Scientific Research Publishing

Journal of Computer and Communications, 2023, 11, 43-72
https://www.scirp.org/journal/jcc
ISSN Online: 2327-5227
ISSN Print: 2327-5219

Hierarchical Datacubes

Mickaël Martin Nevot, Sébastien Nedjar, Lotfi Lakhal

Laboratoire d'Informatique et Système CRNS UMR 7020, Aix-Marseille Université, Marseille, France
Email: mickael.martin-nevot@lis-lab.fr, sebastien.nedjar@lis-lab.fr, lotfi.lakhal@lis-lab.fr

How to cite this paper: Nevot, M.M., Nedjar, S. and Lakhal, L. (2023) Hierarchical Datacubes. *Journal of Computer and Communications*, **11**, 43-72.
https://doi.org/10.4236/jcc.2023.116004

Received: March 31, 2023
Accepted: June 27, 2023
Published: June 30, 2023

Copyright © 2023 by author(s) and Scientific Research Publishing Inc.
This work is licensed under the Creative Commons Attribution International License (CC BY 4.0).
http://creativecommons.org/licenses/by/4.0/

Open Access

Abstract

Many approaches have been proposed to pre-compute data cubes in order to efficiently respond to OLAP queries in data warehouses. However, few have proposed solutions integrating all of the possible outcomes, and it is this idea that leads the integration of hierarchical dimensions into these responses. To meet this need, we propose, in this paper, a complete redefinition of the framework and the formal definition of traditional database analysis through the prism of hierarchical dimensions. After characterizing the hierarchical data cube lattice, we introduce the hierarchical data cube and its most concise reduced representation, the closed hierarchical data cube. It offers compact replication so as to optimize storage space by removing redundancies of strongly correlated data. Such data are typical of data warehouses, and in particular in video games, our field of study and experimentation, where hierarchical dimension attributes are widely represented.

Keywords

ROLAP Cubing, Data Warehouse, Datacube, Big Data, Business Intelligence, Hierarchical Cube, Hierarchical Dimensions

1. Introduction and Motivations

Data warehouses ([1]), allow the storage of huge volumes of data accumulated over time in operational databases. In fact, recently, the evolution of technologies has led companies to store their data and thus preserve the "memory" of their activities. Data warehouses have been designed for this purpose. Unlike operational databases, they have some notable characteristics. First of all, they are not intended for the day-to-day management of the information system, but to provide genuine assistance in decision-making. Their users, the decision-makers, are therefore few in number and are interested not in the detail of the data but in general trends, according to this or that criterion, not explicitly stored. Warehouses' data are also peculiar. Often inserted in the warehouse when

refreshing, these data are persistent as they will not be updated. In addition, when inserted, these data are provided with a timestamp. They are said to be historicized. From the data deposits thus constituted, it was natural to seek to make the best use of it. Here again, it is not a question of formulating classic, simple and frequent requests fetching usually some tuples[1], but to carry out analyses that require aggregation of the data in order to identify major trends. Such queries are particularly expensive because they require scans of large volumes of data and they are inherently complex. However, being part of a decision support process (hence the acronym OLAP for online analytical processing[2]), the formulation of these queries depends on both prior knowledge and the needs of decision-makers ([3] [4]). OLAP is opposed to online transaction processing (OLTP). It therefore takes place on an *ad hoc* basis and ideally should be interactive. However, the underlying calculation is complex and time-consuming, hence the idea of pre-calculating the results.

However, as it is not obvious how to predict the decision-makers' assumptions, the preliminary calculations must consider all possible outcomes. It is this idea that encouraged us to integrate the hierarchical dimensions into these answers ([5]). To meet this need, we propose in this paper a complete redefinition of the framework and formal definition of traditional database analysis ([6]), the datacube, and its closed datacube through the prism of hierarchical dimensions. Hierarchies add their own modeling complexity to those of data warehouses ([7]), it is for this reason, and the consequent computation times that they induce, that few authors have been interested in this field. However, we believe that a work of framing and formal definition of hierarchical dimensions is judicious. Moreover, in general, we will consider two classes of OLAP operators. The first is dedicated to the data structure and allows its manipulation in order to retrieve remarkable information. The second is dedicated to the granularity of the data and proposes operators that aggregate and synthesise the information in one direction (roll-up), and which break them down or specify them in the other (drill down) ([8]). Dimensional hierarchies define the access paths in the data and allow the implementation of this second class of OLAP operators. It is also for this reason that studying the hierarchical dimensions is relevant although the volume of data generated, even greater than in the case of the classic datacube, must be understood by a particularly reduced approach to be completely usable.

Computing a hierarchical datacube, taking into account the hierarchical structure of the dimensions, can be judicious to obtain an optimized aggregation, a simplified analysis, an intuitive navigation and to help decision-making based on the hierarchies. Using the hierarchical structure of dimensions, it is possible to perform optimized aggregations by consolidating data at different levels of gra-

[1]Ordered collection of the values of an indefinite number of attributes relating to the same object.

[2]This term was defined by [2] through twelve rules to be applied to a database so that it is OLAP: 1) Multidimensional conceptual view; 2) Transparency; 3) Accessibility; 4) Constancy of response times; 5) Client/server architecture; 6) Dimension independence; 7) Management of sparse matrices; 8) Multi-user support; 9) No restrictions on inter- and intra-dimensional 10) Intuitive data manipulation 11) Flexible reporting 12) Unlimited number of dimensions and unlimited number of items on dimensions.

nularity. This allows data to be pre-aggregated at higher levels of the hierarchy, which reduces the size of the data cube and therefore improves query performance. A hierarchical datacube also helps simplify analysis by providing an aggregated view of data at different levels of granularity. The aim is to facilitate the understanding of trends and relationships between the different dimensions, avoiding the complexity of detailed data. The hierarchical structure of the dimensions in a hierarchical datacube also allows intuitive and easy navigation between the different levels of granularity. This allows users to explore data more efficiently and navigate between levels of detail and aggregations without extra effort. This can improve user experience and speed up data analysis. Finally, in many cases, the hierarchical structure of dimensions is meaningful from a decision-making perspective. By using a hierarchical datacube, it is possible to take this hierarchical structure into account to obtain more relevant decision information that is better aligned with the reality of the domain.

2. Use Cases

2.1. Open Match 3

For this paper, we introduce an example applied to a video game, the one we initially studied and experimented with when researching this contribution.

In this example, we will use the video puzzle-game[3] of the Match 3[4] type open source[5] Open Match 3, a web browser-based video game[6], developed by Giordano Ferdinandi, Stephen Surtees and Rachel Kehoe of Clockwork Chilli.

Initially, we focused on this type of video game because of its immense popularity, making the example more eloquent and its application relevant. In addition, besides its ethical side, this choice of open source allowed us to have access and to directly modify the source code of the video game in order to accommodate all the probes we wanted to listen to.

[3]A video game of puzzle is a type of video game based on reflection. Its name comes from the jigsaw puzzle, a game consisting of putting pieces in order. It can be a game in which the player has to move elements in a specific way. Many of these games are called tile-matching games.

[4]A match 3 game is a type of video game in which the player must combine three elements of the same colour or shape in order to eliminate them from the game board. These games are often considered as reflection games and are generally based on the combination of strategy and luck. Match 3 games can be played on various platforms, including computers, mobile phones, and game consoles, and they are generally very popular with different age groups. Match 3 games can be simple entertainment or they can offer increasingly complex challenges as the player progresses through the game.

[5]The open source term designates an approach to software development (and now extends to works of the mind) whose license respects criteria precisely established by the Open Source Initiative (https://opensource.org/) and in which the source code is freely available and can be used, modified and distributed by anyone. The aim of the open source approach is to allow more people to contribute to the development and improvement of software, which can lead to increased innovation and improved quality. Open source licences prescribe the conditions under which source code can be used, modified and distributed. Some licences require that changes to the source code be published so that others can benefit from them, while other licences allow the user to keep their changes private).

[6]A browser-based video game is a type of online game that can be played through a web browser and does not require installation or downloading.

2.2. Relation Example

Let's consider the OM3 data warehouse (*cf.* Figure 1) of the Open Match 3 video game (*cf.* Figure 2).

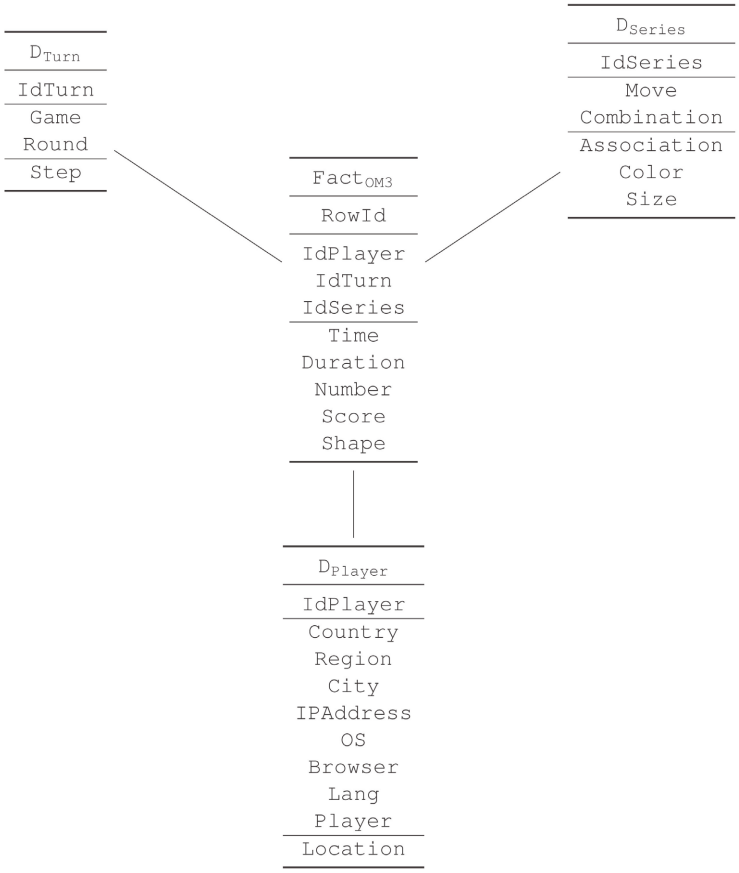

Figure 1. Star schema of data warehouse OM3.

Figure 2. Video game open match 3.

A D_{Turn} is identified by IdTurn and classified by Game and by Round. These attributes constitute a hierarchical structure, called Step (*cf.* Table 1).

A D_{Series} is identified by IdSeries and characterized by a Move causing one (or more, then called a chain) Combination of an Association, characterized by a matching Color (among the 4) and a Size of at least three elements. From an Association of four elements or more, bonus points and special elements (explosion, rayon destructeur, etc., *cf.* Figure 3) are granted (*cf.* Table 2).

Un D_{Player} is identified by IdPlayer and classified by its Location (full). The most general location type is a Country (C), which includes a Region (R), which itself includes a City (C), specified by an IPAddress (I), itself specified by an OS (O, *i.e.* an operating system), a Browser (B), then a Lang (L) thus, at last, a Player (P) name. The values of the different attributes of D_{Player} are coded as follows (*cf.* Table 3):

Country		Region		City		IPAddress	
France	=1	*IDF*	=2	*Paris*	=3	92.88.91.80	=4
		PACA	=9	*Marseille*	=10	139.124.242.125	=11

OS		Browser		Lang		Player	
Windows	=5	*Chrome*	=6	*fr*	=7	P_1	=8
Linux	=12	*Opera*	=13	*en*	=14	P_2	=15
Mac OS	=16	*Firefox*	=17	*es*	=18	P_3	=19

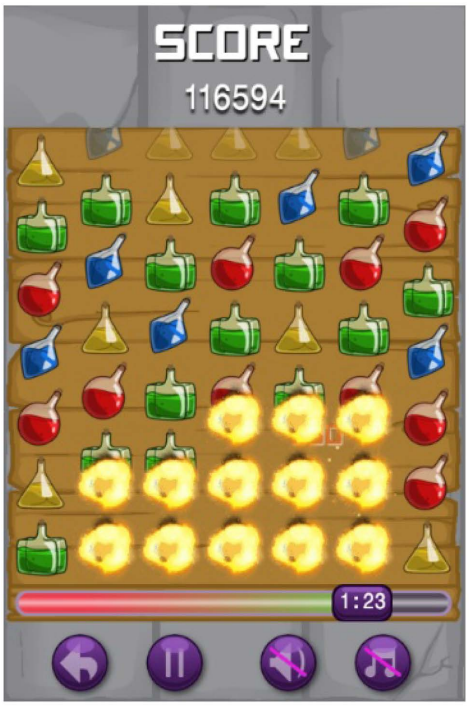

Figure 3. Video game open Match 3, example of an explosion.

Table 1. Relation of the dimension D_{Turn}.

IdTurn	Game	Round	Step
1	1		S_1
2	1	2	$S_{1\text{-}1}$
3	1	3	$S_{1\text{-}2}$
4	4		S_2
5	4	5	$S_{2\text{-}1}$
6	4	6	$S_{2\text{-}2}$
7	4	7	$S_{2\text{-}3}$
8	8		S_3
9	8	9	$S_{3\text{-}1}$
10	8	10	$S_{3\text{-}2}$
11	8	11	$S_{3\text{-}3}$

Table 2. Relation of the dimension D_{Series}.

IdSeries	Move	Combination	Association	Color	Size
1	1		A_1		
2	1	2	$A_{1\text{-}1}$	*red*	3
3	1	3	$A_{1\text{-}2}$	*blue*	3
4	4		A_2		
5	4	5	$A_{2\text{-}1}$	*green*	4
6	4	6	$A_{2\text{-}2}$	*yellow*	3
7	4	7	$A_{2\text{-}3}$	*red*	3
8	4	8	$A_{2\text{-}4}$	*blue*	3
9	4	9	$A_{2\text{-}5}$	*yellow*	3
10	4	10	$A_{2\text{-}6}$	*green*	4
11	11		A_3		
12	11	12	$A_{3\text{-}1}$	*green*	3
13	11	13	$A_{3\text{-}2}$	*red*	3
14	11	14	$A_{3\text{-}3}$	*red*	4
15	11	15	$A_{3\text{-}4}$	*green*	3
16	11	16	$A_{3\text{-}5}$	*yellow*	4
17	11	17	$A_{3\text{-}6}$	*yellow*	3
18	11	18	$A_{3\text{-}7}$	*yellow*	3
19	19		A_4		
20	19	20	$A_{4\text{-}1}$	*green*	3
21	21		A_5		
22	21	22	$A_{5\text{-}1}$	*red*	3

Continued

23	23		A_6		
24	23	24	$A_{6\text{-}1}$	*green*	5
25	25		A_7		
26	25	26	$A_{7\text{-}1}$	*yellow*	3
27	25	27	$A_{7\text{-}2}$	*yellow*	3
28	25	28	$A_{7\text{-}3}$	*red*	3
29	25	29	$A_{7\text{-}4}$	*red*	3
30	25	30	$A_{7\text{-}5}$	*yellow*	3
31	25	31	$A_{7\text{-}6}$	*red*	4
32	25	32	$A_{7\text{-}7}$	*green*	3
33	25	33	$A_{7\text{-}8}$	*yellow*	3
34	25	34	$A_{7\text{-}9}$	*yellow*	3
35	25	35	$A_{7\text{-}10}$	*green*	3
36	25	36	$A_{7\text{-}11}$	*green*	4
37	25	37	$A_{7\text{-}12}$	*blue*	3

Table 3. Relation of the dimension D_{Player}.

IdPlayer	C	R	C	I	O	B	L	P	Location
1	1								*France*
2	1	2							*IDF*
3	1	2	3						*Paris*
4	1	2	3	4					92.88.91.80
5	1	2	3	4	5				*Windows*
6	1	2	3	4	5	6			*Chrome*
7	1	2	3	4	5	6	7		*fr*
8	1	2	3	4	5	6	7	7	P_1
9	1	9							*PACA*
10	1	9	10						*Marseille*
11	1	9	10	11					139.124.242.125
12	1	9	10	11	12				*Linux*
13	1	9	10	11	12	13			*Opera*
14	1	9	10	11	12	13	14		*en*
15	1	9	10	11	12	13	14	15	P_2
16	1	9	10	11	16				*Mac OS*
17	1	9	10	11	16	17			*Firefox*
18	1	9	10	11	16	17	18		*es*
19	1	9	10	11	16	17	18	19	P_3

The table of facts Fact_{OM3} is composed of a Time remaining before the end of the game (naturally decreasing, but also increasing to a maximum with the Score, it is thus a decisional element), a Duration of realization, a Number of elements in correspondence, a Score (number of points won) total and a description of Shape (0.5 for horizontal, 1 for vertical, and between these two values for mixed) for a IdPlayer (IdP), a IdTurn (IdT) and a IdSeries (IdS). The values of the different attributes Fact_{OM3} are coded as follows (*cf.* Table 4):

IdP		IdT		IdS	
P_1	=8	S_1	=1	A_1	=1
P_2	=15	S_2	=4	A_2	=4
P_3	=19	S_3	=8	A_3	=11
				A_4	=19
				A_5	=21
				A_6	=23
				A_7	=25

We only give a representation of a hierarchical dimension for D_{Player} (*cf.* Figure 4), D_{Turn} and D_{Series} being hierarchical dimensionswhich are much less well founded than this one. The maximum element ALL_{Player} of this representation is defined by 3.5 in Subsection 3.2.

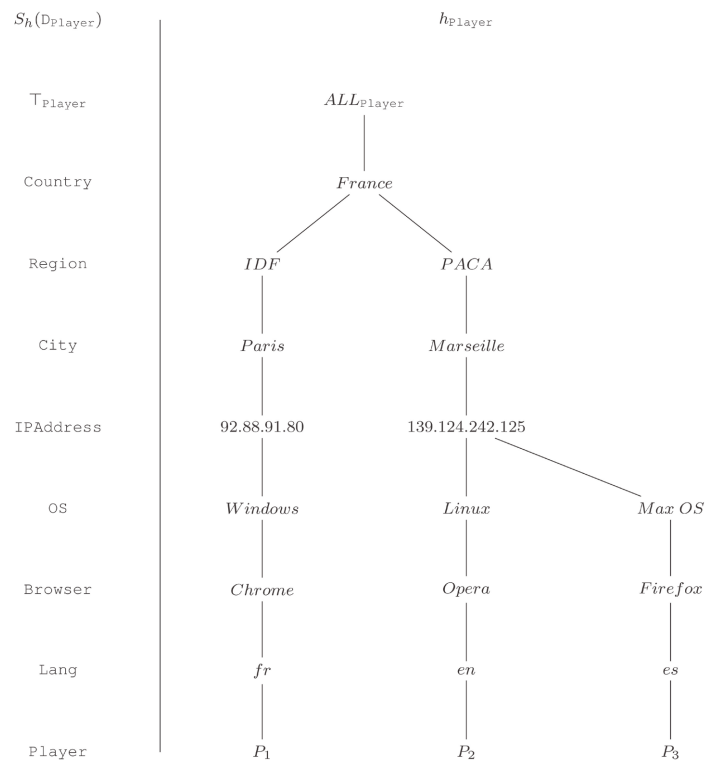

Figure 4. Dimension of D_{Player}.

Table 4. Relation of the table of facts Fact_{OM3}.

RowId	IdP	IdT	IdS	Time	Duration	Number	Score	Shape
1	P_1	S_1	A_1	6.32	2.85	3.5	700	0.5
2	P_1	S_1	A_2	18.9	1.95	3.83	2300	0.5
3	P_2	S_2	A_3	26.39	1.7	3.43	2400	0.71
4	P_2	S_2	A_4	4.1	2.07	3	300	0.5
5	P_2	S_2	A_5	7.38	3.68	3	600	0.5
6	P_3	S_3	A_6	2.14	2.15	3	300	1
7	P_3	S_3	A_7	56.04	2.25	3.17	3800	0.5

3. Hierarchical Dimension

3.1. Types of Hierarchies

Dimensional hierarchies are formed by the set of attributes of a dimensional relation having a hierarchical relationship ($a \in a'$). They correspond to relations of type 1 to several and offer the possibility of using the class of roll up and drill down ([8]).

The literature provides an overview of the different types of hierarchies ([1]) that we propose to present.

3.1.1. Simple Hierarchy

Simple hierarchies have levels that can be considered as lists and their instances form a tree. These hierarchies can be symmetrical if the tree of instances is balanced or asymmetrical if it is not.

3.1.2. Strict Hierarchy and Non-Strict Hierarchy

In the framework of a strict hierarchy, the generalisation of a value at a given level only leads to, at most, one value.

Example 3.1 The month-level generalisation of the date July, 14th 1789 is July.

In practice, there are many situations where this type of hierarchy is too restrictive. Typically, a video game can be of several game types. Such hierarchies are called non-strict.

3.1.3. Multiple Hierarchy

Multiple hierarchies make it possible to model situations where the different levels are no longer lists but directed graphs without cycles.

Example 3.2 Let us consider a time dimension. There are two ways of generalising the day level: on the week level or the month level, although these two levels are not comparable (*i.e.* one is not a generalisation of the other).

3.1.4. Parallel Hierarchies

For dimensions with only one attribute, there can only be one hierarchy per dimension. In practice, it is common for a single dimension to be composed of sever-

al attributes useful for the analysis.

If these attributes have an associated hierarchy, the dimensional hierarchy is said to be parallel and can be considered as a special case of multiple hierarchies. A parallel hierarchy can be composed of other types of hierarchies. Two categories of parallel hierarchies are generally distinguished:

- Independent parallel hierarchy: there is no common level between the different hierarchies composing it;
- Parallel hierarchy: there are common levels between the different hierarchies composing it; the hierarchy is said to be independent.

3.2. Formal Framework of a Hierarchical Dimension

Let r be a relation of schema $\mathcal{R}$. In addition to the attributes of $\mathcal{R}$, we will consider an additional attribute *RowId* qui which is implicit. The values of this attribute serve as a unique identifier for each tuple and are assigned at the time of their insertion. We note t_x the tuple such that $t[RowId] = x$.

The attributes of $\mathcal{R}$ are divided into two sets:

1) $\mathcal{D}$ is the set of attributes of hierarchical dimensions, also called hierarchical categories. Moreover, the attributes of $\mathcal{D}$ are ordered.

2) $\mathcal{M}$ is the set of measured attributes, on which the aggregative or statistical functions are applied.

We simply call dimension a hierarchical dimension attribute. Likewise, we call hierarchy a hierarchy of values.

Definition 3.1 (Dimension)—In r, we call a dimension, a dimensional attribute, or a category, d_i. $\forall d_i \in \mathcal{D}$, $r(d_i)$ is the projection of r onto d_i.

A dimension is characterised by a structure (*cf.* Definition 3.2) and a hierarchy (*cf.* Definition 3.3).

We call $\mathcal{D}$ the set of dimensions d_i of r.

Example 3.3 Let's consider the data warehouse OM3. Its set of dimensions $\mathcal{D}$ is:

$$\mathcal{D} = \{\mathrm{D_{Player}}, \mathrm{D_{Turn}}, \mathrm{D_{Series}}\}$$

Definition 3.2 (Structure of a dimension)—For each dimension $d_i \in \mathcal{D}$, we define the structure of a dimension, or schema, S_h as follows: $\forall d_i \in \mathcal{D}$, $S_h(d_i)$ is the structure of the dimension d_i.

Example 3.4 Let's consider the set of hierarchical dimension structures:

$$S_h(\mathrm{D_{Player}}) = (\mathrm{T_{Player}}, \mathrm{Country}, \mathrm{Region}, \mathrm{City}, \mathrm{IPAddress}, \mathrm{OS}, \mathrm{Browser}, \mathrm{Lang}, \mathrm{Player})$$

$$S_h(\mathrm{D_{Turn}}) = (\mathrm{T_{Turn}}, \mathrm{Game}, \mathrm{Round})$$

$$S_h(\mathrm{D_{Series}}) = (\mathrm{T_{Series}}, \mathrm{Move}, \mathrm{Combination})$$

Definition 3.3 (Hierarchy)—For each dimension $d_i \in \mathcal{D}$, we define h_i the hierarchy associated with that dimension, which we will simply call hierarchy.

Definition 3.4 (Level of a hierarchy)—For each hierarchy h_i, we define

$h_i(e)$, the level, or tier, or echelon, of this hierarchy.

Definition 3.5 (Maximum element of a hierarchy)—A hierarchy h_i associated with the dimension $d_i \in \mathcal{D}$ contains a maximum element denoted T_i defined by the value ALL_i. This value takes the idea of the *ALL* from [9], which represents the generalization of all the values of the domain of a dimension.

Example 3.5 Considering the hierarchy h_{Player} of the dimension $\mathrm{D}_{\text{Player}}$ (*cf.* Figure 4), the value of its maximum element is ALL_{Player}.

Definition 3.6 (Depth of a hierarchy)—For each hierarchy h_i, we define its *Depth* as follows: $\forall h_i \in d_i$, $d_i \in \mathcal{D}$, $Prof(h_i)$ is the depth of the hierarchy h_i, *i.e.* its number of levels.

Example 3.6 Considering the hierarchy h_{Player} of the dimension $\mathrm{D}_{\text{Player}}$ (*cf.* Figure 4), $Depth(h_{\text{Player}}) = 9$.

Definition 3.7 (Domain of a dimension)—We define the domain of a dimension *Dom* as follows: $\forall d_i \in \mathcal{D}$, $Dom(d_i)$ is the domain of dimension d_i.

Definition 3.8 (Domain of a level of a dimension)—We define the domain of a level of a dimension, or dimensional footprint, *Dom* as follows:

- $\forall d_i \in \mathcal{D}$, $\forall e \in S_h(d_i)$, $Dom(d_i, e)$ is the domain of the dimension d_i for the level *e*, *i.e.* $Dom(d_i, e)$ corresponds to the set of possible values for the level *e* of the hierarchy h_i;
- $\forall d_i \in \mathcal{D}$, $Dom(h_i) = \bigcup_{e \in S_h(d_i)} Dom(h_i, e)$.

Definition 3.9 (Dimensional table)—In the "star" data model used in data warehouses (or datamart), each hierarchy can be represented by a table, of dimension D_{d_i} corresponding to the dimension eponymous (the two concepts are mingled), relational schema $S_h(d_i)$.

Definition 3.10 (Dimensional tuple)—For each $x \in Dom(d_i)$, we define the dimensional tuple t_x corresponding to the value *x* in the table D_{d_i} is made up of the list of ancestors of *x* in h_i.

Likewise, for each $x \in Dom(d_i, e)$, we define the dimensional tuple t_x corresponding to the value *x* for the level *e* of the hierarchy h_i in the table D_{d_i} is also made up of the list of ancestors of *x* in h_i.

We simply call tuple a dimensional tuple.

Example 3.7 For $x = P_2$, the tuple t_x corresponding in the relation $\mathrm{D}_{\text{Player}}$ is:

$$t_x = (France, PACA, Marseille, 139.124.242.125, Linux, Opera, en, P_2)$$

For $x = 139.124.242.125$, the tuple t_x corresponding in the relation $\mathrm{D}_{\text{Player}}$ is:

$$t_x = (France, PACA, Marseille, 139.124.242.125, NULL, NULL, NULL, NULL)$$

3.3. Formal Definition of a Hierarchical Dimension

3.3.1. Multidimensional Space

Definition 3.11 (Multidimensional space)—The multidimensional space of *r* is noted and redefined with hierarchical dimensions as follows:

$$Space(r) = \left\{ \times_{d_i \in \mathcal{D}} Dom(h_i) \cup \left\{ ALL_i, \cdots, ALL_j \right\} \right\} \cup \left\{ (\varnothing, \cdots, \varnothing) \right\}$$

where × symbolizes the Cartesian product, and $(\varnothing, \cdots, \varnothing)$ the combination of empty values. is a tuple and represents a multidimensional pattern.

The value of each maximal element ALL_i of a hierarchy h_i is naturally contained in the hierarchy, in the associated domain $Dom(h_i)$, and thus also in $Space(r)$.

Example 3.8 The multidimensional space of the OM3, data warehouse, and its table of facts (*cf.* **Table 4**), is illustrated in **Table 5**. For the sake of clarity and conciseness, not all tuples in the multidimensional space are represented (there would be several thousand), and special values are abbreviated: ALL_{Player} to ALL_{p}, ALL_{Turn} to ALL_{T} and ALL_{Series} to ALL_{S}.

Tuples identified by values 1, …, 13 et 31 are possible tuples for *r* (because all their values are real), even if the tuple identified by value 31 is not explicitly in the relation *r*. In this tuple, the symbol $\varnothing$ means "empty value". The other tuples (identifiers 14, …, 30) cannot be tuples of *r* because they contain at least one ALL_i value. They therefore convey information at a more aggregated level of detail than the previous ones.

3.3.2. Specialisation Orders

The multidimensional space of *r* is structured by the specialisation relation between tiles. This order was originally introduced by [10] [11] in the context of concept learning.

The ordered set $CL(r) = \langle Space(r), \preceq_s \rangle$ is a complete lattice called cube lattice.

Definition 3.12 (Intradimensional specialization order)—Let's consider $x, y \in Dom(h_i)$:

$$x \preceq_{d_i} y \Leftrightarrow x \text{ is an ancestor of } y \text{ on the hierarchy } h_i.$$

Definition 3.13 (Multidimensional specialization order)—Let's consider $t, t' \subseteq Space(r)$, we define the order relation $\preceq_s$ as follows:

$$t \preceq_s t' \Leftrightarrow \forall d_i \in \mathcal{D}, t[d_i] \preceq_{d_i} t'[d_i]$$

3.3.3. Functions

Definition 3.14 (Dimensional *Attribute* function) For a tuple t_x de d_i, the function $Attribute_{d_i}$ returns the set of attributes whose value is different from *NULL*.

Let's consider t_x a tuple of d_i:

$$Attribute_{d_i}(t_x) = \left\{ A \in t_x \mid t_x[A] \neq NULL \right\}$$

Example 3.9 For the dimension D_{Player}:

If $t_x = (France, PACA, Marseille, 139.124.242.125, Linux, Opera, en, P_2)$

Then $Attribute_{\text{D}_{\text{Player}}}(t_x) = \{France, PACA, Marseille, 139.124.242.125, Linux, Opera, en, P_2\}$

Table 5. Multidimensional space of the data warehouse OM3.

RowId	IdP	IdT	IdS
1	*France*	S_1	A_1
2	*France*	S_1	$A_{1\text{-}1}$
3	*France*	S_1	$A_{1\text{-}2}$
4	*France*	S_1	A_2
…	…	…	…
5	*France*	$S_{1\text{-}1}$	A_1
…	…	…	…
6	*IDF*	S_1	A_1
…	…	…	…
7	*es*	$S_{3\text{-}3}$	$A_{7\text{-}12}$
8	P_1	S_1	A_1
…	…	…	…
9	P_1	S_1	A_2
…	…	…	…
10	P_3	S_3	A_7
…	…	…	…
11	*France*	S_1	ALL_S
…	…	…	…
12	*PACA*	$S_{3\text{-}3}$	ALL_S
13	*Marseille*	S_1	ALL_S
…	…	…	…
14	P_1	S_1	ALL_S
…	…	…	…
15	*France*	ALL_T	A_1
16	*France*	ALL_T	$A_{1\text{-}1}$
…	…	…	…
17	*fr*	ALL_T	$A_{7\text{-}12}$
19	P_1	ALL_T	A_1
…	…	…	…
19	ALL_P	S_1	A_1
20	ALL_P	S_1	$A_{1\text{-}1}$
…	…	…	…
21	*France*	ALL_T	ALL_S

Continued

22	*IDF*	ALL_T	ALL_S
23	*Paris*	ALL_T	ALL_S
…	…	…	…
24	ALL_P	S_1	ALL_S
…	…	…	…
25	ALL_P	ALL_T	A_7
…	…	…	…
26	ALL_P	ALL_T	ALL_S
27	∅	∅	∅

If $t_y = (France, PACA, Marseille, 139.124.242.125, NULL, NULL, NULL, NULL)$

Then $Attribute_{D_{Player}}(t_y) = \{France, PACA, Marseille, 139.124.242.125\}$.

3.3.4. Operators

Definition 3.15 (Dimensional *min*/*max* operators) - We define the operators *min* and *max* as follows:

$$min_{\preceq_{d_i}}(d_i) = \{x \in Dom(h_i, e) \mid \nexists e' \in S_h(d_i) : e' < e\}$$

$$max_{\preceq_{d_i}}(d_i) = \{x \in Dom(h_i, e) \mid \nexists e' \in S_h(d_i) : e' > e\}$$

Thus, $min_{\preceq_{d_i}}(d_i)$ represents the set of possible values of the highest (general) level of the hierarchical dimension structure. In a similar way, $max_{\preceq_{d_i}}(d_i)$ represents the set of possible values of the lowest (specific) level of the hierarchical dimension structure.

Example 3.10 For the dimension D_{Player}:

$$min_{\preceq_{DPlayer}}(D_{Player}) = \{France\}$$

$$max_{\preceq_{DPlayer}}(D_{Player}) = \{P_1, P_2, P_3\}$$

The *min*/*max* operators can be overloaded in order to be applied to a set of tuples T of dimension d_i, noted respectively $min_{\preceq_s}(T)$ and $max_{\preceq_s}(T)$, by being implemented as follows:

$$min_{\preceq_{d_i}}(T) = \forall x \in Dom(h_i, e), t_x \mid \nexists e' \in S_h(d_i) : e' < e$$

$$max_{\preceq_{d_i}}(T) = \forall x \in Dom(h_i, e), \{t_x \mid \nexists e' \in S_h(d_i) : e' > e\}$$

Example 3.11 - For the dimension D_{Player}, if $T = \{t_x, t_y\}$ with:

$$t_x = (France, PACA, Marseille, 139.124.242.125, Linux, Opera, en, P_2)$$

$$t_y = (France, PACA, Marseille, 139.124.242.125, Mac\ OS, Firefox, es, P_3)$$

So, we have:

$$min_{\preceq_{D_{Player}}}(T) = (France, NULL, NULL, NULL, NULL, NULL, NULL, NULL)$$

$$max_{\preceq_{D_{Player}}}(T) = \{(NULL, NULL, NULL, NULL, NULL, NULL, NULL, P_2), (NULL, NULL, NULL, NULL, NULL, NULL, NULL, P_3)\}$$

Definition 3.16 (Generalized *min*/*max* operators on the cube lattice)—We generalize the *min* operator on the cube lattice as follows:

$$\begin{cases} \forall T \subseteq CL(r), min_{\preceq_s}(T) = \{t \in T \mid \nexists t' \in T : t' \preceq_s t\} \\ \forall d_i \in \mathcal{D}, min_{\preceq_s}(T[d_i]) = \{t[d_i] \in T[d_i] \mid \nexists t'[d_i] \in T[d_i] : t'[d_i] \preceq_s t[d_i]\} \end{cases}$$

Likewise, we generalize the *max* operator on the cube lattice as follows:

$$\begin{cases} \forall T \subseteq CL(r), max_{\preceq_s}(T) = \{t \in T \mid \nexists t' \in T : t \preceq_s t'\} \\ \forall d_i \in \mathcal{D}, max_{\preceq_s}(T[d_i]) = \{t[d_i] \in T[d_i] \mid \nexists t'[d_i] \in T[d_i] : t[d_i] \preceq_s t'[d_i]\} \end{cases}$$

Definition 3.17 (Dimensional Sum operator). Let's consider $d_i \in \mathcal{D}$, we define the Dimensional Sum $+_{d_i}$ of d_i as follows: $\forall x, y \in Dom(h_i)$, $x +_{d_i} y$ is the nearest (small) common ancestor to *x* and *y* in h_i. In other words:

$$\forall x \in Dom(h_i, f), \forall y \in Dom(h_i, g)$$
$$x +_{d_i} y = z \in Dom(h_i, e) \mid z \preceq_{d_i} x \text{ and } z \preceq_{d_i} y \mid \nexists e' \in S_h(d_i) : e' > e$$

Example 3.12 For the dimension D_{Player}, if $x = Paris$ and $y = Marseille$ then $x +_{D_{Player}} y = France$.

The Dimensional Sum operator can be overloaded to be applied to all tuples t_x and t_y of a dimension d_i, noted $t_x +_{d_i} t_y$, by being implemented as follows:

$$\forall x \in Dom(h_i, f), \forall y \in Dom(h_i, g),$$
$$\exists z \in Dom(h_i, e), e < f \text{ and } e < g, t_z = t_x +_{d_i} t_y = t_{x+_{d_i} y}$$

Example 3.13 For the dimension D_{Player}:

If $x = Paris$,

$$t_x = (France, IDF, Paris, NULL, NULL, NULL, NULL, NULL)$$

And $y = Marseille$,

$$t_y = (France, PACA, Marseille, NULL, NULL, NULL, NULL, NULL)$$

Then $z = France$,

$$t_z = t_x +_{d_i} t_y$$
$$= (France, NULL, NULL, NULL, NULL, NULL, NULL, NULL)$$

Definition 3.18 (Generalized Sum operator on the cube lattice). We generalize the Sum operator on the cube lattice as follows:

$$\begin{cases} \forall u, v \in CL(r), \{z = u + v\} \\ \forall d_i \in \mathcal{D}, z[d_i] = u[d_i] +_{d_i} v[d_i] \end{cases}$$

z is the sum of tuples *u* and *v*.

Definition 3.19 (Dimensional Product operator). Let's consider $d_i \in \mathcal{D}$, we define the Dimensional Product $\bullet_{d_i}$ of d_i as follows: $\forall x, y \in Dom(h_i)$, $x \bullet_{d_i} y$ is the set of nearest common descendants of x and y in h_i. In other words:

$$\forall f, g < Prof(h_i), \forall x \in Dom(h_i, f), \forall y \in Dom(h_i, g),$$
$$x \bullet_{d_i} y = \left\{ z \in Dom(h_i, e) \mid x \preceq_{d_i} z \text{ et } y \preceq_{d_i} z \mid \nexists e' \in S_h(d_i) : e' < e \right\}$$

If x and y have no common descendants in h_i, then:

$$x \bullet_{d_i} y = \{\varnothing\}$$

Example 3.14 For the dimension $\mathrm{D}_{\mathrm{Player}}$:

If $x = Paris$ and $y = 92.88.91.80$, $x \bullet_{\mathrm{D}_{\mathrm{Player}}} y = \{Windows\}$

If $x = Marseille$ and $y = 139.124.242.125$, $x \bullet_{\mathrm{D}_{\mathrm{Player}}} y = \{Linux, Mac\ OS\}$

If $x = Paris$ and $y = Marseille$, $x \bullet_{\mathrm{D}_{\mathrm{Player}}} y = \{\varnothing\}$

The Dimensional Product operator can be overloaded to be applied to all tuples t_x and t_y of a dimension d_i, noted $t_x \bullet_{d_i} t_y$, by being implemented as follows:

$$\forall f, g < Prof(h_i), \forall x \in Dom(h_i, f), \forall y \in Dom(h_i, g),$$
$$t_z = t_x \bullet_{d_i} t_y = \begin{cases} \left\{ t_{x \bullet_{d_i} y} \right\} \text{ if } \exists z \in Dom(h_i, e),\ f < e \text{ and } g < e \\ \left\{ (\varnothing, \cdots, \varnothing) \right\} \text{ otherwise.} \end{cases}$$

Example 3.15 For the dimension $\mathrm{D}_{\mathrm{Player}}$:

If $x = Marseille$,

$$t_x = (France, PACA, Marseille, NULL, NULL, NULL, NULL, NULL)$$

And $y = 139.124.242.125$,

$$t_y = (France, PACA, Marseille, 139.124.242.125, NULL, NULL, NULL, NULL)$$

Then $z = \{Linux, Mac\ OS\}$,

$$\begin{aligned} t_z &= t_x \bullet_{d_i} t_y \\ &= \{(France, PACA, Marseille, 139.124.242.125, Linux, NULL, NULL, NULL) \\ &\quad (France, PACA, Marseille, 139.124.242.125, Mac\ OS, NULL, NULL, NULL)\} \end{aligned}$$

Definition 3.20 (Generalized Product operator on the cube lattice). We generalize the Product operator on the cube lattice as follows:

$$\begin{cases} \forall u, v \in CL(r), \{z = u \bullet v\} \\ \forall d_i \in \mathcal{D}, \{z[d_i] = u[d_i] \bullet_{d_i} v[d_i]\} \end{cases}$$

z is the product of tuples u and v.

Definition 3.21 (Dimensional Semi-product operator). Let's consider $d_i \in \mathcal{D}$, we define the dimensional Semi-product $\odot_{d_i}$ of d_i as follows: $\forall x, y \in Dom(h_i)$,

$x \odot_{d_i} y$ is the set of nearest descendants of x and y in h_i. In other words:

$$\forall f < Prof(h_i), \forall x, y \in Dom(h_i, f),$$
$$x \odot_{d_i} y = \left\{ z \in Dom(h_i, e) \mid x \preceq_{d_i} z \text{ et } y \preceq_{d_i} z \mid \nexists e' \in S_h(d_i) : e' < e \right\}$$

If x and y do not have the same level, or if they do not have descendants, in h_i, then:

$$x \odot_{d_i} y = \{\varnothing\}$$

Example 3.16 For the dimension D_{Player}:

IF $x = Linux$ and $y = Max\ OS$, $x \odot_{D_{Player}} y = \{Opera, Firefox\}$

IF $x = Marseill$ and $y = Max\ OS$, $x \odot_{D_{Player}} y = \{\varnothing\}$

The Dimensional Semi-product operator can be overloaded to be applied to all tuples t_x and t_y of a dimension d_i, noted $t_x \odot_{d_i} t_y$, by being implemented as follows:

$$\forall f < Prof(h_i), \forall x, y \in Dom(h_i, f), \forall z \in Dom(h_i, e),$$
$$t_z = t_x \odot_{d_i} t_y = \left\{ t_{x \odot_{d_i} y} \mid e > f \mid \nexists e' \in S_h(d_i) : e' < e \right\}$$

$$\forall x, y \in Dom(h_i, f),$$
$$t_z = t_x \odot_{d_i} t_y = \begin{cases} \left\{ t_{x \odot_{d_i} y} \right\} \text{ if } \exists z \in Dom(h_i, e),\ f < e \\ \left\{ (\varnothing, \cdots, \varnothing) \right\} \text{ otherwise.} \end{cases}$$

Example 3.17 For the dimension D_{Player}:

If $x = Linux$,

$$t_x = (France, PACA, Marseille, 139.124.242.125, Linux, NULL, NULL, NULL)$$

And $y = Mac\ OS$,

$$t_y = (France, PACA, Marseille, 139.124.242.125, Mac\ OS, NULL, NULL, NULL)$$

Then $z = \{Opera, Firefox\}$,

$$\begin{aligned} t_z &= t_x \odot_{d_i} t_y \\ &= \{ (France, PACA, Marseille, 139.124.242.125, Linux, Opera, NULL, NULL) \\ &\quad (France, PACA, Marseille, 139.124.242.125, Mac\ OS, Firefox, NULL, NULL) \} \end{aligned}$$

Definition 3.22 (Generalized Semi-product operator on the cube lattice) We generalize the Semi-Product operator on the cube lattice as follows:

$$\begin{cases} \forall u, v \in CL(r), \{ z = u \odot v \} \\ \forall d_i \in \mathcal{D}, \{ z[d_i] = u[d_i] \odot_{d_i} v[d_i] \} \end{cases}$$

z is the product of tuples u and v.

3.3.5. Characterization of the Hierarchical Cube Lattice

By endowing the multidimensional space $Space(r)$ of r with orders of specialization and using the operators, notably Sum and Product, we propose an algebraic structure called Hierarchical Datacube Lattice, or Hierarchical Cube Lattice, or simply Cube Lattice, which sets a theoretical and general framework for multidimensional database mining. It is easily transposable from the standard datacube lattice. The following lemmas and propositions give the fundamental properties of the cube lattice, which are repeated in Theorem 6.

Lemma 1. The ordered set $CL(r)=\langle Space(r),\preceq_s\rangle$ is a complete lattice called cube lattice for which:

- $\forall T\subseteq CL(r), \wedge T=+_{t\in T}\, t$ où $\wedge$ symbolises the *infimum*.
- $\forall T\subseteq CL(r), \vee T=\bullet_{t\in T}\, t$ où $\vee$ symbolises the *supremum*.

Lemma 2. The lattice $CL(r)=\langle Space(r),\preceq_s\rangle$ is a co-atomic and atomic lattice.

Proposition 3 (Hierarchical lattice of parts of binary attributes). Let $\mathcal{L}(r)$ be the hierarchical lattice of parts of binary attributes of the binary relation, *i.e.* the lattice $\left\langle \mathcal{P}\left(\bigcup_{d_i\in\mathcal{D}} d_i\cdot a, \forall a\in Dom(d_i)\right),\subseteq\right\rangle$. Then there exists an order- embedding Φ :

$$CL(r)\to\mathcal{L}(r)$$

$$t\mapsto\begin{cases}\bigcup_{d_i\in\mathcal{D}} d_i\cdot a, \forall a\in Dom(d_i) \text{ if } t=(\varnothing,\cdots,\varnothing)\\ \{d_i\cdot t[d_i]\,|\,\forall d_i\in Attribute(t)\} \text{ otherwise.}\end{cases}$$

The rank of a tuple t, noted $rank(t)$, is the length of the smallest path (minimal number of arcs) in the cube lattice connecting it to the tuple $(ALL_i,\cdots,ALL_j)$. We thus have: $rank(t)=|\Phi(t)|$ if $t\neq(\varnothing,\cdots,\varnothing)$, $|\mathcal{D}|+1$ otherwise.

Proposition 4 (Hierarchical co-atom/atom). Hierarchical co-atoms (respectively hierarchical atoms) are the maximal tuples, namely the most specific tuples (respectively maximal tuples) of the lattice deprived of its universal majorant (respectively minorant). The hierarchical co-atoms (respectively hierarchical atoms) of the hierarchical cube lattice of a relation r are noted $\mathcal{CA}t(CL(r))$ (respectively $\mathcal{A}t(CL(r))$).

Lemma 5. The cube lattice $CL(r)$ is graduated. If $|\mathcal{D}|\leq 2$ then $CL(r)$ is not distributive.

Lemma 5 shows that the cube lattice is a graduated lattice. Therefore, we can apply level-wise algorithms on this search space.

Theorem 6 (Hierarchical datacube lattice). Let r be a data warehouse composed of hierarchical dimensions and measures ($\mathcal{D}\cup\mathcal{M}$). The ordered set $CL(r)=\langle Space(r),\preceq_s\rangle$ is a complete, atomic, co-atomic and gradual hierarchical lattice called a hierarchical datacube lattice in which:

- $\forall T\subseteq CL(r), \wedge T=+_{t\in T}t$
- $\forall T\subseteq CL(r), \vee T=\bullet_{t\in T}t$.

Example 3.18: Figure 5 shows the cube lattice of our example table of facts relation (*cf.* Table 4). In this diagram, the edges represent the links of generalization or specialization between tuples. The values of the coded attributes are as follows:

Location		Step		Association	
France	$=F$	$S_x\left(x\in\mathbb{N}^*\right)$	$=x$	$A_x\left(x\in\mathbb{N}^*\right)$	$=x$
IDF	$=I$	$S_{x-y}\left(x,y\in\mathbb{N}^*\right)$	$=x-y$	$A_{x-y}\left(x,y\in\mathbb{N}^*\right)$	$=x-y$
PACA	$=P$	ALL_{Turn}	$=*$	ALL_{Series}	$=*$
Marseille	$=M$				
ALL_{Player}	$=*$				

Disregarding $(\varnothing,\varnothing,\varnothing)$, hierarchical co-atoms are tuples conveying information at the most detailed level, *i.e.* that of the actual values of dimensions. In other words, the hierarchical co-atoms are the potential tuples of a relation. Thus, we have:

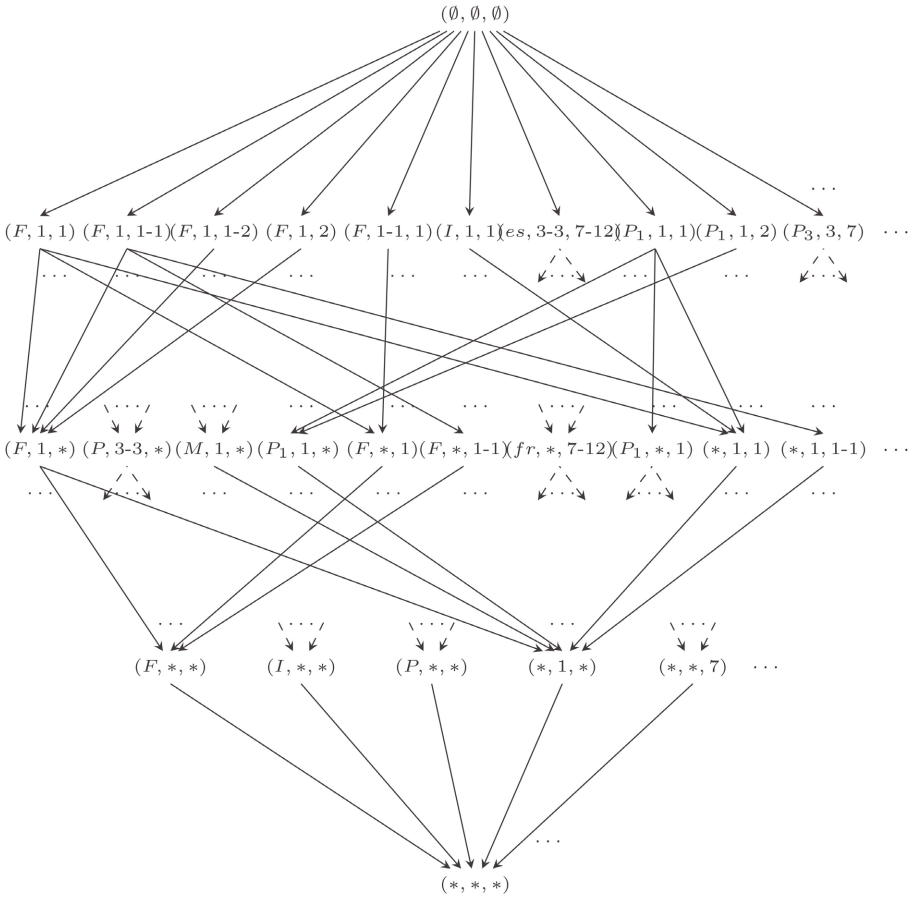

Figure 5. Hasse diagram of the hierarchical cube lattice of OM3.

$$\mathcal{CA}t\left(CL\left(\text{OM3}\right)\right)=\left\{\begin{array}{l}\left(France,S_1,A_1\right),\\ \left(France,S_1,A_{1-1}\right),\\ \left(France,S_1,A_{1-2}\right),\\ \left(France,S_1,A_2\right),\\ \cdots\\ \left(France,S_{1-1},A_1\right),\\ \cdots\\ \left(IDF,S_1,A_1\right),\\ \cdots\\ \left(es,S_{3-3},A_{7-12}\right),\\ \left(P_1,S_1,A_1\right),\\ \cdots\\ \left(P_1,S_1,A_2\right),\\ \cdots\\ \left(P_2,S_3,A_7\right),\\ \cdots\end{array}\right\}$$

The hierarchical atoms of the lattice offer the most synthetic information possible, with the exception of the tuple (ALL_{Player}, ALL_{Turn}, ALL_{Series}). Since we will only consider three dimensions here, all hierarchical atoms have two different synthetic ALL_i values. Thus we have:

$$\mathcal{A}t\left(CL\left(r\right)\right)=\left\{\begin{array}{l}\left(France,ALL_{\text{Turn}},ALL_{\text{Series}}\right),\\ \left(IDF,ALL_{\text{Turn}},ALL_{\text{Series}}\right),\\ \left(Paris,ALL_{\text{Turn}},ALL_{\text{Series}}\right),\\ \cdots\\ \left(ALL_{\text{Player}},S_1,ALL_{\text{Series}}\right),\\ \cdots\\ \left(ALL_{\text{Player}},ALL_{\text{Turn}},A_7\right),\\ \cdots\end{array}\right\}$$

Property 3.1 (Size of a hierarchical cube lattice). The height (number of levels) of the hierarchical cube lattice is $\left|\mathcal{D}\right|+1$. The number of elements for a level $i\left(i\in 1,\cdots,\left|\mathcal{D}\right|\right)$ is:

$$\sum_{\substack{d_i\subseteq\mathcal{D}\\ \left|d_i\right|=i}}\left(\prod_{A\in d_i}\left|Dom\left(h_i\right)\right|\right)\le\binom{\left|\mathcal{D}\right|}{i}\max_{A\in\mathcal{D}}\left(\left|Dom\left(h_i\right)\right|\right)^i.$$

The total number of elements in the hierarchical cube lattice is:

$$\sum_{i=1,\cdots,\left|\mathcal{D}\right|}\left(\sum_{\substack{d_i\subseteq\mathcal{D}\\ \left|d_i\right|=i}}\left(\prod_{A\in d_i}\left|Dom\left(h_i\right)\right|\right)\right)+2=\prod_{A\in\mathcal{D}}\left(\left|Dom\left(h_i\right)\right|+1\right)+1$$

The above property gives an analysis of the number of elements contained in a level of the hierarchical cube lattice and the total number of elements. This property is particularly important as it allows us to characterise the complexity

of algorithms using the hierarchical cube lattice as a search space.

4. Hierarchical Datacube

The concept of a datacube in a relational world ([9]) is immediately transposed into a hierarchical datacube in a data warehouse world, with the integration of hierarchical dimensions.

Thus, we call the result of a partitioning query, according to a set $X \subseteq \mathcal{D}$ of hierarchical dimensions, cuboid, noted $cuboid_X(r)$. Likewise, the datacube constituted by the set of all cuboids is noted $datacube(r)$. The cuboids can be ordered with respect to their level of detail by the partial order relation $\preceq$.

Example 4.1: The cuboid according to the hierarchical dimensions D_{Player}, D_{Turn} is less aggregated (more detailed) than the cuboid according to D_{Player} or that according to D_{Turn}. In other words:

$$Cuboid_{D_{Player}}(r) \preceq Cuboid_{D_{Player},D_{Turn}}(r)$$

$$Cuboid_{D_{Turn}}(r) \preceq Cuboid_{D_{Player},D_{Turn}}(r)$$

In a standard way, the set of cuboids endowed with this order forms a lattice. The cuboids of this lattice are grouped by level according to their number of hierarchical dimensions. These levels are numbered starting from the bottom of the lattice (cuboid bearing no dimension d_i) and going up towards the top (cuboid according to all possible criteria called "base cuboid"). Let's consider two cuboids $cuboid_{\mathcal{U}}(r)$ according to the hierarchical dimension subset $\mathcal{U}$ and $cuboid_{\mathcal{V}}(r)$ according to $\mathcal{V}$, if $\mathcal{U} \subset \mathcal{V}$ then $cuboid_{\mathcal{U}}(r) \preceq cuboid_{\mathcal{V}}(r)$, the cuboids are said to have a family relationship, $cuboid_{\mathcal{V}}(r)$ is called the "ancestor" of $cuboid_{\mathcal{U}}(r)$ and $cuboid_{\mathcal{U}}(r)$ is the "offspring" of $cuboid_{\mathcal{V}}(r)$.

Figure 6 gives the representation of the cuboid lattice of the data warehouse OM3 according to the hierarchical dimensions D_{Player} (IdP), D_{Turn} (IdT) and D_{Series} (IdS).

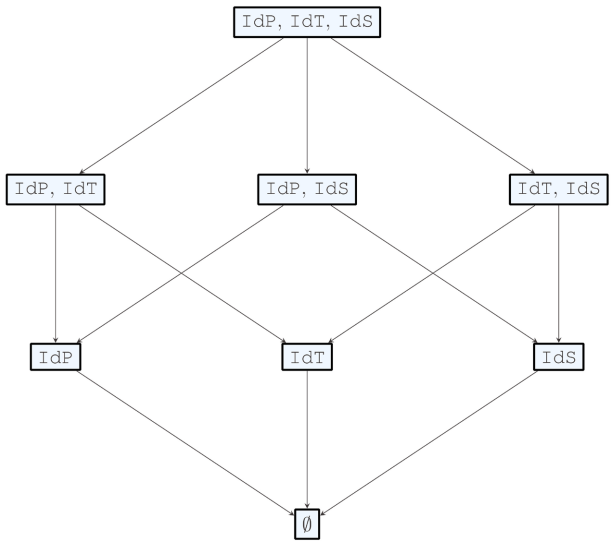

Figure 6. Cuboids of OM3 according to D_{Player} (IdP), D_{Turn} (IdT) and D_{Series} (IdS).

By taking the example of the OM3 and its table of facts (*cf.* Table 4), we can express in SQL the hierarchical datacube necessary for an analysis according to the measurements Time and Score with respect to hierarchical dimensions IdP, IdT and IdS with the following query:

```
01 |  SELECT IdP, IdT, IdS, SUM(Time), MAX(Score)
02 |  FROM OM3
03 |  GROUP BY CUBE IdP, IdT, IdS
```

This query will calculate the $2^3 = 8$ partitionings (where $\varnothing$ represents the partitioning along no dimension d_i, *i.e.* aggregating the entire data warehouse into a single tuple:

- "IdP, IdT, IdS";
- "IdP, IdT";
- "IdP, IdS";
- "IdT, IdS";
- "IdP";
- "IdT";
- "IdS";
- "$\varnothing$".

As with the traditional datacube, the naive way to compute this type of query is to rewrite it as a collection of eight aggregative queries and run them separately. However, each of its subqueries has its own pattern. To make all these cuboids uni-compatible, we use the value ALL_i (*cf.* Definition 3.5): this value has a particular semantics, it is a generalization of all the values of the domain of an attribute of a dimension d_i. Thus, the cuboids all share the same schema ($\mathcal{D} \cup \mathcal{M}$) and can be grouped together in a single relation: the hierarchical datacube. Each multidimensional tuple consists of a set of values for its hierarchical dimensions and numerical values for their measures. The value of the measure is computed by aggregating the set of tuples of the original relation sharing the same values of the selected hierarchical dimensions.

Example 4.2 - The multidimensional tuple $t = (8,1,ALL)$ is obtained by aggregating by IdP and IdT the tuples $t_1 = (8,1,1)$ and $t_2 = (8,1,4)$ of the example data warehouse (*cf.* Table 4). The measures of t are obtained by applying the measure functions on the set of tuples needed for the aggregation.

In our case, we have:

- $f_{\text{Time}}(t,r) = t_1(\text{Time}) + t_2(\text{Time}) = SUM(6.32,18.9) = 25.22$
- …
- $f_{\text{Score}}(t,r) = t_1(\text{Score}) + t_2(\text{Score}) = MAX(700,2300) = 2300$
- …

With the example data warehouse (*cf.* Table 4), the query to naively compute the hierarchical datacube according to the hierarchical dimensions D_{Player} (IdP), D_{Turn} (IdT) and D_{Series} (IdS) and the measures Time, Duration, Number, Score and Shape is as follows:

```
SELECT IdP, IdT, IdS, SUM(Time), MAX(Score)
FROM OM3
GROUP BY IdP, IdT, IdS
UNION
SELECT IdP, IdT, ALL, SUM(Time), MAX(Score)
FROM OM3
GROUP BY IdP, IdT
UNION
SELECT IdP, ALL, IdS, SUM(Time), MAX(Score)
FROM OM3
GROUP BY IdP, IdS
UNION
SELECT ALL, IdT, IdS, SUM(Time), MAX(Score)
FROM OM3
GROUP BY IdT, IdS
UNION
SELECT IdP, ALL, ALL, SUM(Time), MAX(Score)
FROM OM3
GROUP BY IdP
UNION
SELECT ALL, IdT, ALL, SUM(Time), MAX(Score)
FROM OM3
GROUP BY IdT
UNION
SELECT ALL, ALL, IdS, SUM(Time), MAX(Score)
FROM OM3
GROUP BY IdS
UNION
SELECT ALL, ALL, ALL, SUM(Time), MAX(Score)
FROM OM3
```

The result of this query is given by the **Table 6**. For clarity, different cuboids are separated by a horizontal line, and special values are abbreviated: ALL_{Player} as ALL_{P}, ALL_{Turn} as ALL_{T} and ALL_{Series} as ALL_{S}.

5. Closed Hierarchical Datacube

The combinatorial explosion of results during datacube computations is a well known phenomenon ([12]), which is amplified with the computation of hierarchical datacubes. The closed hierarchical cube approach we propose aims at representing the hierarchical datacube without loss of information but with a consequent decrease of the necessary storage space. To do this, we eliminate possible redundancies by keeping only one representative for a set of tuples from the same data of the original relation. It is easily transposable from the standard closed datacube.

5.1. Experimental Results for Closed Datacube

Our objective now is to compare, through various experiments, the sizes of the datacube and the closed cube. In this sub-section, we report a summary of our results. All experiments are conducted on an Intel Pentium G2120 3.10 GHz with 3.6 GB main memory and running on Turnkey LAMP Stack (based on Debian GNU/Linux). We use the algorithm Close [13] (for which we have the sources) in order to perform experimental comparisons between the representations. We use real data sets to evaluate the effectiveness of our approach.

Table 6. Relational representation of the hierarchical datacube of OM3.

IdP	IdT	IdS	Time	Duration	Number	Score	Shape
P_1	S_1	A_1	6.32	2.85	3.5	700	0.5
P_1	S_1	A_2	18.9	1.95	3.83	2300	0.5
P_2	S_2	A_3	26.39	1.7	3.43	2400	0.71
P_2	S_2	A_4	4.1	2.07	3	300	0.5
P_2	S_2	A_5	7.38	3.68	3	600	0.5
P_3	S_3	A_6	2.14	2.15	3	300	1
P_3	S_3	A_7	56.04	2.25	3.17	3800	0.5
P_1	S_1	ALL_S	25.22	2.04	3.8	2300	0.5
P_2	S_2	ALL_S	37.87	1.79	3.4	2400	0.7
P_3	S_3	ALL_S	58.18	2.25	3.17	3800	0.5
P_1	ALL_T	A_1	6.32	2.85	3.5	700	0.5
P_1	ALL_T	A_2	18.9	1.95	3.83	2300	0.5
P_2	ALL_T	A_3	26.39	1.7	3.43	2400	0.71
P_2	ALL_T	A_4	4.1	2.07	3	300	0.5
P_2	ALL_T	A_5	7.38	3.68	3	600	0.5
P_3	ALL_T	A_6	2.14	2.15	3	300	1
P_3	ALL_T	A_7	56.04	2.25	3.17	3800	0.5
ALL_P	S_1	A_1	6.32	2.85	3.5	700	0.5
ALL_P	S_1	A_2	18.9	1.95	3.83	2300	0.5
ALL_P	S_2	A_3	26.39	1.7	3.43	2400	0.71
ALL_P	S_2	A_4	4.1	2.07	3	300	0.5
ALL_P	S_2	A_5	7.38	3.68	3	600	0.5
ALL_P	S_3	A_6	2.14	2.15	3	300	1
ALL_P	S_3	A_7	56.04	2.25	3.17	3800	0.5
P_1	ALL_T	ALL_S	25.22	2.04	3.8	2300	0.5
P_2	ALL_T	ALL_S	37.87	1.79	3.4	2400	0.7
P_3	ALL_T	ALL_S	58.18	2.25	3.17	3800	0.5
ALL_P	S_1	ALL_S	25.22	2.04	3.8	2300	0.5
ALL_P	S_2	ALL_S	37.87	1.79	3.4	2400	0.7
ALL_P	S_3	ALL_S	58.18	2.25	3.17	3800	0.5
ALL_P	ALL_T	A_1	6.32	2.85	3.5	700	0.5
ALL_P	ALL_T	A_2	18.9	1.95	3.83	2300	0.5
ALL_P	ALL_T	A_3	26.39	1.7	3.43	2400	0.71
ALL_P	ALL_T	A_4	4.1	2.07	3	300	0.5
ALL_P	ALL_T	A_5	7.38	3.68	3	600	0.5
ALL_P	ALL_T	A_6	2.14	2.15	3	300	1
ALL_P	ALL_T	A_7	56.04	2.25	3.17	3800	0.5
ALL_P	ALL_T	ALL_S	121.27	2.11	3.32	3800	0.55

We use the real dataset SEP85L containing weather conditions at various weather stations from December 1981 through November 1991. This weather dataset has been frequently used in calibrating various cube algorithms [14]. Mushroom is a dataset widely known in frequent pattern mining. It provides various characteristics of mushrooms. Death is a dataset gathering information about patients' decease with the date and cause. TombNecropolis and TombObjects are issued from archaeological excavation. They encompass a list of necropolises, their tombs and other properties like the country, the funeral rite, the objects discovered in the tombs and their description. Finally, Joint_Objects_Tombs results from the natural join between TombObjects and TombNecropolis according to the identifiers of necropolises and tombs.

Table 7 gives the datasets used for experiments. The columns Attributes and Tuples stand for the number of attributes and tuples respectively. In the last column, the size in bytes of the dataset is reported (each dimension or attribute is encoded as an integer requiring 4 bytes for any value).

Table 8 illustrates the size of the studied representations for the various datasets.

These five datasets are only encompassing strongly correlated data. Thus we are in the most difficult cases. In this context, the closed cube reduces the size of the datacube from 8 to more than 300 times.

By using the SEP85L dataset, we have generated 9 datasets having from 2 to 10 dimensions by projecting the weather dataset on the first k dimensions ($2 \geq k \geq 10$). Table 9 presents the number of resulting patterns for the approaches.

Table 7. Experimental datasets.

Tables	Attributes	Tuples	Size
SEP85L	20	507,684	56,520
Mushroom	23	8124	747,408
Death	5	389	7780
TombNecropolis	7	1846	51,688
TombObjects	12	8278	397,344
Joint_Objects_Tombs	17	7643	519,724

Table 8. Size of the data cubes (in bytes).

	Datacube	Closed cube
Death	220,152	24,984
TombNecropolis	3,639,072	189,728
Joint_Objects_Tombs (1%)	58,848,264	4,485,168
Mushroom (5%)	436,823,808	1,233,984
TombObjects	903,611,124	8,032,648

Table 9. Experimental results for SEP85L.

Dimensions	Number of tuples	
	Datacube	Closed datacube
2	12,234	7928
3	21,114	13,883
4	27,514	18,874
5	38,793	24,478
6	62,248	34,898
7	109,314	52,344
8	199,639	81,640
9	371,791	136,275
10	722,133	172,452

Whatever the number of dimensions is, the closed cube is the smallest representation. It is always significantly reduced when compared to the datacube itself.

5.2. Formal Framework of Closed Hierarchical Datacube

The following definitions and corollary give the fundamental properties of the closed hierarchical datacube, which are repeated in Theorem 7.

Definition 5.1 (Hierarchical cube closure operator)—The hierarchical cube closure operator associates to any tuple t of the hierarchical cube lattice a single tuple called the hierarchical closure of t, or simply the closure of t. It is obtained by considering the set of tuples more specific than t and by determining the most general tuple of this set using the Sum operator. This operator is noted $\mathbb{C}$ and defined as follows:

$$\mathbb{C}: CL(r) \to CL(r)$$
$$t \mapsto \begin{cases} +t' \mid t \preceq_s t' \text{ and } t' \in r \\ (\varnothing, \cdots, \varnothing) \text{ otherwise.} \end{cases}$$

Example 5.1 Considering the multidimensional space of the data warehouse OM3 (*cf.* Table 5) and its table of facts (*cf.* Table 4), we have $\mathbb{C}\left(\left(P_1, ALL_T, ALL_S\right)\right) = \left(P_1, S_1, A_1\right) + \left(P_1, S_1, A_2\right) = \left(P_1, S_1, ALL_S\right)$ and $\mathbb{C}\left(\left(ALL_P, ALL_T, A_7\right)\right) = \left(P_3, S_3, A_7\right)$.

Corollary 1 $\mathbb{C}$ is a closure operator of $CL(r)$ on r. $\mathbb{C}$ satisfies the following properties:

- $t \preceq_g t' \Rightarrow \mathbb{C}(t,r) \preceq_g \mathbb{C}(t',r)$ (isotonicity);
- $t \preceq_g \mathbb{C}(t,r)$ (extensivity);
- $\mathbb{C}(t,r) = \mathbb{C}\left(\mathbb{C}(t,r),r\right)$ (idempotency).

Definition 5.2 (Hierarchical cube closure system) Let's consider

$\mathbb{C}(r)=\{t\in CL(r)\,|\,\mathbb{C}(t,r)=t\}$. $\mathbb{C}(r)$ is a closure system on r and the associated closure operator is $\mathbb{C}$. Any tuple belonging to $\mathbb{C}(r)$ is a hierarchical closed tuple or a hierarchical cube closure.

Example 5.2 Considering the multidimensional space of the data warehouse OM3 (*cf.* Table 5) and its table of facts (*cf.* Table 4), we have:

$$\mathbb{C}(r)=\left\{\begin{array}{l}(P_1,S_1,A_1),\\(P_1,S_1,A_2),\\(P_2,S_2,A_3),\\(P_2,S_2,A_4),\\(P_2,S_2,A_5),\\(P_3,S_3,A_6),\\(P_3,S_3,A_7),\\(P_1,S_1,ALL_S),\\(P_2,S_2,ALL_S),\\(P_3,S_3,ALL_S),\\(\varnothing,\cdots,\varnothing)\end{array}\right\}$$

Theorem 7. The partially ordered set $CCL(r)=\langle\mathbb{C}(r),\preceq_g\rangle$ is a complete and co-atomic lattice called a closed cube lattice such that:

- $\forall T\subseteq CCL(r),\wedge T=+_{t\in T}t$
- $\forall T\subseteq CCL(r),\vee T=\mathbb{C}(\bullet_{t\in T}t,r)$

All tuples with the same closure generalize the same tuples of the original relation. As they generalize the same original tuples, they share the same aggregated value of the measure (property of GROUP BY). For each tuple of the cube lattice, it is sufficient to calculate its closure to find its measure. Thus the closed hierarchical cube is a cover of the hierarchical datacube.

Example 5.3 - Figure 7 shows the closed hierarchical cube lattice of OM3.

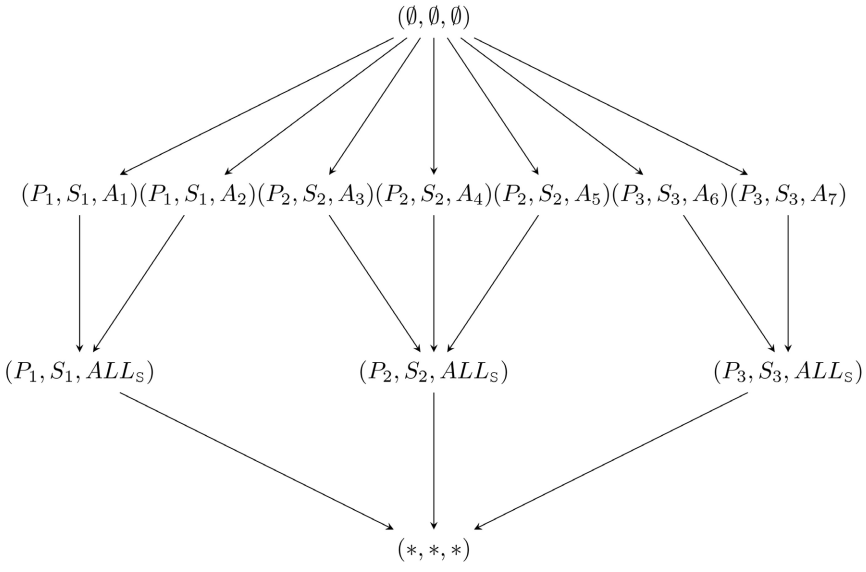

Figure 7. Hasse diagram of the closed hierarchical cube lattice of OM3.

5.3. Possible Limitations of Closed Hierarchical Datacube

Despite offering advantages in term of storage efficiency and query acceleration, closed hierarchical datacube have some possible limitations and limitations to account for, like loss in details, analysis flexibility, hierarchy management and adaptability to non-hierarchical dimensions. One of the possible inconveniences of a closed hierarchical datacube is that data consolidation may imply a loss of detail or an over-aggregation of data. By aggregating data at higher hierarchical levels, specific details may be lost at lower hierarchical levels, which can limit the ability to perform fine-grained data analysis at the lowest levels of granularity. That can be an inconvenience if details analysis is needed for specific decisions or perspectives. A closed datacube can also be less flexible for *ad hoc* analysis, because data are consolidated at specific hierarchical levels. If new analysis questions need different granularity levels of different data groupings, this may require to compute new closed datacubes or to recompute the current ones, which will require more time and resources. Also, closed datacubes are typically precomputed to aggregate data at higher hierarchical levels, which can speed up analysis queries. However, this also means that data are pre-computed and stored in advance, which can incur a cost in terms of storage space for updates and modifications of underlying data. For example, hierarchies management can be complex in a closed datacube, because there can be more consolidation and aggregation levels to consider. The definition and management of hierarchical relations between dimensions can require a special attention to ensure that consolidations and aggregations are correctly aligned with the needs of the analysis. Finally, closed datacubes are made to work with hierarchical dimensions where there is a parent-child relation between granularity levels. However, they might not be as suitable for non-hierarchical dimensions, where data don't follow a clear hierarchical structure. In these cases, closed datacubes may not be as efficient to aggregate and consolidate data, which can lead to a loss in performance and accuracy in analysis results.

6. Conclusions

As part of multidimensional data warehouses (or datamart), after presenting an application case on a video game, having given a definition to the hierarchical dimensions and presented the different types of hierarchies ([1]), we have defined a theoretical framework and a formal definition of the attributes of hierarchical dimensions and their associated structure and hierarchy.

Inspired by the classical theory, and transposing it in a relatively direct way, we have defined, in a hierarchical context, the multidimensional space, specialisation orders ([10] [11]), functions and operators, in particular the Sum operator and the Product operator, and for each, defined at the dimensional level, overloaded and generalized on the cube lattice.

We then characterized the hierarchical datacube lattice and gave a calculation of its size, showing the complexity of algorithms using the hierarchical cube lat-

tice as a search space.

We then proposed the hierarchical datacube, which shows the same appeal as the classic datacube ([9]), major concept for data warehouse management.

As with the original datacube, the closed hierarchical datacube is its most concise representation. It exploits the generally large redundancies in the data. This is a good way to reduce the size of a datacube. By removing them, it allows the necessary size to be reduced substantially without losing useful data. When the data are very correlated, the representation by a closed hierarchical cube shows all its interest because the reduction brought is quite significant, and nevertheless makes it possible to find even more information than with a traditional datacube. These strong correlations appear in real data sets and increase when the dimensional data is very scattered over very large domains. Such characteristics are typical of data warehouses ([15] [16]), and in particular in the field of video games where hierarchical dimension attributes are often legion.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Inmon, W.H. (1996) Building the Data Warehouse. 2nd Edition, John Wiley and Sons, Inc., Hoboken.

[2] Codd, E.F. (1990) The Relational Model for Database Management, Version 2. Addison-Wesley, Boston.

[3] Chaudhuri, S. and Dayal, U. (1997) An Overview of Data Warehousing and OLAP Technology. *SIGMOD Record*, **26**, 65-74. https://doi.org/10.1145/248603.248616

[4] Chaudhuri, S., Dayal, U. and Narasayya, V.R. (2011) An Overview of Business Intelligence Technology. *Communications of the ACM*, **54**, 88-98. https://doi.org/10.1145/1978542.1978562

[5] Hurtado, C.A. and Mendelzon, A.O. (2002) OLAP Dimension Constraints. *Proceedings of the* 21*st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, Madison, June 2002, 169-179. https://doi.org/10.1145/543613.543636

[6] Kuijpers, B. and Vaisman, A. (2016) A Formal Algebra for OLAP.

[7] Malinowski, E. and Zimányi, E. (2004) OLAP Hierarchies: A Conceptual Perspective. *Advanced Information Systems Engineering*, 16*th International Conference*, Riga, 7-11 June 2004, 477-491.

[8] Favre, C., *et al.* (2013) Les entrepots de donnees pour les nuls... ou pas!

[9] Gray, J., *et al.* (1997) Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub Totals. *Data Mining and Knowledge Discovery*, **1**, 29-53. https://doi.org/10.1023/A:1009726021843

[10] Mitchell, T.M. (1982) Generalization as Search. *Artificial Intelligence*, **18**, 203-226. https://doi.org/10.1016/0004-3702(82)90040-6

[11] Mitchell, T.M. (1997) Machine Learning. McGraw-Hill, New York.

[12] Han, J.W., Kamber, M. and Pei, J. (2011) Data Mining: Concepts and Techniques.

3rd Edition, Morgan Kaufmann, Burlington. http://hanj.cs.illinois.edu/bk3

[13] Pasquier, N., *et al.* (1999) Efficient Mining of Association Rules Using Closed Itemset Lattices. *Information Systems*, **24**, 25-46. https://doi.org/10.1016/S0306-4379(99)00003-4

[14] Wang, W., *et al.* (2002) Condensed Cube: An Efficient Approach to Reducing Data Cube Size. *Proceedings of the* 18*th International Conference on Data Engineering*, San Jose, 26 February-1 March 2002, 155-165. https://doi.org/10.1109/ICDE.2002.994705

[15] Ross, K.A. and Srivastava, D. (1997) Fast Computation of Sparse Datacubes. 23*rd International Conference on Very Large Databases*, *VLDB* 1997, Athens, 26-29 August 1997, 116-125.

[16] Beyer, K.S. and Ramakrishnan, R. (1999) Bottom-Up Computation of Sparse and Iceberg CUBEs. In: Delis, A., Faloutsos, C. and Ghandeharizadeh, S., Eds., *SIGMOD Conference*, ACM Press, New York, 359-370. https://doi.org/10.1145/304182.304214