Cube query answering via the results of previous cube queries

Panos Vassiliadis
Univ. Ioannina
Ioannina, Greece
pvassil@cs.uoi.gr

ABSTRACT

In this paper, we come with a model for multidimensional spaces with hierarchically structured dimensions in several layers of abstractions, data cubes and cube queries. The model serves as the basis to offer the main contribution of this paper which includes a theorem and an algorithm for being able to decide and facilitate the computation of the contents of a new cube query from a previous one, defined at a different level of abstraction.

1 INTRODUCTION

Multidimensional spaces with hierarchically structured dimensions over several levels of abstraction, along with data cubes, (i.e., structured collections of data points at the same level of detail in the context of such spaces), provide a paradigm for data management whose simplicity is hard to match.

The problem that this paper addresses can be summarized as the principled answering of the question: *how can we compute the contents of a new cube query, by reusing the existing contents of the result of a previous cube query?* Traditionally, related work has handled the problem of query containment and view usability for the relational case (see [14] for the general problem of answering queries using views, a survey of aggregate query containment in [4] and two lemmas [5] and [21]); also works on *Query Containment*([1], [18], [7], [8]) [9]); *View usability*([16], [10], [14]); *Query rewriting*([17], [2], [13], [3], [9], [12], [11], [6], [7])). However, the existence of hierarchically structured dimensions in the case of multidimensional spaces with different possible levels of aggregations as a context for the determination of cube usability, has not been extensively dealt with by the database community (see [23], [20] for two early attempts). *We attempt to fill this gap by providing a comprehensive rigorous basis and the respective theorems and algorithms for being able to solve the problem of cube usability for a very powerful class of queries.*

Contribution. In this paper, we start in Section 2, with a brief presentation of the core components of a model for multidimensional hierarchical spaces, cubes and cube queries. Based on the intrinsic property of the model that all query semantics are defined with respect to the most detailed level of aggregation in the hierarchical space, and in contrast to all previous models of multidimensional hierarchical spaces, in Section 3, we accompany the proposed model with definitions of equivalent expressions at different levels of granularity. We introduce the necessary terminology and notation, too, to solidify these concepts in the vocabulary of multidimensional modeling. Specifically, we introduce (a) proxies, i.e., equivalent expressions at different levels of abstraction, (b) signatures, i.e., sets of coordinates specifying a "border" in the multidimensional space that specifies a sub-space pertaining to a model's construct, and, (c) areas, i.e., set of cells enclosed within a signature.

Copyright © 2023 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

In Section 5, we address the *usability* problem of computing a new cube query c^n from the cells of a previous one, c^b, defined at a different level of abstraction; we introduce the respective test as well as a rewriting algorithm. As a pre-requisite to address the problem, which comes with the complexity of having to deal with grouper dimensions where selections have also been posed, in Section 4 we introduce the notion of *rollability* which refers to the property of the combination of a filter and a grouper level at the same dimension to produce result coordinates that are fully covering the respective subspace at the most detailed level. Section 6 provides issues for future work.

We encourage the reader to refer to the long version of the paper [22] that comes with (1) a comprehensive model for hierarchical multidimensional spaces and query expressions in them (including all the typical OLAP operations), (2) more explanations, rigorous definitions and proofs for the current paper, and, (3) a principled set of tests and algorithms for the problem of cube query containment at various level of detail (specifically: foundational, same-level, and, different level containment), query intersection (at various levels of detail), and query distance.

2 FORMALIZING DATA, DIMENSION HIERARCHIES CUBES AND CUBE QUERIES

In this Section, we give the background of our modeling concerning multidimensional databases, hierarchies and queries (see [19] for a survey of models). We assume data in a multidimensional space, where *dimensions* provide a *context* for facts [15]. Each dimension comes with a *hierarchy of levels*. Each dimension (e.g., $TaxDate$) is a lattice of levels (e.g., Day, $Week$, $Month$, $Year$). Each level comes with a domain of values; values in different levels are mapped via an *anc*() function for higher levels and and *desc* mapping for lower levels (e.g., $anc_{City}^{Country}(Athens) = Greece$). Each dimension includes a single most detailed level, and, a single top level ALL with a single-valued domain $\{all\}$.

Facts are structured in *cubes*. A cube is defined over the Cartesian Product of several levels from discrete dimensions along with a number of *measures* to hold the measurable aspects of its facts. Each *cell* is a point in the multidimensional space of the cube's dimensions hosting a set of measures. A *detailed cube* is a cube having all its dimensions fixed at the lowest possible level.

A *conjunctive selection condition* is a conjunction of atoms, each of the form $D.L \in \{v_1, \ldots, v_n\}$, where the values v_i belong to the domain of L.

The user can submit *cube queries* to the system. A cube query specifies (a) the detailed data set over which it is imposed, (b) the selection condition that isolates the records that qualify for further processing, (c) the aggregator levels, that determine the level of coarseness for the result, and (d) an aggregation over the measures of the underlying cube that accompanies the aggregator levels in the final result. More formally, a *cube query*, is an expression of the form:

CEUR-WS.org/Vol-3369/short1.pdf

$$q = \left\langle \ \mathbf{DS}^0,\ \phi,\ [L_1,\ldots,L_n,M_1,\ldots,M_m],\ [agg_1(M_1^0),\ldots,agg_m(M_m^0)]\ \right\rangle$$

where

(1) $\mathbf{DS}^0$ is a detailed data set over the schema S =[L_1^0, ..., L_n^0, M_1^0, ... ,M_k^0], $m \leq k$.
(2) ϕ is a multidimensional conjunctive selection condition, with a single atom per dimension (for a non-filtered dimension D, atom $D.ALL \in \{all\}$ is equivalent to $true$)
(3) $L_1,\ldots,L_n$ are *grouper* levels such that $L_i^0 \preceq L_i$, $1 \leq i \leq n$,
(4) $M_1,\ldots,M_m, m \leq k$, are aggregated measures (without loss of generality we assume that aggregation takes place over the first m measures – easily achievable by rearranging the order of the measures in the schema),
(5) $agg_1,\ldots,agg_m$ are aggregate functions from the set $\{sum, min, max, count\}$.

Example. Assume a tax office has a cube on the income tax collected and the effort invested to collect it on its allocated citizens. Due to anonymization, the tax office analyst is presented with a detailed cube without the identity of the citizens and has some (pre-aggregated) information along the following dimensions: $Date$, $WorkClass$, and $Education$, and two measures $TaxPaid$ by the citizens in thousands of Euros and $HoursSpent$. Each dimension is accompanied by hierarchies of dimension levels. $Date$ is organized in $Months$, $Quarters$, $Years$ and ALL. $Education$ has 5 levels, and $Workclass$ 4 levels. A detailed dataset $\mathbf{DS}^0$ is defined over these dimensions with a schema $\mathbf{DS}^0[\underline{D.L_0, W.L_0, E.L_0}, TaxPaid, HoursSpent]$.

A query that can be posed to the abovementioned detailed data set can be:

$$q = \left\langle \mathbf{DS}^0,\ \phi,\ [Month, W.L_1, E.ALL, sumTaxPaid],\ [sum(TaxPaid)]\ \right\rangle$$

with ϕ expressed as $\phi = Year \in \{2019, 2020\} \wedge W.L_2 \in \{with-pay\}$ and actually implying an expression with a single atom per dimension in the form: $\phi = Year \in \{2019, 2020\} \wedge W.L_2 \in \{with-pay\} \wedge Education.ALL \in \{all\}$

3 EQUIVALENT EXPRESSIONS FOR REFERRING TO SUBSETS OF THE MULTIDIMENSIONAL SPACE

In this Section, we deal with two fundamental characteristics of the multidimensional space: (a) the fact that the same data can be viewed from different levels of detail, and (b) the fact that each query in the multidimensional space applies a border of values of the dimensions, thus "framing" a subset of the space. Fig. 1 provides a summary of notation as well as a short description for each of the important concepts involved in our discourse.

A proxy is an equivalent expression at a different level of detail that by construction covers exactly the same subset of the multidimensional space, albeit at different level of coarseness. For example, given a value, its proxy at a lower level of detail is the set of its descendants at this level. The detailed proxy of an aggregate cell is the set of detailed cells who generate it. As another example, the detailed proxy of a query expression is an expression whose schema is at the most detailed level for each of the dimensions participating in the schema of the query, and whose selection condition is equivalent to the one of the query, but at the most detailed level. Moreover, apart from 'the most detailed level', proxies are definable at arbitrary levels of coarseness. Observe also that proxies are of the same type as their "arguments": the proxy of a cell is a set of cells, the proxy of an expression is an expression, etc.

The signature of a construct is a set of coordinates that characterize the subset of the multidimensional space "framed" by the construct. For example, the signature of a cell are its coordinates at the level of coarseness that the cell is defined, whereas the signature of its detailed proxy are the coordinates produced by the Cartesian product of the descendant values of these coordinates at the most detailed level. Similarly, the signature of a selection condition is the set of coordinates of the multidimensional space for which the selection condition evaluates to true.

Areas are sets of cells within the bounds of a signature. For example, the detailed area of a cell is the set of its detailed proxies. Similarly, for a given query q, the expression $q.cells$ refers to the cells belonging to the result of the query and $q^0.cells$ is the detailed area of the query, referring to the cells of the most detail level that produce the query result.

The grouper domains of atoms and formulas practically transform the "accepted" values by the filter to the grouper levels. Assume a selection atom for a dimension level L^σ and a grouper level for the same dimension, say L^γ: we can compute a set of high-level values at level L^γ produced by (a) applying the filter to the respective dimension level L^σ, and, (b) grouping the surviving values at the target grouper level L^γ. This can also be generalized to a selection condition.

4 PERFECT ROLLABILITY: HOW DO SELECTIONS AND GROUPERS RELATE?

In this Section, we discuss a subtle, but most important aspect of computing a new cube query from the pre-existing results of a previous query: the combination of the selection condition and the grouping levels.

Definition 4.1. Given a query q with a schema comprising a set of levels $[D_1.L_1,\ldots,D_n.L_n]$, over the respective dimensions:

- A dimension D is a *non-grouper*, when it's respective schema level is (rolled-up to) the level ALL.
- A dimension D is a *grouper*, when its respective level in the schema is not rolled-up to ALL.

By extension of the terminology, we will also refer to the respective levels as groupers and non-groupers, too.

How can two queries, aggregated at the same level of coarseness, be incompatible for usability? The first possible reason can be *different filters in non-grouper levels*: e.g., two queries group by $Geography.ALL$, but the old one applies the filter $Country = Japan$ and the new has the filter $Country = China$ (thus, the cells of the result of the two cubes will have the same coordinates, but the measure values will be different, due to the different filters in the non-groupers). The same happens even if the relationship of the two filters is a superset (e.g., the old query has $Country \in \{China, Japan\}$. A second reason concerns *problematic partial filters in groupers.* Assume that an old query selects months in [$January$ 2020 .. $November$ 2020] and the new query selecting Day in [1/1/2020 .. 15/11/2020]. The problem here is in November: both queries will roll up at the level of month, and thus will report the month November 2020, but the new query is filtering a subset of this month, and thus the aggregate measures will be different. Practically, we need to have identical selections for non-grouper levels, and "rollable" selection subsumption with respect to the grouping levels, for grouper levels.

	Signature: tuple of dimension values		*Proxy(x): of the same type as x*	
	Coord. signature or coordinates	*Detailed Signature*	*Descendant Proxy*	*Detailed Proxy*
value v			$v^{@L}$ = set of values at desc. level	v^0: set of values at zero level
set of coordinates X			$X^{@\mathbf{L}}$: set of coord. at **L**	X^0: set of coordinates at zero level
atom α	α^+: set of dim. values qualifying the atom, at the level of α	α^{0^+}: set of dim. values qualifying the atom, at the zero level	$\alpha^{@\mathbf{L}}$: equiv. expression at desc. levels	α^0: equiv. expression at zero level
condition ϕ	ϕ^+: coordinates produced by the Cart. Prod. of the α_i^+	ϕ^{0^+}: coord. produced by the Cart. Prod. of the $\alpha_i^{0^+}$	$\phi^{@\mathbf{L}}$: equiv. expression at desc. levels	ϕ^0 equiv. expression at zero level
cell c	c^+: tuple of cell's dim. values	c^{0^+}: set of coord. of detailed proxy	$c^{@\mathbf{L}}$: desc. area = set of cells at lower level	c^0: detailed area = set of cells at zero level
query expression q	q^+: set of coordinates of cube cells	q^{0^+}: set of coord. of detailed proxy	$q^{@\mathbf{L}}$: equiv. query expression at **L**	q^0: equiv. expression at zero level

Figure 1: Notation and central notions for proxies, signatures and areas.

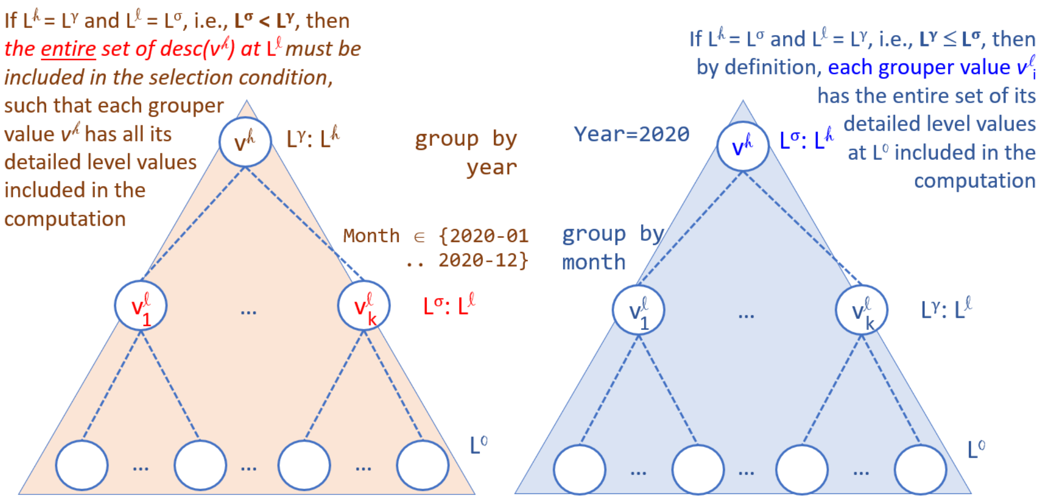


Figure 2: Perfect Rollability

Definition 4.2. Given a multidimensional schema S and a simple selection condition ϕ to which it participates, a dimension D with a grouper level $D.L^Y$ at the schema level and a filter level $D.L^\sigma$ at ϕ, is characterized as follows:

- *unbound*, if $D.L^\sigma = D.ALL$ and the atom of *phi* is $D.ALL \in \{D.all\}$ (equiv., *true*)
- *pinned grouper*, if both $D.L^Y$ and $D.L^\sigma \neq D.ALL$
- *pinned non-grouper*, if $D.L^Y = D.ALL$ and $D.L^\sigma \neq D.ALL$

Example. Assume a cube over *Date*, *Education*, *Workclass* with *TaxPaid* as measure. Assume that two queries roll-up *Geography* at level *ALL*, and report sales per month and product family. Let $q^o = \langle \mathbf{DS}^0, \phi^o, [Month, W.L_1, E.ALL, sumTaxPaid], [sum(TaxPaid)] \rangle$ be a query over this cube.

Then, *Month* and $W.L_1$ are groupers and *Education* is a non-grouper.

Concerning Education:

- if the atom $E.ALL \in \{All\}$ is part of ϕ than the dimension is unbound, i.e., all the members of the education dimension are computed for the final result
- if an atom like $E.L_3 \in \{Post-secondary\}$ is part of ϕ, then the dimension is a pinned non-grouper

Concerning Date:

- if the atom $Date.ALL \in \{All\}$ is part of ϕ than the dimension is unbound
- if an atom like $D.Year \in \{2019, 2020\}$ is part of ϕ, then the dimension is a pinned grouper

Definition 4.3 (Perfectly Rollable Dimension / Perfectly Rollable atom). Assume a grouper level $D.L^Y$ and an atom $\alpha{:}D.L^\sigma \in V$, $V = \{v_1, \ldots, v_k\}$. Then, the dimension D is *perfectly rollable* with respect to the tuple (L^Y, L^σ, V), or, equivalently, α is *perfectly rollable* with respect to L^Y, if one of the following two conditions holds:

(a) $L^Y \preceq L^\sigma$ (which implies that every grouper value of L^Y that qualifies is entirely included, as the selection condition is put at a higher level that the grouping, e.g., group by month, for year = 2020)

(b) $L^\sigma \prec L^Y$, and for each value $u_i \in dom(L^Y)$: $u_i = anc_{L^\sigma}^{L^Y}(v_i)$, *all* $desc_{L^Y}^{L^\sigma}(u_i) \in V$ (i.e., the entire set of children of a grouper value u is included in the computation of u).

Definition 4.4 (Perfectly Rollable Schema / Perfectly Rollable simple selection condition). Assume a schema S: $[D_1.L_1, \ldots, D_n.L_n]$ over a set of dimensions $[D_1, \ldots, D_n]$ with each grouper level belonging to a different dimension and a simple selection condition ϕ: $\bigwedge_{i=1}^{n} \alpha_i$, with each atom α of the form $D.L^{\sigma} \in V$, $V = \{v_1, \ldots, v_k\}$, and exactly one atom per dimension. Then, the schema S is *perfectly rollable* with respect to the tuple (S, ϕ), or, equivalently, ϕ is *perfectly rollable* with respect to S, if each atom α_i is perfectly rollable with respect to its respective grouper level L_i.

The perfectly rollable condition is a "clean" characterization stating that if we group by a level L on *any* possible data set, then, the resulting grouper values of L will be produced by the entire population of their descendants at lower levels (in fact, as far as the semantics are concerned: the most detailed one – see Figure 2). *Perfect rollability guarantees that, given a simple selection condition on a dimension and a grouper level, there are no grouper cells in the result of a cube that could be computed on the basis of only a subset of their detailed descendants, but rather, the entire range of descendant values are taken into consideration for their computation.*

5 CUBE USABILITY: COMPUTING A CUBE FROM ANOTHER CUBE WHOSE RESULT IS AVAILABLE

In this Section, we address the main problem for this paper: assume we have computed a previous query q^b and we want (a) to check whether the contents of a new query q^n are derivable from the cells of q^b, and (b) if this is the case indeed, to actually perform the computation. To support our discussion, in the sequel, we assume two queries, to which we refer to as q^b (with the hidden implication of "broad" in terms of selection condition, "below" in terms of the level of the grouping, and, "before" in terms of creation) and q^n (with the hidden implication of "narrow" in terms of selection condition, "not lower" in terms of the level of the grouping, and, "new" in terms of creation). As in all other cases, we will assume that the selection conditions are simple selection conditions, including n atoms, each of the form $D.L \in \{v_1, \ldots, v_k\}$. We also assume that all aggregate functions are *distributive* (e.g., sum, max, min, count). To simplify the presentation even more, we assume that there is a one-to-one mapping between the measures of the two queries and the respective aggregate functions that produce them (thus, the two queries differ only at the levels of their schema and their selection conditions). See the long version of the paper [22] for the proof and the formalization of distributive functions.

Theorem 5.1 (Cube Usability). *Assume the following two queries:*

$$q^n = \left\langle \mathbf{DS}^0,\ \phi^n,\ [L_1^n, \ldots, L_n^n, M_1, \ldots, M_m],\ [agg_1(M_1^0), \ldots, agg_m(M_m^0)] \right\rangle$$

and

$$q^b = \left\langle \mathbf{DS}^0,\ \phi^b,\ [L_1^b, \ldots, L_n^b, M_1, \ldots, M_m],\ [agg_1(M_1^0), \ldots, agg_m(M_m^0)] \right\rangle$$

The query q^b is usable for computing*, or simply,* usable for *query q^n, meaning that Algorithm 1 correctly computes $q^n.cells$ from $q^b.cells$, if the following conditions hold:*

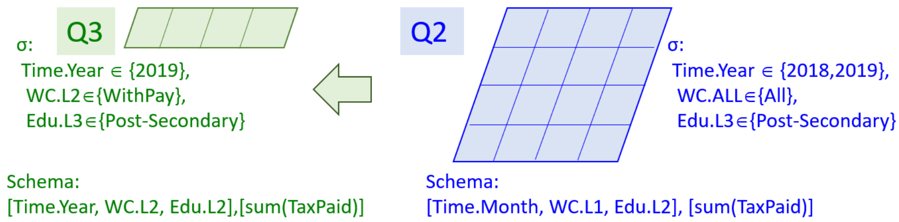
(1) *both queries have exactly the same underlying detailed cube* **DS**,
(2) *both queries have exactly the same dimensions in their schema and the same aggregate measures $agg_i(M_i^0)$, $i \in 1..m$ (implying a 1:1 mapping between their measures), with all agg_i belonging to a set of known distributive functions. To simplify notation, we will assume that the two queries have the same measure names,*
(3) *both queries have exactly one atom per dimension in their selection condition, of the form $D.L \in \{v_1, \ldots, v_k\}$ and selection conditions are conjunctions of such atoms,*
(4) *both queries have schemata that are perfectly rollable with respect to their selection conditions, which means that grouper levels are perfectly rollable with respect to the respective atom of their dimension,*
    - *(for convenience) for both queries, for all dimensions D having $D.L^g$ as a grouper level and $D.L^{\phi}$ as the level involved in the selection condition's atom for D, we assume $D.L^g \preceq D.L^{\phi}$, i.e., the selection condition is defined at a higher level than the grouping*
(5) *all schema levels of query q^n are ancestors (i.e, equal or higher) of the respective levels of q^b, i.e., $D.L^b \preceq D.L^n$, for all dimensions D, and,*
(6) *for every atom of ϕ^n, say α^n, if (i) we obtain $\alpha^{n@L^b}$ (i.e., its detailed equivalent at the respective schema level of the previous query q^b, L^b) to which we simply refer as $\alpha^{n@b}$, and, (ii) compute its signature $\alpha^{n@b^+}$, then (iii) this signature is a subset of the grouper domain of the respective dimension at q^b (which involves the respective atom α^b and the grouper level L^b), i.e., $\alpha^{n@b^+} \subseteq gdom(\alpha^b, L^b)$.*

Algorithm 1: Answer Cube Query from a Pre-Existing Query Result

---

Input: A new query expression q^n and a previously computed query q^b along with its result $q^b.cells$
Output: The result of q^n, $q^n.cells$

begin
  $q^n.cells \leftarrow$ compute q^{n^+} and for every coordinate, create a new cell with all measures initialized to $\emptyset$
  if *q^b and q^n satisfy all conditions of Theorem 5.1* **then**
    forall *dimensions D_i* **do**
      $\alpha_i^{n@b} \leftarrow$ the transformed atom of the new query at the schema grouper level L_i^b of q^b
    end
    $\phi^{n@b} = \wedge\ \alpha_i^{n@b}$
    $q^{n@b}.cells \leftarrow$ apply $\phi^{n@b}$ to $q^b.cells$
    q^{n^G} = group the cells of $q^{n@b}.cells$ according to q^{n^+}
    forall *measures M_j* **do**
      $q^n.cells.M_j \leftarrow$ apply agg_j^F to the j-th measure of the members of the groups of q^{n^G}
    end
  end
  return $q^n.cells$
end



	Schema L^g			Atoms L^ϕ		
	Time	WC	Edu	Time	WC	Edu
Q3	Year	L2	L2	Year ∈ {2019}	WC.L2 ∈ {WithPay}	Edu.L3 ∈ {Post-Sec}
Q2	Month	L1	L2	Year ∈ {2018, 2019}	WC.ALL∈{All}	Edu.L3 ∈ {Post-Sec}

	Time	WC	Edu
Q^{3+}	{2019}	X {WithPay}	X {SomeColl., Assoc., Univ., PostGrad}
$\phi^{3@2+}$ $Q^{3@2+}$	{2019-01, ... 2019-12}	X {Private, SelfEmp, Gov}	X {SomeColl., Assoc., Univ., PostGrad}
ϕ^{2+}	{2018-01, ... 2018-12, 2019-01, ... 2019-12}	X {Private, SelfEmp, Gov, W/OPay}	X {SomeColl., Assoc., Univ., PostGrad}

Figure 3: An example of cube usability

6 CONCLUSIONS

In this paper we have provided a method for computing a new cube from a previous one, defined at a different level of abstraction. The basis of the method is perfect rollability, a property characterizing the combination of selection conditions and groupers that guarantees the correct computation of aggregate measures.

Future work can also target operators comparing cubes for intrinsic properties of their cells (e.g., hidden correlations, predictions, classifications) that have to be decided via the application of knowledge extraction operators to the results, or the detailed areas, of the contrasted cubes.

ACKNOWLEDGMENTS

The author has been co-financed by the European Regional Development Fund of the European Union and Greek national funds through the Operational Program Competitiveness, Entrepreneurship and Innovation, under the call Research - Create - Innovate (prj code:T2EDK-02848).

REFERENCES

[1] Ashok K. Chandra and Philip M. Merlin. 1977. Optimal Implementation of Conjunctive Queries in Relational Data Bases. In *9th Annual ACM Symposium on Theory of Computing (STOC)*. 77–90.

[2] Surajit Chaudhuri, Ravi Krishnamurthy, Spyros Potamianos, and Kyuseok Shim. 1995. Optimizing Queries with Materialized Views. In *Proceedings of the 11th International Conference on Data Engineering(ICDE)*. 190–200.

[3] Surajit Chaudhuri and Kyuseok Shim. 1996. Optimizing Queries with Aggregate Views. In *5th International Conference on Extending Database Technology (EDBT)*. 167–182.

[4] Sara Cohen. 2005. Containment of Aggregate Queries. *SIGMOD Record* 34, 1 (March 2005), 77–85.

[5] Sara Cohen. 2009. Aggregation: Expressiveness and Containment. In *Encyclopedia of Database Systems*, Ling Liu and M. Tamer Özsu (Eds.). Springer US, 59–63.

[6] Sara Cohen, Werner Nutt, and Yehoshua Sagiv. 2003. Containment of Aggregate Queries. In *Database Theory - ICDT 2003, 9th International Conference, Siena, Italy, January 8-10, 2003, Proceedings (Lecture Notes in Computer Science, Vol. 2572)*, Diego Calvanese, Maurizio Lenzerini, and Rajeev Motwani (Eds.). Springer, 111–125.

[7] Sara Cohen, Werner Nutt, and Yehoshua Sagiv. 2006. Rewriting queries with arbitrary aggregation functions using views. *ACM Trans. Database Syst.* 31, 2 (2006), 672–715.

[8] Sara Cohen, Werner Nutt, and Yehoshua Sagiv. 2007. Deciding equivalences among conjunctive aggregate queries. *J. ACM* 54, 2 (2007), 5.

[9] S. Cohen, W. Nutt, and A. Serebrenik. 1999. Rewriting Aggregate Queries Using Views. In *18th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*.

[10] S. Dar, H.V.Jagadish, A. Levy, and D. Srivastava. 1996. Answering queries with aggregation using views. In *22nd International Conference on Very Large Data Bases (VLDB)*.

[11] Stéphane Grumbach, Maurizio Rafanelli, and Leonardo Tininini. 2004. On the equivalence and rewriting of aggregate queries. *Acta Informatica* 40, 8 (2004), 529–584.

[12] Stephane Grumbach and Leonardo Tininini. 2003. On the content of materialized aggregate views. *J. Comput. System Sci.* 66, 1 (2003), 133–168.

[13] Ashish Gupta, Venky Harinarayan, and Dallan Quass. 1995. Aggregate-Query Processing in Data Warehousing Environments. In *Proceedings of 21th International Conference on Very Large Data Bases(VLDB)*. 358–369.

[14] Alon Halevy. 2001. Answering Queries using Views: A Survey. *The VLDB Journal* 10 (2001), 270–294.

[15] Christian S. Jensen, Torben Bach Pedersen, and Christian Thomsen. 2010. *Multidimensional Databases and Data Warehousing*. Morgan & Claypool Publishers.

[16] Per-Åke Larson and H. Z. Yang. 1985. Computing Queries from Derived Relations. In *11th International Conference on Very Large Data Bases (VLDB)*. 259–269.

[17] Alon Y. Levy, Alberto O. Mendelzon, Yehoshua Sagiv, and Divesh Srivastava. 1995. Answering Queries Using Views. In *Proceedings of the 14th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems(PODS)*. 95–104.

[18] Werner Nutt, Yehoshua Sagiv, and Sara Shurin. 1998. Deciding Equivalences Among Aggregate Queries. In *Proceedings of the 17th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems(PODS)*. 214–223.

[19] Oscar Romero and Alberto Abelló. 2007. On the Need of a Reference Algebra for OLAP. In *Proceedings of DaWaK*. 99–110.

[20] Dimitri Theodoratos and Timos K. Sellis. 2000. Answering Multidimensional Queries on Cubes Using Other Cubes. In *Proceedings of the 12th International Conference on Scientific and Statistical Database Management (SSDBM), Berlin, Germany, July 26-28, 2000*. 109–123.

[21] Vasilis Vassalos. 2009. Answering Queries Using Views. In *Encyclopedia of Database Systems*, Ling Liu and M. Tamer Özsu (Eds.). Springer US, 92–98.

[22] Panos Vassiliadis. 2022. A Cube Algebra with Comparative Operations: Containment, Overlap, Distance and Usability. *CoRR* abs/2203.09390 (2022). arXiv:2203.09390 https://arxiv.org/abs/2203.09390

[23] Panos Vassiliadis and Spiros Skiadopoulos. 2000. Modelling and Optimisation Issues for Multidimensional Databases. In *12th International Conference on Advanced Information Systems Engineering (CAiSE 2000), Stockholm, Sweden, June 5-9, 2000*. Springer, 482–497.