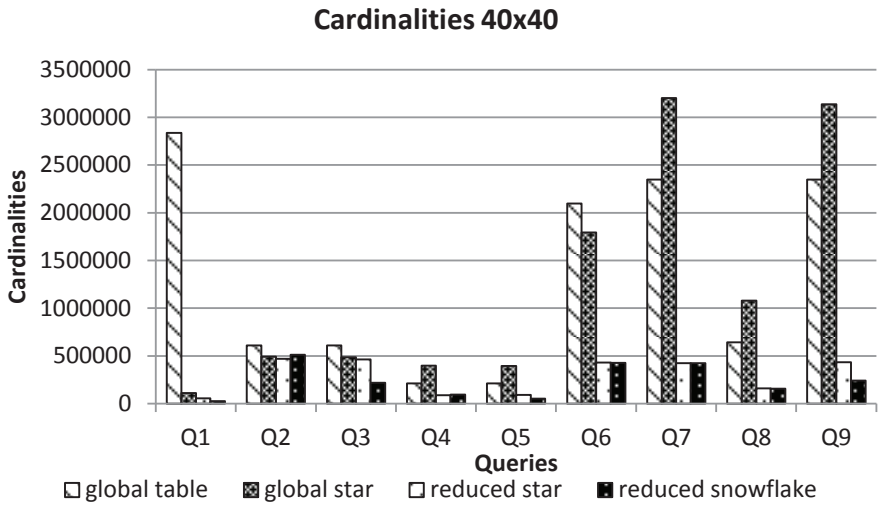


Figure 23. Cardinalities of 9 queries containing restriction predicates



In addition, even if execution costs of Q_7 and Q_8 are similar, we can notice that the cardinality of the result of Q_7 is three times higher than the cardinality of the result of Q8. This is because the DBMS should review all the tuples of the tables before returning the query result. So even if some strict selection criteria exclude lots of data from query result, the execution costs in reduced and unreduced MDW remain proportionally the same and reduced MDW are always more efficient than unreduced MDW.

Influence of aggregation and regrouping functions

This section consists in discussing if different aggregation and regrouping functions have influences on the improvement of query execution efficiency in reduced MDW.

Aggregation functions

The aggregation function we use in the previous experimental assessments is *SUM*. To see if aggregation functions have an influence on experiment results, we replace *SUM* with other commonly used aggregation functions in OLAP analysis, such as *MAX, MIN, COUNT, AVG*, and we rerun the entire experimentations. At last we get exactly the same execution cost, execution time and cardinalities regardless of what aggregation function we apply to the queries. Thus we can conclude that the common aggregation functions have no influence on query execution efficiency. For simplicity, we apply only the *SUM* function to the next experimental assessments which evaluate the impacts of different regrouping functions.

Regrouping functions

All 14 queries contain the classical SQL regrouping clause, namely *GROUP BY*. However, for certain queries, extensions of classical SQL regrouping clause are also applicable. For instance, Q12 can sustain the *GROUP BY ROLLUP* clause if subtotals of published news by city and continent are required. In the same way, extensions like *GROUP BY ROLLUP* and *GROUP BY CUBE* are both applicable to Q7 and Q8 to calculate subtotals for all possible combinations.

In order to study the impacts of different regrouping functions, we apply *GROUP BY ROLLUP* and/or *GROUP BY CUBE* to queries containing more than one analysis axis, namely Q5, Q6, Q7, Q8, Q10, Q11, Q12 and Q14. Before carrying out these tests, our intuition was the execution time between these three aggregation modes would be quite different and thus modify the gains of execution cost. Nevertheless, the execution cost and time of the same query are exactly the same regardless of different regrouping functions. The average gain is always in the same order of magnitude 94.86%.

CONCLUSION

This paper provides a contribution in the context of MDW. Our objective is to specify aggregated schema over time in order to retain only the data useful for decision support according to the needs of users. Firstly, we define a conceptual model that specifies MDW schemata composed of states varying over time. Each state consists of a star schema and is defined with a mapping function, itself defined with reduction operators based on an extension of classical OLAP operators adapted to the reduction context. We defined a minimum core of elementary operators { $\text{RollUp}^{\text{reduce}}$; $\text{Drop}^{\text{reduce}}$; $\text{Add}^{\text{reduce;}}$ $\text{Slice}^{\text{reduce}}$ } in order to carry out not only schema reduction operations but also instances reduction operations.

Secondly, an implementation in R-OLAP environment is described. The basis of our implementation is the metamodel allowing the management of reduced and unreduced MDW. The metamodel is presented both at conceptual and relational levels. By instantiating the metamodel, we can obtain reduced MDW composed of a set of states and unreduced traditional MDW.

Finally, we defined experimental assessments. Evaluating our solution consists in executing different queries in various environments: R-OLAP schema without reduction, single fact table schema without reduction as well as star and snowflake schemata with reductions. We use multidimensional databases with different sizes; the fact table size ranges from 840,100 to 13,441,600 tuples. Whatever the data warehouse volume, the execution time gain between unreduced and reduced databases is up to 97%. Moreover, the more the data warehouse volume increases, the more the execution cost and time gain is important. These gains remain in the same proportions when we apply restriction predicates or not on the queries, neither the cardinality of the result affects this gain. Finally, the execution cost is independent of the regrouping SQL clauses.

In the future, we intend to extend our conceptual proposal in order to integrate other operators in the definition of the reduction function. We also intend to propose a graphical tool for querying a MDW with a set of states and presenting the analysis' result through a graphical interface. This extension requires the definition of algebra along with a graphical language suitable for reduced data model. At last we wish to apply the principles of reduction to a reel data case study of analytic domain such as banking or insurance.

REFERENCES

Atigui, F., Ravat, F., Song, J., & Zurfluh, G. (2014). Reducing Multidimensional Data. In L. Bellatreche & M. K. Mohania (Eds.), *Data Warehousing and Knowledge Discovery* (Vol. 8646, pp. 208–220). Cham: Springer International Publishing.

Boly, A., Hebrail, G., Boly, A., & Goutier, S. (2007). Forgetting data intelligently in data warehouses. *IEEE International Conference on Research, Innovation and Vision for the Future* (pp. 220–227). New York: IEEE.

Chen, Y., Dong, G., Han, J., Pei, J., Wah, B. W., & Wang, J. (2002). OLAPing Stream Data: Is It Feasible? *Proc. Workshop on Research Issues in Data Mining and Knowledge Discovery* (pp. 53–58). Presented at the ACM SIGMOD, Madison, Wisconsin: ACM Press.

Darmont, J., Bentayeb, F., & Boussaid, O. (2007). Benchmarking data warehouses. *International Journal of Business Intelligence and Data Mining*, *2*(1), 79–104.

Garcia-Molina, H., Labio, W., & Yang, J. (1998). Expiring data in a warehouse. *24rd International Conference on Very Large Data Bases* (pp. 500–511). New York: Morgan Kaufmann Publishers Inc.

Golfarelli, M., Maio, D., & Rizzi, S. (1998). Conceptual design of data warehouses from E/R schemes. *Thirty-First Annual Hawaii International Conference on System Sciences* (Vol. 7, pp. 334–343). Kohala Coast, HI: IEEE Computer Society.

Golfarelli, M., & Rizzi, S. (2009). *Data warehouse design: modern principles and methodologies*. New York: McGraw-Hill.

Iftikhar, N., & Pedersen, T. B. (2010). Using a Time Granularity Table for Gradual Granular Data Aggregation. In B. Catania, M. Ivanović, & B. Thalheim (Eds.), *Advances in Databases and Information Systems* (Vol. 6295, pp. 219–233). Berlin, Heidelberg: Springer Berlin Heidelberg.

Iftikhar, N., & Pedersen, T. B. (2011). A rule-based tool for gradual granular data aggregation. *International Workshop on Data Warehousing and OLAP* (pp. 1–8). Glasgow, United Kingdom: ACM Press.

Kimball, R. (1996). *The data warehouse toolkit: practical techniques for building dimensional data warehouses*. New York: John Wiley & Sons.

Last, M., & Maimon, O. (2000). Automated dimensionality reduction of data warehouses. *International Workshop on Design and Management of Data Warehouses* (pp. 7.1–7.10). Stockholm, Sweden: CEUR Workshop Proceedings.

Okun, O., & Priisalu, H. (2007). Unsupervised data reduction. *Signal Processing*, *87*(9), 2260–2267.

Ravat, F., & Teste, O. (2000). A Temporal Object-Oriented Data Warehouse Model. In M. Ibrahim, J. Küng, & N. Revell (Eds.), *Database and Expert Systems Applications* (Vol. 1873, pp. 583–592). Berlin, Heidelberg: Springer Berlin Heidelberg.

Ravat, F., Teste, O., Tournier, R., & Zurfluh, G. (2008). Algebraic and Graphic Languages for OLAP Manipulations: *International Journal of Data Warehousing and Mining*, *4*(1), 17–46.

Skyt, J., Jensen, C. S., & Pedersen, T. B. (2008). Specification-based data reduction in dimensional data warehouses. *Information Systems*, *33*(1), 36–63.

Udo, I. J., & Afolabi, B. (2011). Hybrid Data Reduction Technique for Classification of Transaction Data. *Journal of Computer Science and Engineering*, *6*(2), 12–16.

Wang, X. S., Bettini, C., Brodsky, A., & Jajodia, S. (1997). Logical design for temporal databases with multiple granularities. *ACM Transactions on Database Systems*, *22*(2), 115–170.