Controlling the Correctness of Aggregation Operations During Sessions of Interactive Analytic Queries

Eric Simon, Bernd Amann, Rutian Liu, Stéphane Gançarski

▶ **To cite this version:**

Eric Simon, Bernd Amann, Rutian Liu, Stéphane Gançarski. Controlling the Correctness of Aggregation Operations During Sessions of Interactive Analytic Queries. 2022. hal-03772799

HAL Id: hal-03772799
https://hal.science/hal-03772799v1

Preprint submitted on 8 Sep 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

arXiv:2111.13927v2 [cs.DB] 6 Dec 2021

Controlling the Correctness of Aggregation Operations During Sessions of Interactive Analytic Queries

ERIC SIMON, SAP France, France
BERND AMANN, LIP6 – Sorbonne Université, CNRS, France
RUTIAN LIU, SAP France, LIP6 – Sorbonne Université, CNRS, France
STÉPHANE GANÇARSKI, LIP6 – Sorbonne Université, CNRS, France

We present a comprehensive set of conditions and rules to control the correctness of aggregation queries within an interactive data analysis session. The goal is to extend self-service data preparation and BI tools to automatically detect semantically incorrect aggregate queries on analytic tables and views built by using the common analytic operations including filter, project, join, aggregate, union, difference, and pivot. We introduce *aggregable properties* to describe for any attribute of an analytic table, which aggregation functions correctly aggregate the attribute along which sets of dimension attributes. These properties can also be used to formally identify attributes which are *summarizable* with respect to some aggregation function along a given set of dimension attributes. This is particularly helpful to detect incorrect aggregations of measures obtained through the use of non-distributive aggregation functions like average and count. We extend the notion of summarizability by introducing a new *generalized summarizability condition* to control the aggregation of attributes after any analytic operation. Finally, we define *propagation rules* which transform aggregable properties of the query input tables into new aggregable properties for the result tables, preserving summarizability and generalized summarizability.

CCS Concepts: • **Information systems → Data management systems**; *Data provenance*; *Inconsistent data*; **Data warehouses**.

Additional Key Words and Phrases: analytic queries, summarizability, data quality, multi-dimensional data model, interactive query sessions

1 INTRODUCTION

1.1 Problem statement and motivations

Analytic datasets are ubiquitous in numerous application domains and their usage includes, for example, the classic reporting on business activities in transactional applications [26], the monitoring of the behavior of on-line systems based on log analysis (e.g., Splunk [49], Elasticsearch/Kibana [25], Datadog [11]), trend analysis in finance or social networks, or the conduct of epidemiological studies in healthcare [17]. In a world where an overwhelming amount of raw data is collected and stored at an affordable price in cloud object stores (e.g., Amazon S3 [45], Azure Blob Storage [35]), properly aggregated and cleaned data is the data foundation layer on which "augmented" analytics are built with the help of machine learning pipelines.

The creation and maintenance of analytic datasets for supporting Business Intelligence (BI) applications has traditionally been the entitlement of experienced data engineers in IT organizations. Today, the emergence of self-service data preparation and BI tools (e.g., [37, 46, 53], [41, 42, 51]) empowers business users and data scientists to directly create and mash up analytic datasets according to their needs. With these tools, data analysis becomes an interactive and iterative process whereby a user issues a data analysis action (translated into a query), receives a result, and possibly

Authors' addresses: Eric Simon, eric.simon@sap.com, SAP France, France; Bernd Amann, bernd.amann@lip6.fr, LIP6 – Sorbonne Université, CNRS, France; Rutian Liu, rutian.liu.fr@gmail.com, SAP France, LIP6 – Sorbonne Université, CNRS, France; Stéphane Gançarski, stephane.gancarski@lip6.fr, LIP6 – Sorbonne Université, CNRS, France.

decides which action to perform next. Eventually, a user may decide to share the final analytic dataset thus obtained in the form of a reusable view. Interactive data analysis sessions facilitate the exploration and creation of analytic datasets, even for users lacking knowledge of SQL, MDX and any programming languages.

However, data experts who directly manipulate analytic datasets created by others expose themselves to possible disappointments, particularly when data aggregation – the most common operation done by analysts – is involved. Imagine a simple use case with the analytic datasets shown in Table 1, representing multidimensional *facts* that hold *measures* and refer to one or more hierarchical *dimensions* [21]. The dimension table *REGION* (Table 1b) describes a list of cities. These cities are referenced by the fact table DEM(ographics) which contains three dimension attributes CITY, STATE, COUNTRY from dimension *REGION* and one attribute YEAR from another dimension table *TIME* (not displayed). Attributes POP and UNEMP are measure attributes that respectively represent the population and the unemployment rate in that city.

Table 1. Fact and dimension tables for demographics

(a) Fact table DEM (Demographics)

CITY	STATE	COUNTRY	YEAR	POP	UNEMP (%)
Dublin	California	USA	2017	61	3.1
Palo Alto	California	USA	2017	67	2.1
Dublin	California	USA	2018	63	3.0
Palo Alto	California	USA	2018	66	2.0
San Jose	California	USA	2018	1,028	2.2
Dublin	Ohio	USA	2018	44	3.7
Washington D.C	-	USA	2018	672	6.2
Dublin	-	Ireland	2018	1,348	6.71

(b) Dimension table *REGION*

CITY	STATE	COUNTRY	REGION
Dublin	California	USA	North America
Palo Alto	California	USA	North America
San Jose	California	USA	North America
Dublin	Ohio	USA	North America
Washington D.C	-	USA	North America
Dublin	-	Ireland	Europe

Suppose that a business user wants to aggregate the measures in the DEM fact table. A first concern is to express aggregations that produce semantically correct results. For measure POP, any common aggregation function can be used, but the dimension attributes along which aggregation can be done must be restricted to CITY, STATE and COUNTRY. That is, aggregation can only be done within every partition of DEM by YEAR, otherwise the population will be double counted for the cities of 'Palo Alto' and 'Dublin' in California. For measure UNEMP, only a limited set of aggregation functions can be applied (MIN, MAX), because the attribute represents a ratio that cannot be summed or averaged along any dimension. Expressing valid aggregation operations therefore requires a clear understanding of the semantics of measure attributes and the dimensions on which they depend. Ideally, the querying system should automatically control which aggregation operation is valid using metadata properties that express the above restrictions on the DEM table.

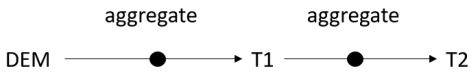


Fig. 1. Interactive data analysis session 1 over DEM

Now, suppose that a business user, in the interactive data analysis session displayed in Figure 1, first wants to count (without duplicates) the number of cities per state, and country. This can be achieved using a "roll-up" action which aggregates the DEM data along attribute CITY and attribute YEAR. This action can be translated into a SQL aggregate query on table DEM by doing a COUNT_DISTINCT(CITY) group by STATE and COUNTRY whose result table T_1 is displayed in Table 2a (the count has been renamed into NB_CITIES which is a measure).

Table 2. Results of aggregate queries in the session of Figure 1

(a) Table T_1

NB_CITIES	STATE	COUNTRY
1	Ohio	USA
3	California	USA
1	-	USA
1	-	Ireland

(b) Table T_2

SUM(NB_CITIES)	COUNTRY
5	USA
1	Ireland

(c) Table T_2'

SUM(NB_CITIES)	COUNTRY
7	USA
1	Ireland

Later, suppose that the business user, in the same interactive session, aggregates further NB_CITIES by COUNTRY using function SUM, yielding a new table T2 displayed in Table 2b. The value of SUM(NB_CITIES) in T_2 is however hard to interpret: for country 'USA', it is neither the count with duplicates nor the count without duplicates of cities by country, if we refer to the original table DEM. If the intention of the user was to obtain a count without duplicates of cities, the result of that interactive session is *incorrect*. On the other hand, if the first aggregate query in the session of Figure 1 was counting cities *with duplicates*, and the subsequent aggregate query was summing NB_CITIES as before, the result table T_2' of the interactive session, displayed in table Table 2c, would be *correct*. It is easy to figure out that this problem is non-trivial for a non-expert user.

This problem is known as a *summarizability* issue: we shall say that attribute CITY is *not* summarizable with respect to grouping set {STATE, COUNTRY} and function COUNT_DISTINCT using function SUM. As before, a business user may expect that the querying system controls what aggregation is valid on table T_1 using proper metadata associated with that table. Thus, if the user cannot compute a global count of cities *without duplicates* per country using T_1, she would have to backtrack within the interactive session to a result over which such a global count is expressible (in our example, backtrack to the original table DEM).

This summarizability issue can be generalized to an arbitrary sequence of interactive analytic queries. Consider the analytic datasets shown in Table 3. The dimension table *SALESORG* describes a list of stores that are referenced by the fact table STORE_SALES containing four dimension attributes STORE_ID, CITY, STATE and COUNTRY from dimension *SALESORG* and one attribute YEAR from dimension *TIME*. Attribute AMOUNT is a measure attribute and UNIT is a detail attribute of that measure.

A data analyst might want to build a new analytic dataset, named SALES_DEM_USA, using the interactive data analysis session shown in Figure 2 (the dashed lines will be explained later). First, STORE_SALES is filtered on

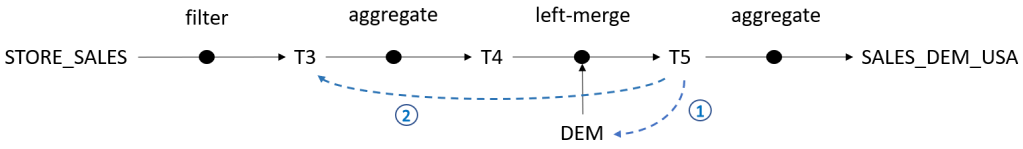


Fig. 2. Interactive data analysis session 2 yielding SALES_DEM_USA

COUNTRY = 'USA' and YEAR = '2018', yielding a table named T3. Then, an aggregate SUM(AMOUNT) is computed for each partition of CITY, STATE, COUNTRY, YEAR, yielding a table named T4 displayed in Table 4. At this point, we can control that each tuple in T4 is *correct* because it would also be a tuple in the result of the *same* aggregate query computed over STORE_SALES (the original data).

Table 3. Fact and dimension tables for store sales

(a) Dimension table *SALESORG*

STORE_ID	CITY	STATE	COUNTRY
Ca_01	Dublin	California	USA
Sa_01	San Jose	California	USA
Oh_01	Dublin	Ohio	USA
Wa_01	Washington DC	-	USA
Du_01	Dublin	-	Ireland

(b) Fact table STORE_SALES

STORE_ID	CITY	STATE	COUNTRY	YEAR	AMOUNT	UNIT
Ca_01	Dublin	California	USA	2018	5.3	mega dollar
Ca_02	Dublin	California	USA	2018	1.4	mega dollar
Ca_01	Dublin	California	USA	2017	3.5	mega dollar
Sa_01	San Jose	California	USA	2018	22.8	mega dollar
Oh_o1	Dublin	Ohio	USA	2018	1.2	mega dollar
Wa_o1	Washington DC	-	USA	2018	16.1	mega dollar
Wa_o2	Washington DC	-	USA	2018	27.6	mega dollar
Du_01	Dublin	-	Ireland	2018	7.8	mega euro

Next, the schema of T4 is augmented with the measure attribute POP of table DEM, yielding a new table named T5 (Table 5a). This latter action, called a *left-merge*, can be translated into a natural left outer join SQL query between T4 and DEM on attributes CITY, STATE, COUNTRY and YEAR. Thus, in table T5, attributes CITY, STATE and COUNTRY represent attributes of both dimensions *REGION* and *SALESORG*. However, measure attribute POP depends on dimension *REGION* while attribute SUM(AMOUNT) depends on dimension *SALESORG*.

In the last step of the interactive data analysis session 2, the measure attributes SUM(AMOUNT) and POP of T5 are summed by STATE, COUNTRY and YEAR, yielding the final result SALES_DEM_USA displayed in Table 5b. However, the value of SUM(POP) in SALES_DEM_USA is *misleading* because it does not correspond to the population of each state as it would be obtained from the DEM table. Indeed, the population of cities without any store, such as the city of 'Palo Alto', has not been counted. Thus, the aggregation along CITY of SUM(POP) should not be allowed on T5 (or at least,

Table 4. Result T4 in session 2 of Figure 2

CITY	STATE	COUNTRY	YEAR	SUM(AMOUNT)
Dublin	California	USA	2018	6.7
San Jose	California	USA	2018	22.8
Dublin	Ohio	USA	2018	1.2
Washington DC	-	USA	2018	43.7

a warning must be raised that it only accounts for the population of cities in dimension *SALESORG*). To obtain the total population of each state, suppose that the user backtracks to the previous step of the session and expresses the summation of POP along CITY on table DEM, yielding a new table DEM′ (in Table 6a), before performing the left-merge operation. This backtracking is depicted by the dashed line labelled "1" in Figure 2.

Table 5. Results T5 and SALES_DEM_USA in session of Figure 2

(a) Result T5 in session of Figure 2

CITY	STATE	COUNTRY	YEAR	SUM(AMOUNT)	POP
Dublin	California	USA	2018	6.7	61
San Jose	California	USA	2018	22.8	1,028
Dublin	Ohio	USA	2018	1.2	44
Washington	-	USA	2018	43.7	672

(b) Fact table SALES_DEM_USA with misleading SUM(POP)

STATE	COUNTRY	YEAR	SUM(AMOUNT)	SUM(POP)
California	USA	2018	29.5	1,089
Ohio	USA	2018	1.2	44
-	USA	2018	43.7	672

However, after performing the left-merge of T4 with DEM′ (result is displayed in Table 6b), the summation of SUM(POP) should again be disallowed. Indeed, it would be *incorrect* with respect to the same summation computed over table DEM′, since population of California would be double-counted. The proper explanation is that tuples from DEM′ match multiple tuples of T4 because they don't have the same dimension granularity.

Hence, the user has to backtrack to T3 (backtracking is depicted by the dashed line labelled "2" in Figure 2) and aggregate SUM(AMOUNT) by STATE, COUNTRY and YEAR, yielding a new table T4′. After merging T4′ with DEM′, the final table SALES_DEM_USA is obtained, as displayed in Table 7. The actual flow of interactive queries that produced the final result is displayed in Figure 3.

The previous examples hopefully showed that it is easy for an end user such as an analyst to perform erroneous or misleading aggregation operations during an interactive data analysis session. This motivated our design of a method that automatically controls the validity of aggregation operations and provides explanations that are easy to understand for an end user.

Table 6. Results after first backtracking in session of Figure 2

(a) Fact table DEM′

STATE	COUNTRY	YEAR	SUM(POP)
California	USA	2017	61
California	USA	2017	128
California	USA	2018	1,157
Ohio	USA	2018	44
-	USA	2018	672
-	Ireland	2018	1,348

(b) Result of the left-merge of T4 with DEM′

CITY	STATE	COUNTRY	YEAR	SUM(AMOUNT)	SUM(POP)
Dublin	California	USA	2018	6.7	1,157
San Jose	California	USA	2018	22.8	1,157
Dublin	Ohio	USA	2018	1.2	44
Washington	-	USA	2018	43.7	672

Table 7. Result of the left-merge of T4′ with DEM′ with correct SUM(POP)

STATE	COUNTRY	YEAR	SUM(AMOUNT)	SUM(POP)
California	USA	2018	29.5	1,157
Ohio	USA	2018	1.2	44
-	USA	2018	43.7	672

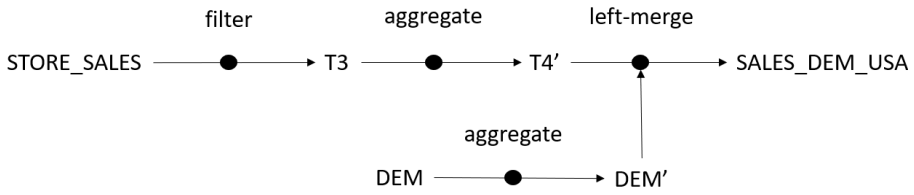


Fig. 3. Final flow of interactive queries yielding a correct instance of SALES_DEM_USA

1.2 Limitations of previous related work

As mentioned before, the occurrence of an incorrect sequence of two aggregations is known as a *summarizability* problem. In the original definition of the problem [30], an initial fact table represents "micro-data", at the finest granularity level, and a *summarization query* is expressed over an attribute A of this fact table, yielding another fact table representing "macro-data". A summarization query over a fact table performs an aggregation operation $F(A)$ using a function F for each partition of the table defined by a grouping set of attributes X. In essence, the problem of summarizability is to determine whether, for some summarization query over attribute $F(A)$ of the macro-data using a function G (possibly identical to F), there exists a summarization query over attribute A of the micro-data using F that returns exactly the same result. If this is the case, the summarization query over $F(A)$ of the macro-data is said to be

correct. For instance, using the previous example of Figure 1, if table DEM represents the micro-data, and table T1 represents the macro-data obtained after summarizing attribute CITY using function COUNT_DISTINCT, then the query that summarizes NB_CITIES using function SUM is *incorrect*.

In the most general formulation of the problem, attribute A in the micro-data is defined to be summarizable with respect to a grouping set X and a function F using function G if for any subset of attributes Z of X, any aggregation $G(F(A))$ with grouping set Z over the macro-data is correct (with the above meaning). Ideally, we want to determine the largest subset of attributes of X for which the above summarizability condition holds.

To address the summarizability problem, a first group of *model-based* solutions proposes to model dimension and fact tables in a restricted and controlled way so that any aggregation query over a previously aggregated fact table is always correct. Some solutions even propose to modify the hierarchical dimension data to enforce the restrictions that assure the summarizability of aggregate queries. See [34] for a survey of these solutions and more recently [8].

A second group of *constraint-based* solutions defines summarizability constraints over the schemas and the data of dimension and fact tables, which can be evaluated to determine whether an attribute of a fact table is summarizable with respect to a grouping set using an aggregation function. We focus our work on this second group of solutions which, instead of imposing constraints on the analytic data model, control the summarizability of aggregate queries to avoid incorrect results. These solutions are better suited to an environment where analytic data is created by multiple independent parties using different data modeling techniques.

However, the detailed analysis of the best existing constraint-based solutions to the summarizability problem [20, 27, 28, 30, 40], reveals the following limitations. Firstly, within the hierarchy of a dimension, any non-null value of an attribute must map to a single parent attribute value. This discards the use of dimension tables like *SALESORG* (in Table 3a), wherein a city can have multiple states. Secondly, measure attributes in a fact table must depend on all the identifiers of the dimensions over which the fact table is defined. This discards the use of fact tables such as the result of the left-merge of T4 with DEM′ in Table 6b, where measure attribute SUM(POP) depends only on STATE, COUNTRY and YEAR. Thirdly, it is assumed that aggregate (summarization) operations do not handle null values in their grouping set, which creates too restrictive conditions for summarizable queries in the case of SQL aggregate operations. Finally, summarizability conditions depend on the size of the dimension data, because they require either testing disjointness conditions over fact table partitions defined by some dimension attributes [30] or reasoning on *dimension constraints* whose number depends on the paths that exist in the dimension hierarchies [20].

Another important limitation is that the existing methods do not consider the case when an aggregate query occurs after another type of query such as a filter or a left-merge query, as in the data analysis session of Figure 2. In real life scenarios though, "mash-up" queries are popular because analytic data is often siloed in the context of a specific business activity (e.g., product marketing, medical care) or a particular application domain (e.g., monitoring system logs). For example, analytic data on hospitalized patients contains measures on those patients that are treated in hospitals. Other analytic data may contain measures about ambulatory patients who are treated by medical doctors in the city, or measures about patient demographics. Typical epidemiological studies therefore require mashing up this data using filter and merge operations.

Other research efforts related to the problem studied in this paper exist. Some works address the correctness issues raised by the semantics of various aggregation functions and their applicability depending on the domain of values on which they are applied [29, 31] or the issues caused by the SQL implementation of relational operations when they are applied over an empty set of values or over a set of values containing *null* values [10, 16]. In our work, we assume a standard implementation of SQL aggregate queries for *null* values.

Other works propose methods which automatically control or enforce the consistency of arithmetic and aggregation query results with respect to the scales, units and currencies associated with measure attributes in fact tables [18, 44, 52]. We are not addressing this problem which is complementary to the correctness issues targeted by this paper.

Finally, recommendation-based approaches suggest queries for the interactive exploration of databases. The *collaborative filtering* approach uses previously collected query logs of a dataset (SQL queries in [13, 24], and OLAP queries in [1, 33]) to recommend queries on the same initial dataset. The *data-driven approach* [5, 9, 12, 23, 47, 48], recommends a single type of exploration actions whose result are expected to optimize a measure of "interestingness" with respect to the current analysis context of a user on a given dataset. For instance, [23] suggests different "drill-down" operations on a given table, each producing a different set of tuples. However, these works do not control the correctness of the recommended "drill-down" exploration actions with respect to the summarizability property we introduced before.

1.3 Research contributions

In this article we present a comprehensive set of conditions to control the *correctness of aggregation queries* within a data analysis session consisting of a large variety of interactive queries, which includes the most common operations that are supported by self-service data preparation and BI tools, such as filter, project, inner and outer joins, aggregate, union, difference, and pivot. These operations also subsume the traditional operations used in interactive exploration sessions of OLAP cubes, such as roll-up, drill-down, dice, and slice and our correctness criteria for aggregate queries include as a special case the summarizability property addressed by previous work.

The *analytic data model* we introduce in Section 2 is more expressive than the other data models considered by the previous work on summarizability, because it accepts arbitrary dimension hierarchies and fact tables. In our data model, dimension tables are defined as views over non-analytic tables (that is, regular relational tables), and fact tables are initially defined as views over dimension tables and non-analytic tables. Then, new fact or dimension tables are defined as the result of interactive queries over previously defined dimension and fact tables.

At the core of our approach is the definition of two types of metadata associated with analytic tables which help to check the correctness of analytic queries. Firstly, *attribute graphs* are used to describe literal functional dependencies between the attributes of hierarchical dimensions with possible *null* values (as in the example of dimension *SALESORG*). We showed in a previous paper [32] how to efficiently compute attribute graphs through the analysis of dimension data samples. Secondly, *aggregable properties* describe, for any attribute of an analytic table, which aggregation functions can be used, and along which set of dimension attributes these aggregation functions can be applied. *Default rules* assist the designer of a table to define aggregable properties when the table is initially created as a view from source data (i.e., from non-analytic tables) and, using these properties, it is then possible to automatically control which aggregations are possible on an analytic table.

The central technical results of this article are *propagation rules*, which automatically compute the aggregable properties for a table resulting from an interactive analytic query and thereby allow us to control the *correctness* of aggregate queries at any stage of an interactive data analysis session. Our correctness criteria for aggregate queries include the semantic properties of measure attributes (like in the first examples of aggregate queries over table DEM that we presented before). Furthermore, these criteria not only subsume the sufficient conditions defined in previous work to assure that aggregate queries are expressed over summarizable attributes, but their propagation makes it also possible to characterize the results of sessions composing two or more aggregate queries as being correct, with respect to summarizability, when previous work would view them as being incorrect. Conversely, any aggregate query that previous works characterize as correct is also detected as correct using our aggregable properties. Finally, in the case of

a sequence composed of any interactive query followed by an aggregate query, we introduce the novel notion of *G (generalized) summarizability* to characterize correct aggregate queries.

In summary, we make the following main research contributions:

(1) In Section 3, we extend the notion of aggregable properties introduced in [32], as a general means to express, for any attribute of an analytic table, which aggregation functions are correctly applicable along which sets of dimension attributes. We use aggregable properties to express the semantic properties of measures previously defined in [19, 26, 30, 36, 40, 50] and provide default rules to minimize the effort of end users for defining aggregable properties on analytic tables built from source data (i.e., non-analytic data). We then provide a first set of propagation rules to automatically compute aggregable properties in the results of interactive analytic queries.
(2) In Section 4.1, we formally define summarizability conditions for attributes and in Section 4.2, we refine our propagation rules for the case of aggregate queries to compute aggregable properties of attributes such that subsequent aggregate queries over these attributes can only be expressed if the attributes are summarizable. In Section 5, we show that our aggregable properties subsume the summarizability conditions defined in previous work [20, 27, 28, 30, 40].
(3) In Section 4.3, we introduce the new notion of G-summarizability that extends the summarizability property of attributes to the case of an aggregate query expressed over the result of an arbitrary analytic query. We then refine our propagation rules in Section 4.4 to compute aggregable properties such that aggregate queries over some attributes can be expressed only if these attributes are G-summarizable.
(4) Finally, in Section 5 we focus on previous works that propose conditions on the schema of a fact table, or on the parameters of an aggregate query expressed over that fact table, to determine if the aggregate query returns a correct result with respect to some summarizability definition. In our analysis, we establish that our data model is more general than the data models considered by previous work. In the case of a sequence of two aggregate queries, Q_1 followed by Q_2, our sufficient conditions to determine if Q_2 is correct, subsume the conditions proposed by previous work. To the best of our knowledge, no previous work addressed the case of a sequence made of an arbitrary analytic query followed by an aggregate query, which is addressed by our notion of G-summarizability.

2 MULTI-DIMENSIONAL DATA MODEL AND ANALYTIC QUERIES

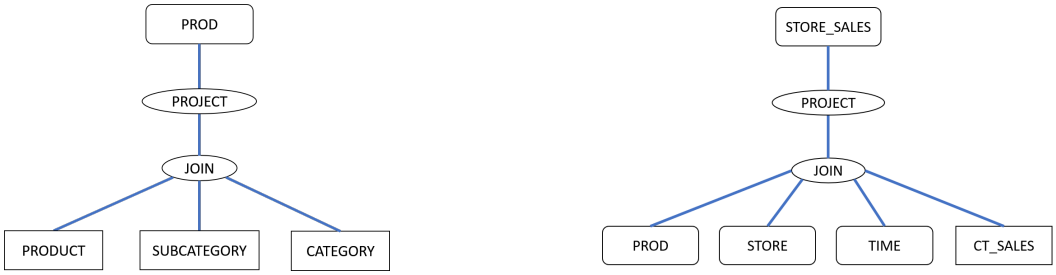
In this section, we first present our multidimensional data model, composed of dimension and fact tables, and some logical and structural constraints on dimensions expressed using an *attribute graph*. We then present the types of analytic queries that can be expressed on our data model. We use conventional relational database notations [14]. Each table T is a finite multiset of tuples over a set of domains of values $S = \{A_1, ..., A_n\}$, called *attributes*, where each domain may contain a *null* marker. We call S the *schema* of T.

2.1 Dimension and fact tables

We consider datasets in which data is separated into *non-analytic tables* and *analytic tables*. Non-analytic tables correspond to relational tables storing the source data. *Analytic tables*, or *analytic views*, are defined by queries over non-analytic and analytic tables.

Example 1. Figure 4 details the definitions of two analytic tables STORE_SALES and *PROD*. The analytic table *PROD* represents a dimension that is defined by a join-project query over three non-analytic tables (represented by

square rectangles). The analytic table STORE_SALES represents a fact table that is defined by a join-project query over a non-analytic table ct_SALES and three analytic tables *TIME*, *PROD* and *STORE* representing dimensions.



(a) Definition of dimension table *PROD*

(b) Definition of fact table STORE_SALES

Fig. 4. Two analytic tables (views)

Attributes in analytic tables are categorized into two types: *dimension attributes* and *measures*. Dimension attributes describe entities like stores, customers and dates whereas measure attributes are used to define facts about these entities. Following this distinction of attributes, analytic tables are categorized into two types: *dimension tables* and *fact tables*. An analytic table is a *dimension table* if it only contains *dimension attributes* and a *fact table* if it contains at least one or more *dimension attributes* from one ore more dimensions and one *measure attribute*.

Example 2. A complete example of analytic tables, which will be used throughout this paper, is shown in Table 8. Dimension table names are in italic font to distinguish them from fact tables. The first three tables are dimension tables identifying and describing products (dimension *PROD*), stores (dimension *SALESORG*) and dates (dimension *TIME*). The schema of fact tables STORE_SALES was already introduced in Section 1. Fact table PRODUCT_LIST, shown in Table 8e, describes the sold quantity (QTY) of products (attributes PROD_SKU, BRAND and COUNTRY from dimension *PROD*), by year (attribute YEAR from dimension *TIME*). Attribute value "-" in these tables represents a null marker.

We consider a classical multi-dimensional data model which organizes a set of dimension attributes X into an *attribute hierarchy* noted $(X, \preccurlyeq)$. Unlike several other models, which we shall review in Section 5, we make no special assumption on the attribute hierarchy: there can be one or more bottom or top level attributes, and an attribute can have multiple parents.

A *hierarchy instance* of an attribute hierarchy $\mathcal{A} = (X, \preccurlyeq)$ is a set of values N and a partial order $\leq$, where N contains for each attribute $\mathrm{X}_i \in X$ a non empty subset of values $N_i \subseteq N$ such that each order relation $v_i \leq v_j$ preserves the ancestor/descendant relation $\preccurlyeq^*$ between the corresponding attributes X_i and X_j, *i.e.*, $v_i \in N_i, v_j \in N_j \Rightarrow \mathrm{X}_i \preccurlyeq^* \mathrm{X}_j$. We also assume that $(N, \leq)$ is *transitively reduced*, *i.e.*, there is no pair of values that is connected by an order relation $(\leq)$ and a sequence of two or more order relations.

Example 3. The left part of Figure 5 illustrates an *attribute hierarchy* for dimension *PROD*: PROD_SKU$\preccurlyeq$ BRAND $\preccurlyeq$ COUNTRY, and PROD_SKU $\preccurlyeq$ SUBCATEGORY $\preccurlyeq$ CATEGORY, in which PROD_SKU is a bottom level attribute and CATEGORY and COUNTRY are two top level attributes. An instance of that attribute hierarchy is displayed on the right where attribute values are horizontally aligned with the name of each attribute.

We can now formally define dimension and fact tables.

Table 8. Fact and dimension tables

(a) Dimension table *PROD*

PROD_SKU	BRAND	COUNTRY	SUBCATEGORY	CATEGORY
coco-can-33cl	Coco Cola	USA	Soft Drinks	Drinks
coco-can-25cl	Coco Cola	USA	Soft Drinks	Drinks
cz-tshirt-s	Zora	Spain	Woman Tops	Clothes
cz-tshirt-s	Coco Cola	USA	Woman Tops	Clothes

(b) Dimension table *SALESORG*

STORE_ID	CITY	STATE	COUNTRY
Oh_01	Dublin	Ohio	USA
Ca_01	Dublin	California	USA
Ca_02	Palo Alto	California	USA
Pa_01	Paris	-	France
Ly_01	Lyon	-	France
Ir_01	Dublin	-	Ireland

(c) Dimension table *TIME*

DATE	WEEK	MONTH	YEAR
1/1/2018	1	1	2018
2/1/2018	1	1	2018
3/1/2018	1	1	2018
…	…	…	…

(d) Fact table STORE_SALES

STORE_ID	CITY	STATE	COUNTRY	YEAR	AMOUNT
Oh_01	Dublin	Ohio	USA	2017	3.2
Ca_01	Dublin	California	USA	2017	5.3
Oh_01	Dublin	Ohio	USA	2018	8.2
Ca_01	Dublin	California	USA	2018	6.3
Pa_01	Paris	-	France	2017	45.1

(e) Fact table PRODUCT_LIST

PROD_SKU	BRAND	COUNTRY	YEAR	QTY
cz-tshirt-s	Coco Cola	USA	2017	5 000
cz-tshirt-s	Coco Cola	USA	2018	7 000
cz-tshirt-s	Zora	Spain	2017	5 000
cz-tshirt-s	Zora	Spain	2018	7 000
coco-can-33cl	Coco Cola	USA	2017	10 000

Definition 1 (Dimension table)**.** A dimension table D over some attribute hierarchy $\mathcal{A} = (\mathrm{S}, \preccurlyeq)$ is a table $D(\mathrm{S})$ where each tuple t of D corresponds to a complete path in the hierarchy instance of $\mathcal{A}$. Attributes of S are henceforth called *dimension attributes*.

In practice [26], dimension tables also include *detail attributes* that functionally depend on one or more dimension attributes, and these dependencies are part of the metadata of the dimension table. Examples of detail attributes for the dimension table *SALESORG* could be: ZIPCODE, COUNTRY_CODE, STORE_NAME, STORE_SQUARE_METERS, etc. We shall not consider such detail attributes because they do not impact the results presented in this paper.

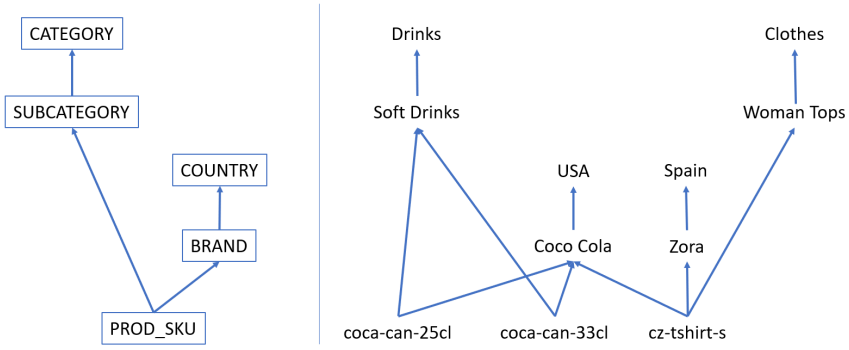


Fig. 5. Attribute hierarchy and hierarchy instance defined by dimension table *PROD*

Definition 2 (Fact table)**.** A *fact table* over a set of dimensions $D_1, \cdots, D_n$ is a table $T(\mathrm{S})$ without any duplicate where schema S contains a non-empty subset X_i of dimension attributes from dimension D_i, and a non-empty set of attributes Z representing one or more *measures*. Each tuple of values $t.X_i$ in T has a corresponding tuple of values in D_i.

In practice [26], each measure in a fact table is usually represented by one attribute having the role of *Value* and a possibly empty group of attributes having the role of *Detail*. The *Value* attribute of a measure carries the actual value while the *Detail* attributes provide optional auxiliary information on the measure such as a unit (as in example of fact table STORE_SALES in the introduction) or a currency. In this paper, we shall not discuss how to control the quality of aggregation queries with respect to different units and currencies, and refer the interested reader on this topic to [44]. So we shall later only consider measure attributes carrying actual values.

2.2 Literal functional dependencies and attribute graphs

Null markers in dimension attributes represent *non applicable values*. This semantics is different from other interpretations where null values represent missing or unknown values and are considered as placeholders for non-null values. We consider null markers as regular values and apply the same literal equality semantics as in SQL unique constraints (see e.g.,[14]): two attribute values $t_1.\mathrm{A}$ and $t_2.\mathrm{A}$ are *literally equal*, denoted by $t_1.\mathrm{A} \equiv t_2.\mathrm{A}$, iff $t_1.\mathrm{A} = t_2.\mathrm{A}$ or both values are null markers. Observe that $t_1.\mathrm{A} = t_2.\mathrm{A}$ implies $t_1.\mathrm{A} \equiv t_2.\mathrm{A}$ but the opposite is not true. Literal equality naturally extends to sets of attributes and leads to the notion of *Literal Functional Dependencies (LFD)* [3]. Let X and Y be two sets of attributes in a schema S, an LFD $\mathrm{X} \mapsto \mathrm{Y}$ holds for some table T over S iff for any two tuples t_1, t_2 of T, when $t_1.\mathrm{X} \equiv t_2.\mathrm{X}$ then $t_1.\mathrm{Y} \equiv t_2.\mathrm{Y}$. Note that if X does not contain any nullable attribute, the LFD $\mathrm{X} \mapsto \mathrm{Y}$ is equivalent to the *Functional Dependency with Nulls (NFD)* $\mathrm{X} \rightarrow \mathrm{Y}$ [2]. A set of LFDs on a schema S expresses semantic properties constraining the possible "valid" tables over S.

LFDs provide a formal system to define a set of logical and structural constraints over dimension tables. However, their practical use for characterizing a set of valid dimension tables is limited. The number of LFDs might rapidly increase for non-linear hierarchy types and the rule-based syntax does not exploit the hierarchical type structure to help user in defining validity constraints. In [32], we thus introduced the notion of *attribute graph*, which is a graph representation for LFDs in dimension tables, that characterizes all possible "valid" hierarchy instances of a dimension in a simple and natural way.

We provide an informal definition of attribute graphs through the following example.

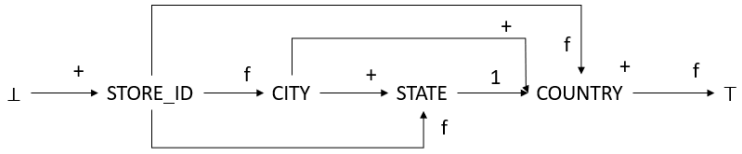


Fig. 6. Attribute graph of dimension *SALESORG*

Example 4. Figure 6 shows an attribute graph that is validated by dimension table *SALESORG*. First, the lower and upper bound attributes are respectively STORE_ID and COUNTRY. In the attribute graph, they are therefore respectively connected to special nodes ⊥ and ⊤. Second, for each pair of parent-child attributes in the attribute hierarchy of the dimension, there is a corresponding edge in the attribute graph. This yields edges: (STORE_ID, CITY), (CITY, STATE) and (STATE, COUNTRY). An additional edge is added between two attributes when the attributes between them in the attribute hierarchy can have *null* values and the higher attribute functional depends (literally or not) on the lower attribute). This yields edges: (STORE_ID, STATE), (STORE_ID, COUNTRY), and (CITY, COUNTRY), because both CITY and STATE can have null values. Third, each edge is assigned a unique label encoding the presence of functional (label 1) or literal functional dependency constraints (label f), or none of them (+), between the connected attributes.

By convention, the labels of outgoing edges of ⊥ are labelled as + and the labels of the incoming edges of ⊤ are labelled as f. Attribute STORE_ID literally determines CITY, STATE and COUNTRY and, therefore, the three arcs (STORE_ID, CITY), (STORE_ID, STATE) and (STORE_ID, COUNTRY) are labeled by f. The arc (CITY, STATE) is labeled by +, which signifies that tuples with the same (possibly *null*) value for CITY can have different values for STATE. Similarly, the arc (CITY, COUNTRY) is labeled by +. Finally, the arc (STATE, COUNTRY) is labeled by 1, since there exists a functional dependency from non-null STATE values to COUNTRY, but not a literal functional dependency.

Similarly, we define the attribute graph of dimension *PROD* in Figure 7.

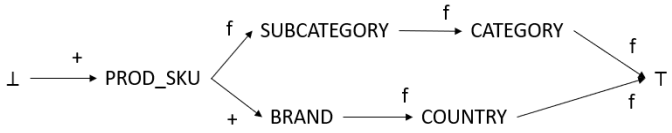


Fig. 7. Attribute graph of dimension *PROD*

Attribute graphs must be defined by the designer of dimension tables. However, we showed in [32] that attribute graphs can also be automatically and efficiently computed from dimension tables or samples thereof. More details about the acquisition and maintenance of attribute graphs can be found in [44].

We also provided in [32] an efficient algorithm to compute the minimum set of dimension attributes (called *dimension identifier*) that literally determines all other dimension attributes. Using the same properties of attribute graphs, we can determine if a set of dimensions attributes U in a dimension table, literally determines a dimension attribute B (*i.e.* $U \mapsto$ B).

2.3 Analytic queries

An *interactive data analysis session* consists of a tree of interactive analytic queries having one or more (input) analytic tables as leaves and a single root which is the final result of the session. We assume that the result of any interactive data analysis session can be saved as an *analytic view* whose definition is the tree of interactive queries that have been performed in the session. Hence, views can be reused to start a new interactive data analysis session. As usual, users can backtrack in their session and come back to a previous result. An example of an interactive data analysis session was given in Figure 3.

In this paper, we consider analytic queries consisting of unary operations (filter, project, aggregation, pivot, and binary operations (union, difference, merge). All these operations, except pivot, are based on relational operations but their semantics is tailored to the case of analytic tables by restricting their usage depending on the type of attributes (dimension or measure) that are manipulated. Our set of operations includes the most common data transformation operations supported by self-service data preparation and BI tools [37, 41, 42, 46, 51, 53]. They also subsume the traditional interactive operations on a multidimensional (OLAP) cube, such as Roll-up, Drill-down, Slice, or Dice, as defined for instance in [15, 21, 54, 56]. In this section, we precisely define the semantics of these operations with a special attention to their manipulation of null values.

2.3.1 *Analytic filter queries.*

The first single table analytic queries allow users to select a subset of tuples in the input table.

Definition 3 (Filter query)**.** Let $T(\mathrm{S})$ be an analytic table. We denote by $Q(T) = \mathit{Filter}_T(P \mid \mathrm{Y})$, an *analytic filter query* that returns all tuples in T satisfying a predicate P on a set of attributes $\mathrm{Y} \subseteq \mathrm{S}$.

Observe that P can be any well-formed Boolean predicate using negation, conjunction and disjunction over any subset of attributes in S. We consider that predicate P is a Boolean function which is also defined for tuples with *null* value attributes: except for literal equality, any comparison of an attribute value with a null marker evaluates to false.

Analytic filter queries support operations on a multidimensional cube known as "slice" (selection by subset of values of a dimension) or "dice" (selection by subset of values of more than one dimension) [56]. However, in our definition, a filter predicate can be expressed on any attribute.

Example 5. Consider the table $T(\mathrm{S})$ in Table 9a. The result of two filter queries $Q_1 = \mathit{Filter}_T(\{\mathrm{A}_1 = \text{`}a_1\text{'}\} \mid \{\mathrm{A}_1\})$ and $Q_2 = \mathit{Filter}_T(\{\mathrm{A}_2 \neq \text{`}b_2\text{'}\} \mid \{\mathrm{A}_2\})$ are shown in Table 9b and Table 9c.

Table 9. Filter queries

T	A_1	A_2	A_3	M	N
	a_1	b_1	c_1	x_1	y_1
	a_1	b_1	–	x_2	y_2
	a_2	b_1	c_1	x_3	y_3
	a_2	b_2	–	x_4	y_4

(a) Input table T

Q_1	A_1	A_2	A_3	M	N
	a_1	b_1	c_1	x_1	y_1
	a_1	b_1	–	x_2	y_2

(b) $\mathit{Filter}_T(\{\mathrm{A}_1 = a_1\} \mid \{\mathrm{A}_1\})$

Q_2	A_1	A_2	A_3	M	N
	a_1	b_1	c_1	x_1	y_1
	a_1	b_1	–	x_2	y_2
	a_2	b_1	c_1	x_3	y_2

(c) $\mathit{Filter}_T(\{\mathrm{A}_2 \neq b_2\} \mid \{\mathrm{A}_2\})$

2.3.2 *Analytic projection queries.*

Projection can be used to remove measure attributes and add new calculated measure attributes.

Definition 4 (Analytic projection query)**.** Let $T(\mathrm{S})$ be an analytic table with dimension attributes $\mathrm{S}_D \subseteq \mathrm{S}$. Let Y be a subset of S such that $\mathrm{S}_D \subseteq \mathrm{Y} \subseteq \mathrm{S}$. Let $f(\mathrm{Z}) \rightarrow A$ be an optional expression where $f(\mathrm{Z})$ is an expression involving a set of attributes $\mathrm{Z} \subseteq \mathrm{S}$, constants, arithmetic operators and string operators, and A is a new name for a measure attribute that results from the calculation implied by $f(Z)$. We denote by $Q(T) = \mathit{Project}_T(\mathrm{Y}, f(\mathrm{Z}) \rightarrow \mathrm{A})$ (resp. $Q(T) = \mathit{Project}_T(\mathrm{Y})$) an *analytic projection* which returns a table T_r with schema $\mathrm{Y} \cup \{A\}$ (resp. Y), such that for every tuple T of T, there exists a unique tuple t' in T_r such that $t'.B = t.B$ for every $B \in \mathrm{Y}$, and $t'.\mathrm{A} = f(t.Z)$.

An analytic projection over an analytic table $T(\mathrm{S})$ is a special case of an *extended projection* [14]. It can add a new measure attribute, whose value for each tuple is possibly computed from the values of other attributes of that tuple. The definition can easily be extended to a set of expressions $f(Z) \rightarrow A$. Note that expression $f(Z) \rightarrow A$ is optional in a projection query.

Table 10. Analytic projection queries

T	A1	A2	A3	M	N
	a_1	b_1	c_1	x_1	y_1
	a_1	b_1	–	x_2	y_2
	a_2	b_1	c_1	x_3	y_3
	a_2	b_2	–	x_4	y_4

(a) Input table T

Q_3	A1	A2	A3	M
	a_1	b_1	c_1	x_1
	a_1	b_1	–	x_2
	a_2	b_1	c_1	x_3
	a_2	b_2	–	x_4

(b) $\mathit{Project}_T(\{\mathrm{A1}, \mathrm{A2}, \mathrm{A3}, \mathrm{M}\})$

Q_4	A1	A2	A3	M′
	a_1	b_1	c_1	$x_1 + y_1$
	a_1	b_1	–	$x_2 + y_2$
	a_2	b_1	c_1	$x_3 + y_3$
	a_2	b_2	–	$x_4 + y_4$

(c) $\mathit{Project}_T(\{\mathrm{A1}, \mathrm{A2}, \mathrm{A3}\}, (\mathrm{M} + \mathrm{N}) \rightarrow \mathrm{M}')$

Example 6. Reconsider the table $T(\mathrm{S})$ in Table 10a. Table 10b and Table 10c show the result of two projections. The first projection $\mathit{Project}_T(\{\mathrm{A1}, \mathrm{A2}, \mathrm{A3}, \mathrm{M}\})$ simply keeps a subset of attributes of S whereas the second projection creates a new attribute M′ which is the sum of M and N.

Projections must keep all dimension attributes of the original table. To remove dimension attributes, we introduce aggregate queries as explained next.

2.3.3 *Analytic aggregate queries.*

Aggregate queries generally partition analytic tables along a subset of dimension attributes and aggregate the values of certain attribute in each partition. Analytic aggregate queries support operations on a multidimensional cube known as "roll-up" (aggregation of data from a lower level to a higher level of granularity within a dimension hierarchy) or "dice" (grouping of data with respect to a subset of dimensions of a cube).

Definition 5 (Analytic aggregate query)**.** Let $T(\mathrm{S})$ be an analytic table with dimension attributes $\mathrm{S}_D \subseteq \mathrm{S}$, A be an aggregable attribute in S, and F be an aggregation function. We denote by $Q(T) = \mathit{Agg}_T(\mathrm{F}(\mathrm{A}) \mid \mathrm{X})$ where $\mathrm{X} \subseteq \mathrm{S}_D$, an *analytic aggregate query* on table T that aggregates A using aggregation function F with group-by attributes X. We say that T is *aggregated along* A using F. The result contains one tuple for every tuple of distinct values of attributes in X including *null* values (as for SQL group-by operations).

The above definition can be easily generalized by replacing attribute A with a set of attributes. Unlike SQL Rollup [15, 54], note that our definition does not include tuples that represent subtotals in the query result. This facilitates the composition of aggregate queries, without having to deal with these special tuples, and better fits the purpose of interactive data analysis sessions.

Example 7. Reconsider the table T(S) in Table 11a with dimensional attributes A_1, A_2 from D_1 and A_3 from dimension D_2. The result of $Q_5 = \mathcal{A}gg_T(\text{SUM}(M) \mid \{A_3\})$ is shown in Table 11b. Note that, as with SQL semantics, a group-by operator supports literal equality semantics for *null* values. The result of aggregate query $Q_6 = \mathcal{A}gg_T(\text{SUM}(M) \mid \{A_1, A_3\})$ is shown in Table 11c.

Table 11. Analytic aggregate queries

(a) Input table T

T	A_1	A_2	A_3	M	N
	a_1	b_1	c_1	x_1	y_1
	a_1	b_1	$-$	x_2	y_2
	a_2	b_1	c_1	x_3	y_3
	a_2	b_2	$-$	x_4	y_4

(b) $\mathcal{A}gg_T(\text{SUM}(M) \mid \{A_3\})$

Q_5	A_3	SUM(M)
	c_1	$x_1 + x_3$
	$-$	$x_2 + x_4$

(c) $\mathcal{A}gg_T(\text{SUM}(M) \mid \{A_1, A_3\})$

Q_6	A_1	A_3	SUM(M)
	a_1	c_1	x_1
	a_1	$-$	x_2
	a_2	c_1	x_3
	a_2	$-$	x_4

2.3.4 *Analytic pivot queries.*

Pivot queries also partition tables along a subset of dimension attributes. But instead of aggregating all values of a non partitioning attribute into a single value for each partition, it generates a new attribute for each value. Analytic pivot queries are particularly useful in the data preparation phase of machine learning application scenarios like feature engineering [32, 57]. They should not be mistaken with the OLAP cube pivot operation that keeps the schema of the input table unchanged.

Definition 6 (Analytic pivot query)**.** Let T(S) be a fact table with dimension attributes $S_D \subseteq S$ and A be a measure attribute in S. We denote by $Q(T) = \mathcal{P}ivot_T(A \mid X)$, where $X \subset S_D$, an *analytic pivot query* which pivots attribute A over X. The result is a table T_r with all attributes in $S_D - X$ and an attribute A_v_i for each value v_i in the domain of $T.X$. The value $t.A$ of each tuple $t \in T$ such that $t.X = v_i$ is a value in the attribute A_v_i of the unique tuple t' in T_r such that $t.(S_D - X) = t'.(S_D - X)$.

The above definition can be easily generalized by replacing attribute A with a set of attributes.

Example 8. Reconsider the table T(S) in Table 12a. The result of pivot query $Q_7 = \mathcal{P}ivot_T(M \mid A_1)$ that pivots attribute M over A_1 is shown in Table 12b. The schema of the resulting table T_r contains all attributes in $S - A_1$ and two new attributes M_a_1 and M_a_2 for each value of $T.A_1$. The value $t.M$ of each tuple $t \in T$ such that $t.A_1 = v$ is a value in the attribute M_v of the unique tuple t' in T_r such that $t.(\{A_2, A_3\}) = t'.(\{A_2, A_3\})$. The result of another pivot query $Q_8 = \mathcal{P}ivot_T(M \mid A_2)$ is shown in Table 12c.

2.3.5 *Analytic merge queries.*

Analytic left-merge queries combine the tuples of two analytic tables and correspond to natural left outer-join operations defined in the extended relational algebra with null values. Analytic left-merge queries play an important

Table 12. Analytic pivot queries

T	A1	A2	A3	M	N
	a_1	b_1	c_1	x_1	y_1
	a_1	b_1	–	x_2	y_2
	a_2	b_1	c_1	x_3	y_3
	a_2	b_2	–	x_4	y_4

(a) Input table T

Q_7	A2	A3	M_a_1	M_a_2
	b_1	c_1	x_1	x_3
	b_1	–	x_2	–
	b_2	–	–	x_4

(b) $\mathcal{P}ivot_T$(M | A1)

Q_8	A1	A3	M_b_1	M_b_2
	a_1	c_1	x_1	-
	a_1	–	x_2	-
	a_2	c_1	x_3	-
	a_2	–	-	x_4

(c) $\mathcal{P}ivot_T$(M | A2)

role in so-called schema augmentation scenarios [32] and can support the OLAP cube operation known as "drill-down" (the inverse of roll-up) by merging a fact table, that provides a higher-level of granularity within a dimension hierarchy for some measures, with a fact table that provides a lower level of granularity for the exact same measures.

In our definition, we allow the merge of two fact tables on their common dimension attributes (which have the same names in the two fact tables) but we accept that the common attributes belong to different dimensions in each fact table. An example of such a merge operation was given in Section 1, between SALES_STORES and DEM, on common attributes CITY, STATE, COUNTRY. These attributes belong to dimension *SALESORG* in SALES_STORES and dimension *REGION* in DEM.

Definition 7 (Analytic left-merge query)**.** Let $Q = \pi_X(T)$ where $T = T \rtimes_{P_1 \wedge \cdots \wedge P_k} T'$, T(S) and $T'(S')$ are two analytic tables, $\rtimes$ is a *left-outer join* operator, P_i are join equality predicates over a set of (common) dimension attributes Y, and π_X is the duplicate elimination relational projection over a set of attributes X defined below. Then Q is a *left-merge merge analytic query* if the following conditions hold:

(1) For each $A_i \in$ Y, $\exists P_i$ such that $P_i = (T.A_i = T'.A_i) \vee (T.A_i = null \wedge T'.A_i = null)$ (*null* is a literal).
(2) If for each pair of attributes $A_1, A_2 \in$ Y that belongs to both a dimension D_1 in T and a dimension D_2 in T' ($D_1 \neq D_2$), A_1 and A_2 are either connected with the same labelled paths in their respective attribute graphs or not connected by any path, then X $= S \cup S'$ else X $= S \uplus S'$ ($\uplus$ denotes disjoint union, *i.e.* union after renaming conflicting attributes).

In the following, we will abbreviate $Q = \pi_X(T \rtimes_{P_1 \wedge \cdots \wedge P_k} T'))$ by $Q = T \rtimes_Y T'$, where Y is the set of join attributes, call it a *left-merge query*, and refer to the result of Q as a *merge table*.

Item 1 manages the join predicates in the merge query in the presence of nulls (we apply literal equality which is different from the SQL equality semantics for null values). The merge table preserves all rows in T (with possible row duplication) and contains all attributes in T and T' (the merge query does not project out any attribute).

Item 2 checks that, when two different dimensions are joined on their common attributes, the structure and properties of their respective hierarchies for the joined attributes are identical, that is, the attribute graphs (defined in Section 2.2), restricted to all common attributes and all paths connecting these attributes, are identical. When this is not the case (e.g., they differ on their hierarchical relationships or they have different functional dependencies), the merge query keeps the join attributes separately for each table and applies a disjoint union.

Left-merge queries can also be generalized to a full-outer join ($\rtimes\ltimes$) between two tables, called an *analytic full merge query*, or restricted to a natural join ($\bowtie$), called an *analytic strict merge query*. Right-merge queries $Q(T, T') = T \rtimes\ltimes_Y T'$ are equivalent to the symmetric left-merge queries $Q(T', T) = T' \rtimes_Y T$ on the switched tables.

Example 9. Consider the fact tables T and T' in Table 13a and Table 13b, respectively defined over dimension D_1 (where $\mathrm{A}_1 \preccurlyeq \mathrm{A}_2 \preccurlyeq \mathrm{A}_3$) and dimension D_2 (where $\mathrm{A}_2 \preccurlyeq \mathrm{A}_3$). Suppose that in both dimensions, we have $\mathrm{A}_2 \mapsto A_3$, then by Item 2, since attributes A_2 and A_3 are connected by the same labelled paths in their respective attribute graphs, they appear only once in the merge table, and the result of a left merge $Q_{10} = T \rtimes T'$ is shown in Table 13c. If the labelled paths between A_2 and A_3 were different in the attribute graphs of D_1 and D_2, all the dimension attributes of T and T' will be kept separately in the result of the merge.

Table 13. Analytic merge queries

T	A1	A2	A3	M
	a_1	b_1	c_1	x_1
	a_1	$-$	c_2	x_2
	a_2	b_1	c_3	x_3
	a_3	$-$	c_2	x_4

(a) Input table T

T'	A2	A3	N
	b_1	c_1	y_1
	b_1	c_3	y_2
	b_2	c_4	y_3
	b_3	c_4	y_4

(b) Input table T'

Q_{10}	A1	A2	A3	M	N
	a_1	b_1	c_1	x_1	y_1
	a_1	$-$	c_2	x_2	$-$
	a_2	b_1	c_3	x_3	y_2
	a_2	$-$	c_2	x_4	$-$

(c) $T \rtimes T'$

2.3.6 *Analytic set queries.*

Analytic tables are sets of tuples and can therefore be combined using set operations. However, compared to standard relational set operations, analytic set operations must respect additional constraints related to the separation between dimension and measure attributes and the condition that all measure attributes are determined by a subset of dimension attributes.

Definition 8 (Analytic set queries)**.** Let T and T' be two analytic tables having the same schema with a set of dimension attributes Y (referring to the same dimensions):

- If $\pi_{\mathrm{Y}}(T) \cap \pi_{\mathrm{Y}}(T') = \emptyset$, $Q = T \cup T'$ is a union analytic query where $\cup$ is the set union operator.
- $Q = T - T'$ is a difference analytic query where "$-$" is the set difference operator based on literal equality of attribute values.

Observe that analytic intersection $T \cap T'$ is equivalent to $T - (T - T')$. Analytic union queries are useful to complement dimension or fact tables. A full merge query $Q = T \rtimes\!\subset_{\mathrm{Y}} T'$ between two analytic tables T and T' having exactly the same set of dimension attributes Y, and such that $\pi_{\mathrm{Y}}(T) \cap \pi_{\mathrm{Y}}(T') = \emptyset$, expresses an *analytic outer-union* between the two tables. Note that a *data fusion* operation [4] can be expressed as a full merge query followed by an analytic projection.

Example 10. Table 14 shows two tables and the result of two analytic set queries. Observe that the union $T \cup T'$ is not defined since $\pi_{A_1,A_2,A_3} T \cap \pi_{A_1,A_2,A_3} T' \neq \emptyset$.

3 AGGREGABILITY OF ATTRIBUTES IN ANALYTIC TABLES

An attribute of an analytic table does not necessarily aggregate with all aggregation functions along all dimension attributes. Describing when this aggregation is possible has been extensively studied for statistical and OLAP databases (see [34] for a survey). Focusing on function SUM, [26], [19] and [36] proposed that the designer of a fact table declares the *additivity* category of each measure: *fully-additive* measures can be summed along any dimension, *semi-additive*

Table 14. Analytic set queries

T	A1	A2	A3	M
	a_1	b_1	c_1	x_1
	a_1	b_1	$-$	x_2
	a_2	b_1	c_1	x_3
	a_2	b_2	$-$	x_4

(a) Input table T

T'	A1	A2	A3	M
	a_1	b_1	c_1	x_1
	a_1	b_1	$-$	x_2
	a_1	b_2	c_1	x_5
	a_2	b_2	c_1	x_6
	a_2	b_1	$-$	x_7

(b) Input table T'

T	A1	A2	A3	M
	a_2	b_1	c_1	x_3
	a_2	b_2	$-$	x_4

(c) $T - T'$

T	A1	A2	A3	M
	a_1	b_1	c_1	x_1
	a_1	b_1	$-$	x_2
	a_2	b_1	c_1	x_3
	a_2	b_2	$-$	x_4
	a_1	b_2	c_1	x_5
	a_2	b_2	c_1	x_6
	a_2	b_1	$-$	x_7

(d) $(T - T') \cup T'$

measures can be summed along some, but not all, dimensions, and *non-additive* measures cannot be summed along any dimension. This approach has been implemented in several OLAP systems.

Generalizing this approach, we introduce aggregable properties that enable a designer to declare for any attribute of an analytic table, which aggregation function is applicable and the set of dimension attributes along which this aggregation function can be computed. We introduce default rules that assist the designer of a table to define these properties. Finally, we show that aggregable properties can be automatically computed on the tables resulting from an analytic query, thereby saving the human effort to define them.

3.1 Aggregable properties of attributes

If some attribute A is aggregable along a set of dimension attributes X, then it is also aggregable along any subsets of X. In the following, we denote by $\mathrm{agg}_{\mathrm{A}}(\mathrm{F}, \mathrm{X})$ the *aggregable property* of A and state that property $\mathrm{agg}_{\mathrm{A}}(\mathrm{F}, \mathrm{X})$ holds in T if X is the maximal set of attributes along which A is aggregable using F in T. We now formally define aggregable properties $\mathrm{agg}_{\mathrm{A}}(\mathrm{F}, \mathrm{X})$.

Definition 9 (Aggregable Property)**.** Let S_D be the set of dimension attributes in an analytic table $T(\mathrm{S})$, A be an aggregable attribute in S and F be an aggregation function.

- Let $\mathrm{X}_f \subseteq \mathrm{S}_D$ be the set all dimension attributes B such that any aggregation of A with F along B is considered to be *meaningless* by the user. We call X_f the set of *forbidden* dimension attributes along which A cannot be aggregated using F.
- If A is a measure attribute, let $\mathrm{X}_d \subseteq \mathrm{S}_D$ be a minimal subset of dimension attributes such that $\mathrm{X}_d \mapsto \mathrm{A}$. Let X_d^+ be the set of all dimension attributes $\mathrm{B} \in \mathrm{S}_D$ such that $\mathrm{X}_d \mapsto \mathrm{B}$. We call X_d a determinant of A and X_d^+ the closure of X_d in S_D.

Then the aggregable property $\mathrm{agg}_{\mathrm{A}}(\mathrm{F}, \mathrm{X})$ holds in T for F and $\mathrm{X} \subseteq \mathrm{S}_D - \mathrm{X}_f$, where:

(1) Function F is *applicable* to A.
(2) If A is a measure attribute then $\mathrm{X} = \mathrm{X}_d^+ - \mathrm{X}_f$.
(3) If A is a dimension attribute then $\mathrm{X} = \mathrm{S}_D - \{\mathrm{A}\} - \mathrm{X}_f$.

Item 1 and the definition of the forbidden attributes X_f in Definition 9 cover the "information semantics" of an attribute A and restrict the functions and the dimensions for the aggregation of the attribute. Different categorizations have been proposed in the literature to determine the aggregation functions which are *applicable* to some measure

attribute, such as a statistic classification of measurements [36, 50, 52], the attribute's aggregation behaviour [39, 40], or the compatibility between the type of dimensions and the type of a measure [30]. These categorizations can also be used in our context to determine both the "applicability" of a function F and the set X_f of forbidden dimensions for a given measure attribute.

Item 2 covers the "logical semantics" defined by the literal functional dependencies between dimension and measure attributes. Essentially the closure X_d^+ of the determinant X_d of A contains all dimension attributes which are "logically related to" measure attribute A through literal functional dependencies. First, it is easy to see that all partitions generated by X_d^+ have the same value for A. Second, Item 2 considers that any aggregation of A along any subset of X_d^+ is *logically correct* and it is *semantically meaningful* if it also respects the applicability constraint (Item 1) and excludes the attributes in X_f. Symmetrically, all attributes that are not in X_d^+ are considered as logically independent of measure A and must be preserved by the partitioning (i.e., appear in the GROUP BY clause of an SQL query).

Finally, Item 3 mainly states that any dimension attribute can be aggregated along all other dimension attributes except those defined as "meaningless" in X_f. This follows from the observation that all dimension attributes are considered to be descriptive and the only aggregation functions that can be applied are COUNT and COUNT_DISTINCT (see also Table 16 below). Then, we assume that there exists no "logical" constraint defined by LFDs when counting some values along any "semantically meaningful" set of attributes (see Example 11).

Observe that in Item 2, there may exist several determinants X_d of A and each such determinants might define a different set of attributes X_i along which A can be aggregated using F. However, it is easy to show that if A can be aggregated along any subset of X_1 and any subset of X_2 using F, it also can be aggregated along any subset of the union $X_1 \cup X_2$.

In practice, the designer of a fact table is asked to indicate the set of semantically meaningless dimension attributes X_f and, in the case of a measure attribute only, a minimum set of logically correct dimension attributes X_d on which this attribute depends. The closure X_d^+ is *automatically obtained* using the attribute graphs of the dimensions. Thus, for each minimal set X_d provided by the user, Item 2 of Definition 9 gives the corresponding aggregable property of a measure attribute.

Example 11. Consider the fact table PRODUCT_LIST (PROD_SKU, COUNTRY, BRAND, YEAR, QTY) displayed in Table 15. Suppose that the designer of the fact table indicates that the measure attribute QTY literally depends on the minimal set of attributes X_d = {PROD_SKU, YEAR}, and that SUM aggregation is meaningful along any dimension attribute (i.e., $X_f = \emptyset$). Since X_d does not determine any other attribute, by Item 2 of the definition, X_d is the largest set of attributes for which the aggregable property $\text{agg}_{\text{QTY}}(\text{SUM}, X_d)$ holds in PRODUCT_LIST. In other words, QTY can only be aggregated along attributes PROD_SKU and YEAR. For dimension attribute PROD_SKU, suppose that the designer indicates that $X_f = \emptyset$, then $\text{agg}_{\text{PROD\_SKU}}(F, X)$ holds for $F \in \{\text{COUNT}, \text{COUNT\_DISTINCT}\}$ and X = {BRAND, COUNTRY, YEAR}.

Next, consider the fact table STORE_SALES (Table 3b) and suppose that the designer of the fact table indicates that the measure attribute AMOUNT depends on X_d = {STORE_ID, YEAR} and $X_f = \emptyset$ for function SUM. Since STORE_ID literally determines the dimension attributes CITY, STATE, and COUNTRY, by Item 2 of the definition, AMOUNT is aggregable along any subset of the dimension attributes of STORE_SALES.

Finally, consider the fact table DEM (Demographics) displayed in Table 1a. Summing up measure attribute POP along dimension attribute YEAR would clearly be incorrect, while it would be correct along any attribute of dimension *REGION*. Thus, the designer of the fact table should define X_f = {YEAR} for SUM and POP. After indicating that POP depends

on dimension attributes X_d = {CITY, COUNTRY, YEAR}, property $agg_{POP}(SUM, \{CITY, COUNTRY\})$ can automatically be computed.

Table 15. PRODUCT_LIST

PROD_SKU	BRAND	COUNTRY	YEAR	QTY
cz-tshirt-s	Coco Cola	USA	2017	5 000
cz-tshirt-s	Zora	Spain	2017	5 000
coco-33cl-can	Coco Cola	USA	2017	10 000

Two aggregable properties:

with user input: $agg_{QTY}(F, \{PROD\_SKU, YEAR\})$ for $F \in \{SUM, COUNT, AVG, ...\}$
default: $agg_{PROD\_SKU}(F, \{BRAND, COUNTRY, YEAR\})$ for $F \in \{COUNT, COUNT\_DISTINCT\}$

3.2 Default rules for aggregable properties

By inspecting the properties of measure attributes and aggregation functions, we define rules to obtain default aggregable properties for every aggregable attribute of every analytic table. The effort required from the designer of analytic tables is then to inspect and possibly correct the result produced by the application of the default rules, according to the known information semantics of attributes. With respect to Definition 9, the only possible corrective actions taken by a designer consist of adding dimension attributes to the forbidden attribute set X_f, or removing attributes from the determinant set X_d if X_d is not minimal.

Default applicable functions: Existing methods that categorize measures to determine the applicability of an aggregation function rely on some external knowledge and require an analysis of every aggregable attribute of an analytic table. To reduce the user effort, we provide a default categorization into three categories of attributes NUM (numerical), DESC (descriptive/categorical) and STAT (statistical). These categories can automatically be extracted from the schema metadata: the two categories NUM and DESC are inferred from the (SQL) data type of attributes and the category STAT denotes a result from the use of some statistical function. Table 16 describes the six common SQL aggregation functions applicable to each category, which will be used in the examples of this paper. We therefore use the attribute category of A to define which aggregation function F is applicable to A. As mentioned before, a scale-based categorization of measure attributes could also be used (e.g., [52]).

Table 16. Categories of attributes and their properties

Attribute category	Properties
NUM	• Numerical values • Applicable functions: SUM, AVG, COUNT, COUNT_DISTINCT, MIN, MAX
DESC	• Descriptive or categorical values • Applicable functions: COUNT, COUNT_DISTINCT
STAT	• Numerical statistical values • Applicable functions: COUNT, COUNT_DISTINCT, MIN, MAX

Default values of X_d and X_f for a measure attribute: If A is a measure attribute of T for which no minimal set of attributes that determines A has been defined by a user, then we use the default rule that A depends on all dimension attributes. This actually means that in Item 2, X_d contains the identifiers of all dimensions (automatically determined using the attribute graphs of the dimensions).

We implicitly assume that X_d is minimal, which is a necessary condition in the definition of aggregable property. If A does not logically depend on some dimension, this must be indicated by the designer of the fact table, and the corresponding dimension attributes are removed from X_d.

We assume by default that the set of meaningless attributes is empty ($X_f = \emptyset$). If there exists a "meaningless" aggregation along some dimensions (like in the fact table DEM of the previous example), this should be indicated by the designer of the fact table, by adding the corresponding dimension attributes to X_f.

Default value of X_f for a dimension attribute: As already mentioned before, if A is a dimension attribute, we assume that its category is DESC to determine the applicable aggregation functions (COUNT and COUNT_DISTINCT). By definition, we also assume that all aggregations using these two functions along any set of attributes (except A) are correct. As before, we also assume that there exist no meaningless aggregations, and we use the default rule that $X_f = \emptyset$.

Important consequence. We assure that each aggregable property $\mathrm{agg_A(F, X)}$, with its determinant X_d and forbidden attribute set X_f, is part of the metadata of attribute A in table T. This is particularly needed when a user takes some action to either minimize the default value of X_d or add attributes to X_f. Without keeping the values of X_d and X_f, it would not be possible to infer them from the value of X.

The default values and possible user actions are summarized in Table 17.

Table 17. Default values of X_d and X_f and possible user actions

Attribute	Default values	Possible user action
Measure	X_d = fact identifier; $X_f = \emptyset$	remove attributes to minimize X_d; add attributes to X_f
Dimension	$X_f = \emptyset$	add attributes to X_f

Example 12. Continuing with Example 11, since attribute QTY in table PRODUCT_LIST is of category NUM, we get from Table 16 the list of applicable aggregation functions. Then, by default, the minimum set of attributes X_d that determines QTY will be the fact identifier of PRODUCT_LIST, X_d = {PROD_SKU, BRAND, YEAR}. This set is however not minimal (QTY only depends on {PROD_SKU, YEAR}) and the designer of the table should remove attribute BRAND from X_d. Finally, using the default rule that $X_f = \emptyset$, we get the aggregable property displayed on the bottom of Table 15.

The dimension attribute PROD_SKU is of category DESC, which determines its applicable aggregation functions. Then, using the default rule for X_f, we get the aggregable property displayed on the bottom of Table 15. However, if the user considers that it makes no sense to count products along the time dimension, he might remove YEAR from the aggregable property by adding it to the set of forbidden attributes X_f.

3.3 Propagating aggregable properties

We wish to limit the effort required from the designers of analytic tables, regarding the verification and possible correction of default rules, to the case of analytic tables that are defined from non-analytic tables. As shown in Figure 8, this corresponds to dimension tables built from non-analytic tables, or fact tables built from dimension tables and non-analytic tables, using database queries (represented by bold arrows). These are the tables over which all other custom analytic tables are built, using self-service data preparation and BI tools.

For analytic tables that result from (analytic) queries over analytic tables with aggregable properties (represented by dashed arrows in Figure 8), the following two sections present *propagation rules* to obtain the aggregable properties of their attributes. In most of the cases, these rules do not require any user input.

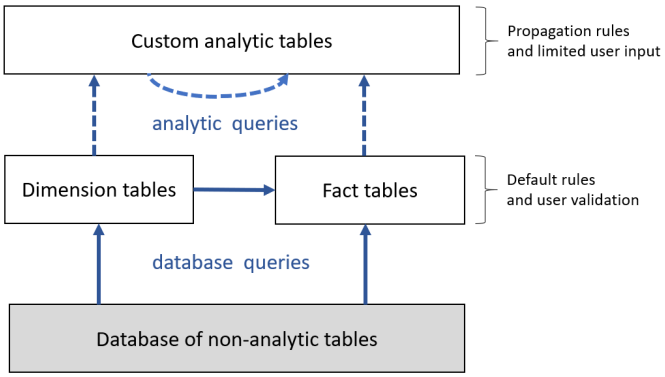


Fig. 8. Definition of aggregable properties

3.3.1 *Propagating aggregable properties to the results of unary operations.*

To determine the aggregable property of some attribute A' in the result $T_r = Q(T)$ of a query Q over T, we must first identify the aggregate functions which are applicable to A'. This falls into one of the following cases:

(1) If A' is also an attribute of T and F is applicable to A' in T then F is also applicable to A' in T_r.
(2) If A' holds pivoted values of an attribute A of T and F is applicable to A in T, then F is also applicable to A' in T_r.
(3) If $\mathrm{A}' = \mathrm{F}(\mathrm{A})$ is the result of applying some aggregation function F over an attribute A in T, then the aggregate functions that are applicable to A' are determined by the co-domain category of function F using Table 18.
(4) If A' is a new attribute resulting from the evaluation of an expression $f(\mathrm{Z}) \rightarrow A'$ in Q, then the aggregation functions that are applicable to A' are determined by the category of A' (default or user-defined) using Table 16.

Filter and pivot queries do not change the category of aggregable attributes of T that are in the result T_r. Therefore, all functions that were applicable for attributes in T, are still applicable to these attributes in the result of any filter or pivot query over T. This is not true for aggregate and projection queries which might generate new attribute values of a different category than the aggregated or projected attributes by applying a function. For example, while an attribute A of category NUM in T is still of category NUM in T_r when $\mathrm{F} = \mathrm{SUM}(\mathrm{A})$, the resulting attribute becomes of category STAT when $\mathrm{F} = \mathrm{AVG}(A)$. This change is detected using the classification in Table 18.

Table 18. Domain and co-domain categories for common SQL aggregation functions

Functions	Domain category	Co-domain category
SUM, MIN, MAX	NUM	NUM
MIN, MAX	STAT	STAT
COUNT, COUNT_DISTINCT	NUM, DESC, STAT	NUM
AVG	NUM	STAT

The identification of all aggregation functions F that are applicable to an attribute A′ in the result T_r of a query is not sufficient for defining the aggregable properties of A′ that hold in T_r. We must also determine for each attribute A′, the maximal subset of dimension attributes X′ of T_r along which an aggregation using F is correct. The propagation rules in Table 19 for determining X′, can be applied when A′ is an attribute in the result of a filter, projection, pivot or aggregate query. The last column shows how to determine the new determinant X'_d (for measure attributes) and the new forbidden attribute set X_f (for dimension and measure attributes) as well as the required user actions displayed in italics font (*None* means no action required).

Proposition 1 (Propagation rules for filter, project and pivot)**.** Let $T_r(S_r) = Q(T)$ be the result of a filter, project or pivot query Q over an analytic table $T(S)$, A′ be an attribute of T_r, and S_D be the set of dimension attributes in T. Then the propagation rules of Table 19 for filter, project and pivot are correct.

Proof. Suppose that whenever $A' \in T$ then $\mathrm{agg}_{A'}(F, X)$ holds in T, for $X \subseteq S_D$.

(1) Filter queries: Let $T_r = \mathit{Filter}_T(P \mid Y)$. Then T_r is a subset of T and all conditions, and in particular the literal functional dependencies, in the Definition 9 of for $\mathrm{agg}_{A'}(F, X)$ still hold for A′ and we obtain $X'_d = X_d$ and $X'_f = X_f$.

(2) Projection queries: Let $T_r = \mathit{Project}_T(Y, f(Z) \rightarrow M)$.
- $A' \in Y$: Since, by definition of projection, Y contains all dimension attributes of T, T_r also contains all dimension attributes of T (and possibly some other measure attributes). Therefore, all conditions in Definition 9 still hold for all measure and dimension attributes $A' \in Y$ and we obtain $X'_d = X_d$ and $X'_f = X_f$.
- For new measure attribute M, we have to show that X'_d must be a determinant of M: Since X'_d is a determinant of Z and M is the result of a function applied to attributes Z, by transitivity, X'_d is also a determinant of M.

(3) Pivot queries: Let $T_r = \mathit{Pivot}_T(A \mid Y)$.
- A′ in $S - Y - \{A\}$ and $\mathrm{agg}_{A'}(F, X)$ holds in T: If $X_d \cap Y = \emptyset$, by definition of pivot, each tuple $t \in T$ is mapped to a tuple $t' \in T_r$ where $t.(X_d \cup A') = t'.(X_d \cup A')$ and we obtain $X'_d = X_d$ is a determinant of A′. If $X_d \cap Y \neq \emptyset$, we cannot conclude that the remaining attribute set in $X_d - Y$ is still a determinant of A′ and we must apply the default rules to find X'_d. However, observe that A′ existed in the input table and we can conclude that all remaining forbidden attributes $X'_f = X_f - Y$ are still forbidden.
- A′ is a new attribute that holds pivoted values of A and $\mathrm{agg}_A(F, X)$ holds in T: If X_d is a determinant of A in T and $X_d \not\subseteq Y$, we can conclude that the "remaining" attributes $X'_d = X_d - Y$ are a determinant of A′. Suppose that there are two tuples t_1 and t_2 in T_r which have the same value for $X_d - Y$, but different values for attribute A′: $t_1.A' \neq t_2.A'$. By definition of pivot, these two tuples are the result of two distinct tuples t'_1 and t'_2 in T where $t_1.A' = t'_1.A'$, $t_2.A' = t'_2.A'$, $t'_1.Y = t'_2.Y$, and $t'_1.(X_d - Y) = t'_2.(X_d - Y)$. We get $t.X_d = t'.X_d$ and $t'_1.A' \neq t'_2.A'$ which is in contradiction with X_d is a determinant of A′. If $X_d \subseteq Y$ we have to recompute

Table 19. Propagation rules for unary operations on $T(S)$ returning table $T_r(S')$

Unary query on $T(\mathrm{S})$	Propagation rule for inferring the aggregable properties of attribute $\mathrm{A}' \in S_r$ in the result $T_r(S_r)$	*User action*
$\mathit{Filter}_T(P \mid \mathrm{Y})$	attribute $\mathrm{A}' \in S_r$ and $\mathrm{agg}_{\mathrm{A}'}(\mathrm{F}, \mathrm{X})$ holds in T: $\mathrm{agg}_{\mathrm{A}'}(\mathrm{F}, \mathrm{X}')$ holds in T_r with $\mathrm{X}' = \mathrm{X}$, $\mathrm{X}'_d = \mathrm{X}_d$ and $\mathrm{X}'_f = \mathrm{X}_f$.	*None*
$\mathit{Project}_T(\mathrm{Y}, f(\mathrm{Z}) \rightarrow \mathrm{M})$	dimension attribute $\mathrm{A}' \in \mathrm{Y}$ and $\mathrm{agg}_{\mathrm{A}'}(\mathrm{F}, \mathrm{X})$ holds in T: $\mathrm{agg}_{\mathrm{A}'}(\mathrm{F}, \mathrm{X}')$ holds in T_r with $\mathrm{X}' = \mathrm{X}$, $\mathrm{X}'_d = \mathrm{X}_d$ and $\mathrm{X}'_f = \mathrm{X}_f$.	*None*
	new measure attribute $\mathrm{A}' = \mathrm{M}$: if G can be applied on M as defined in Table 16 then $\mathrm{agg}_{\mathrm{A}'}(\mathrm{G}, \mathrm{X}')$ holds in T_r where X' is defined by the rules of Table 17 with X'_d = determinant of Z and $\mathrm{X}'_f = \emptyset$.	*Minimize* X'_d *Complete* X'_f
$\mathit{Pivot}_T(\mathrm{A} \mid \mathrm{Y})$	attribute $\mathrm{A}' \in S_r - \{\mathrm{A}\}$ and $\mathrm{agg}_{\mathrm{A}'}(\mathrm{F}, \mathrm{X})$ holds in T: if $\mathrm{X}_d \cap \mathrm{Y} = \emptyset$ then $\mathrm{agg}_{\mathrm{A}'}(\mathrm{F}, \mathrm{X}')$ holds in T_r with $\mathrm{X}' = \mathrm{X} - \mathrm{Y}$, $\mathrm{X}'_d = \mathrm{X}_d$ and $\mathrm{X}'_f = \mathrm{X}_f - \mathrm{Y}$.	*None*
	else $\mathrm{agg}_{\mathrm{A}'}(\mathrm{F}, \mathrm{X}')$ holds in T_r where X' is defined by the rules of Table 17 with X'_d = fact identifier and $\mathrm{X}'_f = \mathrm{X}_f - \mathrm{Y}$.	*Minimize* X'_d
	new pivot attribute $\mathrm{A}' \in S_r$ and $\mathrm{agg}_{\mathrm{A}}(\mathrm{F}, \mathrm{X})$ holds in T: if $\mathrm{X}_d \not\subseteq \mathrm{Y}$ then $\mathrm{agg}_{\mathrm{A}'}(\mathrm{F}, \mathrm{X}')$ holds in T_r with $\mathrm{X}' = \mathrm{X} - \mathrm{Y}$, $\mathrm{X}'_d = \mathrm{X}_d - \mathrm{Y}$ and $\mathrm{X}'_f = \mathrm{X}_f - \mathrm{Y}$	*None*
	else $\mathrm{agg}_{\mathrm{A}'}(\mathrm{F}, \mathrm{X}')$ holds in T_r where X' is defined by the rules of Table 17 with X'_d = fact identifier and $\mathrm{X}'_f = \mathrm{X}_f - \mathrm{Y}$	*Minimize* X'_d
$\mathit{Agg}_T(\mathrm{F}(\mathrm{A}) \mid \mathrm{Y})$	dimension attribute $\mathrm{A}' \in \mathrm{Y}$ and $\mathrm{agg}_{\mathrm{A}'}(\mathrm{F}, \mathrm{X})$ holds in T: $\mathrm{agg}_{\mathrm{A}'}(\mathrm{F}, \mathrm{X}')$ holds in T_r with $\mathrm{X}' = \mathrm{X} \cap \mathrm{Y}$ and $\mathrm{X}'_f = \mathrm{X}_f \cap \mathrm{Y}$	*None*
	new measure attribute $\mathrm{A}' = \mathrm{F}(\mathrm{A})$ and $\mathrm{agg}_{\mathrm{A}}(\mathrm{F}, \mathrm{X})$ holds in T: if G can be applied on A' as defined in Table 16 then $\mathrm{agg}_{\mathrm{F}(\mathrm{A})}(\mathrm{G}, \mathrm{X}')$ holds in T_r with $\mathrm{X}' = \mathrm{Y}$, X'_d = fact identifier and $\mathrm{X}'_f = \emptyset$.	*Minimize* X'_d *Complete* X'_f

the determinant set of all pivoted attributes. All attributes which were forbidden for A are also forbidden for A' and we obtain $\mathrm{X}'_f = \mathrm{X}_f - \mathrm{Y}$.

□

We next define the following proposition for attributes in the result of an aggregate query.

Proposition 2 (Propagation rule for aggregation)**.** Let $T(\mathrm{S})$ be an analytic table with dimension attributes $\mathrm{S}_D \subseteq \mathrm{S}$, and $\mathrm{agg}_{\mathrm{A}}(\mathrm{F}, \mathrm{X})$ be an aggregable property that holds in T with determinant X_d and forbidden set X_f. Let $T_r = \mathit{Agg}_T(\mathrm{F}(\mathrm{A}) \mid \mathrm{Y})$ be a valid aggregate query (i.e., $\mathrm{S}_D - \mathrm{X} \subseteq \mathrm{Y}$). Then the propagation rule of Table 19 for aggregation is correct.

Proof. We prove each case of attribute A':

(1) Every attribute $A' \in Y$ is a dimension attribute in T and T_r. For every aggregable property $\text{agg}_{A'}(F, X)$ that holds in T, we must only determine X' and X'_f for $\text{agg}_{A'}(F, X')$ in T_r. Since A' is aggregable along X, it is also aggregable along the subset of remaining attributes $X' = X \cap Y$ and all remaining forbidden attributes $X'_f = X_f \cap Y$ are still forbidden. We conclude that $\text{agg}_{A'}(F, X')$ holds in T_r where $X' = X \cap Y$ and $X'_f = X_f \cap Y$.
(2) We show that $\text{agg}_{F(A)}(G, X')$ holds for new attribute $F(A)$ in T_r with $X' = X'^{+}_d - X'_f = Y$. By the assumption in Item 1 of Definition 9, G is applicable to $F(A)$. By applying the default rules of Table 17 X'_d is the fact identifier of T_r and determines $F(A)$ as well as all attributes in Y ($Y = X'^{+}_d$ is the closure of X_d). X'_f is by default empty. X'_d and X'_f and must be validated by the user by removing incorrect attributes from X'_d and adding meaningless attributes to X_f.

□

Example 13. Consider fact table PRODUCT_LIST in Table 20 and attribute QTY. As seen in Example 11, $\text{agg}_{\text{QTY}}(\text{SUM} \mid X)$ holds in PRODUCT_LIST for $X = \{\text{PROD\_SKU, YEAR}\}$, $X_d = \{\text{PROD\_SKU, YEAR}\}$ and $X_f = \emptyset$.

Table 20. Table PRODUCT_LIST

PROD_SKU	BRAND	COUNTRY	YEAR	QTY
cz-tshirt-s	Coco Cola	USA	2017	5 000
cz-tshirt-s	Coco Cola	USA	2018	7 000
cz-tshirt-s	Zora	Spain	2017	5 000
cz-tshirt-s	Zora	Spain	2018	7 000
coco-can-33cl	Coco Cola	USA	2017	10 000

First, let $T_r = \mathit{Filter}_{\text{PRODUCT\_LIST}}(\{\text{YEAR} = \text{'2017'}\})$. By Table 19, $\text{agg}_{\text{QTY}}(\text{SUM} \mid X)$ still holds in T_r. Next, let $T_r = \mathit{Pivot}_{\text{PRODUCT\_LIST}}(\text{QTY} \mid \text{BRAND})$ be a query producing two new attributes QTY_COCOCOLA and QTY_ZORA with values from attribute QTY. Then, by Table 19, since $X_D \not\subseteq Y$, both aggregable properties $\text{agg}_{\text{QTY\_COCOCOLA}}(\text{SUM} \mid X')$ and $\text{agg}_{\text{QTY\_ZORA}}(\text{SUM} \mid X')$ hold in T_r where $X' = X - \{\text{BRAND}\} = X = \{\text{PROD\_SKU, YEAR}\}$. Finally, let $T_r = \mathit{Agg}_{\text{PRODUCT\_LIST}}(\text{SUM(QTY)} \mid Y)$ with $Y = \{\text{BRAND, YEAR}\}$. Function SUM returns a value of category NUM, so by Table 19, $\text{agg}_{\text{SUM(QTY)}}(G \mid X')$ holds in table T_r with $X' = Y = \{\text{BRAND, YEAR}\}$ and $G \in \{\text{SUM, AVG, COUNT, COUNT\_DISTINCT, MIN, MAX}\}$. By default, $X'_d = \{\text{BRAND, YEAR}\}$ is the fact identifier of T_r and X_f is empty.

Let us now consider attribute PROD_SKU. As seen in Example 11, properties $\text{agg}_{\text{PROD\_SKU}}(\text{COUNT} \mid X)$ and $\text{agg}_{\text{PROD\_SKU}}(\text{COUNT\_DISTINCT} \mid X)$ hold in PRODUCT_LIST for $X = \{\text{BRAND, COUNTRY, YEAR}\}$. Let $T_r = \mathit{Agg}_{\text{PRODUCT\_LIST}}(F(\text{PROD\_SKU}) \mid \{\text{BRAND, YEAR}\})$, with F = COUNT or F = COUNT_DISTINCT.

Both of these functions return values of category NUM. So by Table 19, $\text{agg}_{F(\text{PROD\_SKU})}(G \mid X')$ holds in table T_r for $G \in \{\text{SUM, AVG, COUNT, COUNT\_DISTINCT, MIN, MAX}\}$ and $X' = X \cap \{\text{BRAND, YEAR}\} = \{\text{BRAND, YEAR}\}$.

3.3.2 *Propagating aggregable properties to the results of binary operations.*

We now consider the problem of determining the aggregable properties of the attributes in the result of binary merge queries and binary set queries (union, difference). The propagation rules are summarized in Table 21

Proposition 3 (propagation rule for merge)**.** Let $T(S)$ and $T'(S')$ be two analytic tables with dimension attributes $S_D \subseteq S$ and $S'_D \subseteq S'$ respectively. Let $T_r(S_r)$ be the result of a merge query between T and T' over a set of common

Table 21. Propagation rules for binary operations

Binary query on $T(S)$ and $T'(S')$	Propagation rule for inferring the aggregable properties of attribute $\mathrm{A}' \in S$ in the result $T_r(\mathrm{S}_r)$	*User action*
$T_r = T \rtimes_{\mathrm{Y}} T'$ $T_r = T \bowtie\!\!\!\!\subset_{\mathrm{Y}} T'$	dimension attribute $\mathrm{A}' \in S_D$ and $\mathrm{agg}_{\mathrm{A}'}(\mathrm{F}, \mathrm{X})$ holds in T: $\mathrm{agg}_{\mathrm{A}'}(\mathrm{F}, \mathrm{X}')$ holds in T_r with $\mathrm{X}' = \mathrm{X} \cup (S'_D - \mathrm{Y}) - \mathrm{X}'_f$ and $\mathrm{X}'_f = \mathrm{X}_f$.	*Complete* X'_f
$T_r = T \rtimes\!\!\!\subset_{\mathrm{Y}} T'$ $T_r = T \bowtie_{\mathrm{Y}} T'$	measure attribute $\mathrm{A}' \in S - S_D$ and $\mathrm{agg}_{\mathrm{A}'}(\mathrm{F}, \mathrm{X})$ holds in T : $\mathrm{agg}_{\mathrm{A}'}(\mathrm{F}, \mathrm{X}')$ holds in T_r with $\mathrm{X}' = \mathrm{X}$, $\mathrm{X}'_d = \mathrm{X}_d$ and $\mathrm{X}'_f = \mathrm{X}_f$	*Complete* X'_f
$T_r = T \cup T'$	dimension attr. $\mathrm{A}' \in S_r$ and $\mathrm{agg}_{\mathrm{A}'}(\mathrm{F}, \mathrm{X})$ holds in T and T' $\mathrm{agg}_{\mathrm{A}'}(\mathrm{F}, \mathrm{X}')$ holds in T_r with $\mathrm{X}' = \mathrm{X}$, $\mathrm{X}'_d = \mathrm{X}_d$ and $\mathrm{X}'_f = \mathrm{X}_f$.	*None*
	measure attr. $\mathrm{A}' \in S_r$ and $\mathrm{agg}_{\mathrm{A}'}(\mathrm{F}, \mathrm{X})$ holds in T and T': if $\mathrm{X}_d \mapsto \mathrm{A}'$ holds in T_r then $\mathrm{agg}_{\mathrm{A}'}(\mathrm{F}, \mathrm{X}')$ holds in T_r with $\mathrm{X}' = \mathrm{X}$, $\mathrm{X}'_d = \mathrm{X}_d$ and $\mathrm{X}'_f = \mathrm{X}_f$.	*None*
	else $\mathrm{agg}_{\mathrm{A}'}(\mathrm{F}, \mathrm{X}')$ holds in T_r where X' is defined by the rules of Table 17 with X'_d = fact identifier and $\mathrm{X}'_f = \mathrm{X}_f$.	*Minimize* X'_d
$T_r = T - T'$	attribute $\mathrm{A}' \in S_r$ and $\mathrm{agg}_{\mathrm{A}'}(\mathrm{F}, \mathrm{X})$ holds in T and T' : $\mathrm{agg}_{\mathrm{A}'}(\mathrm{F}, \mathrm{X}')$ holds in T_r with $\mathrm{X}' = X'^{+}_d - X'_f$ where X'^{+}_d is the set of attributes in $S_D \cup S'_D$ literally determined by X'_d with $\mathrm{X}'_d = \mathrm{X}_d$ and $\mathrm{X}'_f = \mathrm{X}_f$.	*None*

dimension attributes $\mathrm{Y} \subseteq S_D \cap S'_D$ and let $S^r_D = S_D \cup S'_D$ be the dimension attributes in T_r. Let $\mathrm{A}' \in S_r$ be an attribute of T with aggregable property $\mathrm{agg}_{\mathrm{A}'}(\mathrm{F}, \mathrm{X})$ holding in T.

Then the propagation rules of Table 21 for merge queries are correct.

Proof. We proceed with each case of attribute A':

(1) If A' is a dimension attribute, by Definition 9, $\mathrm{X} = S_D - \{\mathrm{A}'\} - \mathrm{X}_f$ and since $\{\mathrm{A}'\} \cup \mathrm{X}_f \subseteq S_D$ (all forbidden attributes are in S_D), we can add all attributes of S'_D which are not in S_D to X': $\mathrm{X}' = \mathrm{X} \cup (S'_D - \mathrm{Y})$. The user must add all new meaningless attributes in S'_D to $\mathrm{X}'_f = \mathrm{X}_f$.

(2) If A' is a measure attribute, A' is an attribute of table T but not of table T'. Let $\mathrm{X} = \mathrm{X}^{+}_d - \mathrm{X}_f$, as in Definition 9. We show that the LFD $\mathrm{X}_d \mapsto \mathrm{A}'$ is still valid in T_r. Suppose that there exist two tuples t and t' in T_r such that $t.X_d \equiv t'.X_d$ and $t.\mathrm{A}' \not\equiv t'.\mathrm{A}'$. We show that, for each merge operation, the projection of these two tuples on S would also exist in T, which contradicts that LFD $\mathrm{X}_d \mapsto \mathrm{A}'$ holds in T:
   - If $T_r = T \rtimes_{\mathrm{Y}} T'$ (left merge): by definition of $\rtimes_{\mathrm{Y}}$ any projection on S of a tuple T in T_r is also a tuple in T (similar to filter queries).
   - If $T_r = T \bowtie\!\!\!\!\subset_{\mathrm{Y}} T'$ (right merge): any projection on S of a tuple T in T_r is either a tuple in T or a tuple that does not exist in T and has a null value for each attribute in $S - \mathrm{Y}$. In the latter case, the LFD is preserved in T_r because all these tuples also have a null value on A and there is no tuple in T that has null values on X_d and a non-null value for A.

- If $T_r = T \rtimes\!\!\!\subset_{\mathrm{Y}} T'$ (full merge): The proof as a combination of the proofs for left merge and right merge (any projection on S of a tuple T in T_r is either a tuple in T or a tuple that does not exist in T and has a null value for each attribute in $S - \mathrm{Y}$).
- If $T_r = T \bowtie_{\mathrm{Y}} T'$ (strict merge): any projection on S of a tuple T in T_r is also a tuple in T.

Then X_d is still a minimum set of dimension attributes in T_r which literally determines A' (otherwise it would not be minimum in T) and, by Definition 9, X' contains all attributes determined by X_d in T_r (closure of X_d in T_r). Similarly, all meaningless attributes in X_f remain meaningless and the user can add new meaningless attributes from $S' - \mathrm{Y}$.

□

In Table 21, in the case of a merge query, $\mathrm{X}'_f = \mathrm{X}_f$ by default and for a measure attribute, the attribute graphs of each dimension are used to compute the closure X^+_d over S'_D. In the propagation rule, we only considered the case of an attribute A in T, but the same result would apply for an attribute A in T' due to the symmetry of the merge operations.

Example 14. In Table 22, table STORE_SALES_YEARLY is defined over dimensions *SALESORG* and *TIME*, table DEM2 is defined over *REGION* (see Table 1b) and *TIME*, and table STORE_SALES_DEM is the result of the left-merge query:

$$\text{STORE\_SALES\_DEM} = \text{STORE\_SALES\_YEARLY} \rtimes\!\!\!\subset_{\text{CITY,YEAR}} \text{DEM2}$$

We first consider attribute AMOUNT. Suppose that $\mathrm{agg}_{AMOUNT}(\mathrm{SUM}, \mathrm{X})$ holds in STORE_SALES_YEARLY for $\mathrm{X} = \{$STORE_ID, CITY, YEAR$\}$, with $\mathrm{X}_d = \{$STORE_ID, YEAR$\}$ and $\mathrm{X}_f = \emptyset$. Then, $\mathrm{Z} = \{$CITY, STATE, COUNTRY$\}$ is the set of attributes determined by X_d in DEM2. Then, $\mathrm{X}'^+_d = \mathrm{X}^+_d \cup \mathrm{Z} = \{$STORE_ID, YEAR, CITY, STATE, COUNTRY$\}$ is the set of attributes determined by X_d in STORE_SALES_DEM. So, by the propagation rule of Table 21, aggregable property $\mathrm{agg}_{\text{AMOUNT}}(\mathrm{SUM}, \mathrm{X}')$ holds in STORE_SALES_DEM where $\mathrm{X}' = \mathrm{X}'^+_d - \mathrm{X}'_f = \mathrm{X}'^+_d$.

Observe that the aggregable property computed by the propagation rule for merge queries does not guarantee that any aggregation query of AMOUNT produces the same result when it is applied on STORE_SALES_YEARLY or on STORE_SALES_DEM. We will show in Section 4.2 how to refine the propagation rule to guarantee this summarizability property.

We now consider attribute UNEMP in DEM2 of category STAT. Suppose that $\mathrm{agg}_{\text{UNEMP}}(\mathrm{F}, \mathrm{X})$ holds in DEM2 for $\mathrm{X} = \{$YEAR, CITY, STATE, COUNTRY$\}$) and $\mathrm{F} \in \{\mathrm{COUNT}, \mathrm{COUNT\_DISTINCT}, \mathrm{MIN}, \mathrm{MAX}\}$ with $\mathrm{X}_d = X$. We use the propagation rule of Table 21 on the (equivalent) right-merge of DEM2 with STORE_SALES_YEARLY. No new attribute of STORE_SALES_YEARLY is determined by X_d, so $\mathrm{X}'^+_d = \mathrm{X}_d$. Hence, by the propagation rule, $\mathrm{agg}_{\text{UNEMP}}(\mathrm{F}, \mathrm{X}')$ holds in STORE_SALES_DEM for $\mathrm{X}' = \mathrm{X}'^+_d - \mathrm{X}'_f = \mathrm{X}_d = \mathrm{X}$.

The next proposition states which aggregable properties hold for attribute A' in the union $T_r = T \cup T'$ and the set difference $T_r = T - T'$, knowing the aggregable properties of A' in tables T and T'.

Proposition 4 (propagation rules for union and difference)**.** Let $T(S)$ and $T'(S)$ be two tables over the same schema S having a set of dimension attributes X. Let A' $\in S$ be an attribute with aggregable property $\mathrm{agg}_{\mathrm{A}'}(\mathrm{F}, \mathrm{X})$ *holding in both tables* T *and* T'. Then the propagation rules of Table 21 for set union and difference are correct.

Proof. We distinguish each case where A' is a dimension attribute or a measure attribute in $T_r(S_r)$:

Table 22. Example of a left-merge query

(a) STORE_SALES_YEARLY

STORE_ID	CITY	YEAR	AMOUNT
Oh_01	Dublin	2017	3.2
Ca_01	Dublin	2017	5.3
Oh_01	Dublin	2018	8.2
Ca_01	Dublin	2018	6.3
Pa_01	Paris	2017	45.1

(b) DEM2

CITY	STATE	COUNTRY	YEAR	UNEMP
Dublin	Ohio	USA	2017	4.2
Dublin	California	USA	2017	3.1
Palo Alto	California	USA	2017	2.1
Paris	-	France	2017	11.9
Dublin	-	Ireland	2017	6.7

(c) STORE_SALES_DEM (left merge of STORE_SALES_YEARLY and DEM2)

STORE_ID	CITY	STATE	COUNTRY	YEAR	AMOUNT	UNEMP
Oh_01	Dublin	Ohio	USA	2017	3.2	4.2
Oh_01	Dublin	California	USA	2017	3.2	3.1
Oh_01	Dublin	-	Ireland	2017	3.2	6.7
Ca_01	Dublin	California	USA	2017	5.3	3.1
Ca_01	Dublin	Ohio	USA	2017	5.3	4.2
Ca_01	Dublin	-	Ireland	2017	5.3	6.7
Oh_01	Dublin	-	-	2018	8.2	-
Ca_01	Dublin	-	-	2018	6.3	-
Pa_01	Paris	-	France	2017	45.1	11.9

(1) If A' is a dimension attribute: let $\mathrm{X}' = \mathrm{S}_D - \{A\} - \mathrm{X}_f$, where S_D is the set of dimension attributes in S. Since $S_r = \mathrm{S}$ for difference and union, and X_f is defined for a given set of attributes (independently of a specific table), X' does not change in the aggregable property of T_r for A'.
(2) If A' is a measure attribute: let $\mathrm{Z} = \mathrm{X}_d^+ - \mathrm{X}_f$. Then $\mathrm{X}_d \mapsto \mathrm{A}'$ in each table T and T'. We analyze each query case:
   - Difference: By definition of analytic difference queries, $T_r \subseteq T$, thus $\mathrm{X}_d \mapsto \mathrm{A}'$ also holds in $T_r = T - T'$.
   - Union: there could be two tuples that have the same values on their X attributes, so we must check that $\mathrm{X}_d \mapsto A$ holds in $T_r = T \cup T'$. Otherwise, we apply the default rules for initializing X'_d.

   Finally, X_f must be the same in T_r since it only depends on the schema.

□

Propagation rules for all operations, except union, are immediate to compute because they only involve the manipulation of metadata properties. In the case of union, when a measure depends on a subset of the dimensions of a fact table, the propagation rule requires a uniqueness test (to check the LFD $\mathrm{X}_d \mapsto A$) on the result of the union. Since the uniqueness test involves hierarchical dimension attributes only, it can be performed efficiently using specific data structures used to represent fact tables in main-memory (see [6, 7]).

4 SUMMARIZABILITY OF AGGREGABLE ATTRIBUTES

In this section, we consider the properties of attributes that characterize the equivalence between computing an aggregated value of an attribute from a table T and computing the same aggregated value from the result of a query Q over T. In Section 4.1, we first define the property of *summarizable attributes* in the case when Q is an aggregate query, which corresponds to the traditional notion of summarizability addressed by previous work. In Section 4.2, we then

extend our propagation rules to compute aggregable properties such that aggregate queries can only be expressed over summarizable attributes. Finally, in Section 4.3, we introduce the new property of *G-summarizability* of attributes in the case when Q is any analytic query. In Section 4.4, we then again extend our propagation rules to compute aggregable properties such that aggregation queries are expressed over G-summarizable attributes only.

4.1 Summarizable attributes and distributive functions

Figure 9 illustrates the definition of summarizable attributes. It depicts that when some attribute A of table T is aggregated with some function F for each partition of attributes Z_2, it is possible to obtain the same result, by first aggregating A for each partition of attributes Z_1 where $Z_2 \subset Z_1$, using function F, and then further aggregating A for each partition Z_2, using either the same function F or a different function G. We shall say that A is summarizable with respect to grouping set Z_1 and function F, using function G.

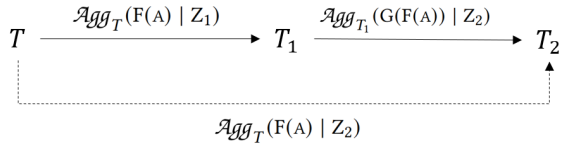


Fig. 9. Summarizable attribute A with respect to Z_1 and F using G

The Definition 10 below formalizes Figure 9 and, as shown later in Section 5, subsumes the definition of summarizability addressed in previous work. For simplicity, we use the expression *valid aggregate query* for a query $\mathcal{A}gg_T(F(A) \mid Z)$ such that there exists an aggregable property $agg_A(F, X)$ that holds in T, and Z contains the dimension attributes of T that are not in X.

Definition 10 (Summarizable attribute)**.** Let $T(S)$ be an analytic table, A be an aggregable attribute of T, and $T_1 = \mathcal{A}gg_T(F(A) \mid Z_1)$ be a valid aggregate query over T. If for *any* subset $Z_2 \subset Z_1$, there exists an applicable aggregate function G such that the equation

$$\mathcal{A}gg_{T_1}(G(F(A)) \mid Z_2) = \mathcal{A}gg_T(F(A) \mid Z_2) \tag{1}$$

holds, then A is said to be *summarizable in T with respect to grouping set Z_1 and function F using function G*.

Attribute summarizability is related to the notion of distributive aggregation functions, also called decomposable aggregation functions in [22].

Definition 11 (Distributive aggregation function)**.** Let F be an aggregation function applicable to a set of domain values V and $P = \{V_1, \ldots, V_n\}$, $n \geq 1$ be a partitioning of V. If there exists an aggregate function G such that $F(V_1 \cup \ldots \cup V_n) = G(F(V_1) \cup \ldots \cup F(V_n))$ then F is said to be *distributive on partitioning P* using function G.

If F is distributive using function G on any partitioning of domain V, we say that F is distributive using function G over domain V. If F is distributive using function G over any domain V, we say that F is distributive using function G, and if F = G, we simply say that F is distributive.

It is easy to show that functions SUM, MIN and MAX are distributive, function COUNT is distributive using function SUM whereas function COUNT_DISTINCT is distributive using function SUM only on partitionings

where the same value does not appear in two distinct partitions. Finally, function AVG is distributive over all domains V containing only two elements or where all elements are identical.

Example 15. For $V = \{1, 2, 2, 3\}$ we have COUNT_DISTINCT(V) = 3. Then COUNT_DISTINCT is distributive using SUM on partitioning $P' = \{\{1, 2, 2\}, \{3\}\}$:

$$\text{SUM}(\text{COUNT\_DISTINCT}(\{1, 2, 2\}), \text{COUNT\_DISTINCT}\{3\})) = 3$$

However, COUNT_DISTINCT is not distributive using SUM on partitioning $P = \{\{1, 2\}, \{2, 3\}\}$:

$$\text{SUM}(\text{COUNT\_DISTINCT}(\{1, 2\}), \text{COUNT\_DISTINCT}(\{2, 3\}) = \text{SUM}(2, 3) = 4$$

We say that F is *distributive using function* G *on attribute* A *of table* T *with partitioning attributes* Z if F is distributive using function G on all partitions of the values of A in T defined by Z and any subset of Z. The following proposition relates the definition of distributive functions to the notion of summarizable attributes.

Proposition 5 (Function distributivity and attribute summarizability)**.** Let T(S) be an analytic table with dimension attributes $S_D \subseteq S$ and an aggregable attribute A such that $\text{agg}_A(F, X)$ holds in T. If F is distributive using function G on attribute A in table T with partitioning attributes $Z \supseteq S_D - X$ then A is summarizable with respect to grouping set Z and function F using function G.

Proof. Suppose that $\text{agg}_A(F, X)$ holds in T and $T_1 = \mathcal{A}gg_T(F(A) \mid Z_1)$ and F is *distributive using function* G *on attribute* A *of table* T *with partitioning attributes* Z_1. To prove that A is summarizable in T with respect to grouping set Z_1 and F using function G, we prove that for any subset $Z_2 \subset Z_1$, the following equation holds:

$$\mathcal{A}gg_T(F(A) \mid Z_2) = \mathcal{A}gg_{T_1}(G(F(A)) \mid Z_2) \tag{2}$$

First, it is obvious that both tables T and T_1 contain the same Z_2 values and therefore, the result tables in Eq. (2) contain the same tuples with distinct Z_2 values. We now show that for each pair of tuples $t \in \mathcal{A}gg_T(F(A) \mid Z_2)$ and $t' \in \mathcal{A}gg_{T_1}(G(F(A)) \mid Z_2)$ where $t.Z_2 = t'.Z_2$, we have $t.F(A) = t'.G(F(A))$. Let $x = t.Z_2 = t'.Z_2$ and $T^x = \sigma_{Z_2=t.Z_2}(T)$ and $T_1^x = \sigma_{Z_2=t.Z_2}(T_1)$ be the partitions of T and T_1 on attributes Z_2 corresponding to the tuples used to compute $t.F(A)$. For each tuple $t'_i \in T_1^x$ there also exists a partition $T^{y_i} = \sigma_{Z_1=y_i}(T)$ of T where $y_i = t'_i.Z_1$ and $t'_i.F(A) = F(\pi_A(T^{y_i}))$. All tuples t'_i have the same Z_2 value $x = t.Z_2$ and are aggregated to tuple t' whose value for attribute $t'.G(F(A)) = G(F(\pi_A(T^{y_1}) \cup \ldots \cup F(\pi_A(T^{y_n})$. Since F is distributive using function G and $T^x = T^{y_1} \cup \ldots \cup T^{y_n}$, we obtain $G(F(\pi_A(T^{y_1}) \cup \ldots \cup F(\pi_A(T^{y_n})) = F(\pi_A(T^{y_1}) \cup \ldots \cup \pi_A(T^{y_n})) = F(\pi_A(T^x))$. We conclude $t.Z_2 = t'.Z_2$ and $t.F(A) = t'.G(F(A))$. □

Example 16. Function COUNT is distributive using function SUM. Therefore, PROD_SKU in table PRODUCT_LIST (Table 15 on Page 21) is summarizable with respect to grouping set Z = {BRAND, COUNTRY, YEAR} and COUNT using function SUM. Thus, if Z_2 = {COUNTRY, YEAR} and $T_1 = \mathcal{A}gg_{PRODUCT\_LIST}(\text{COUNT}(\text{PROD\_SKU}) \mid Z)$, the following equation folds:

$$\mathcal{A}gg_{PRODUCT\_LIST}(\text{COUNT}(\text{PROD\_SKU}) \mid Z_2) = \mathcal{A}gg_{T_1}(\text{SUM}(\text{COUNT}(\text{PROD\_SKU})) \mid Z_2)$$

However, as explained before, COUNT_DISTINCT is only distributive using SUM if no pair of partitions share the same value. This is not the case (there exist two partitions of Z with the same product "cz-tshirt-s"), so attribute PROD_SKU is not summarizable with respect to grouping set Z and function COUNT_DISTINCT.

Function distributivity is a sufficient but not a necessary condition for summarizability. This is illustrated in the following proposition, which defines a sufficient condition for summarizability with COUNT_DISTINCT and SUM.

Proposition 6 (Summarizability with COUNT_DISTINCT and SUM)**.** Let $T(\mathrm{S})$ be an analytic table with a set of dimension attributes S_D and an aggregable attribute A. Let $T_1 = \mathcal{A}gg_T(\mathrm{COUNT\_DISTINCT}(\mathrm{A}) \mid \mathrm{Z}_1)$ be a valid aggregate query over T, where $\mathrm{Z}_1 \subseteq \mathrm{S}_D$. If $\mathrm{Z}_2 \subset \mathrm{Z}_1$ and the literal functional dependency $\mathrm{Z}_2 \cup \{\mathrm{A}\} \mapsto \mathrm{Z}_1$ holds in T, the following equation is true:

$$\mathcal{A}gg_{T_1}(\mathrm{SUM}(\mathrm{COUNT\_DISTINCT}(\mathrm{A})) \mid \mathrm{Z}_2) = \mathcal{A}gg_T(\mathrm{COUNT\_DISTINCT}(\mathrm{A}) \mid \mathrm{Z}_2) \tag{3}$$

We say that attribute A (in T) is summarizable with respect to grouping set Z_1 and COUNT_DISTINCT using function SUM *with partitioning attributes* Z_2.

Proof. The previous proposition mainly states that A is summarizable with respect to Z_1 and COUNT_DISTINCT using function SUM with partitioning attributes Z_2 if all tuples T in some partition $T^x \subseteq T$ defined by attributes $\mathrm{Z}_2 \subseteq \mathrm{Z}_1$ which have the same value for attribute $t.\mathrm{A}$ are assigned to the *same partition* $T^y \subseteq T$ defined by attributes Z_1. This avoids the double counting of distinct A values when taking the SUM of COUNT_DISTINCT over the partitions generated by attributes Z_1.

We first show by contradiction that when $\mathrm{Z}_2 \cup \{\mathrm{A}\} \mapsto \mathrm{Z}_1 - \mathrm{Z}_2$ holds in T, all tuples T in some partition $T^x \subseteq T$ of T generated by attributes Z_2 with the same value for attribute $t.\mathrm{A}$ are assigned to the same partition $T^y \subseteq T$ generated by attributes Z_1.

Let T^x be a partition of T which contains all tuples T such that $t.\mathrm{Z}_2 = x$. Since $\mathrm{Z}_2 \subset \mathrm{Z}_1$, T^x is the union of a set of partitions $T_0^y \ldots, T_n^y$, $n \geq 0$ of T defined by attributes Z_1. Suppose that there exist two tuples $t \in T_i^y$ and $t' \in T_j^y$ where $i \neq j$ and $t.\mathrm{A} = t'.\mathrm{A}$. Then, we have $t.\mathrm{Z}_2 = t'.\mathrm{Z}_2 = x$, $t.\mathrm{A} = t'.\mathrm{A}$ and, since $i \neq j$, $t.\mathrm{Z}_1 \neq t'.\mathrm{Z}_1$ (two different partitions generated by Z_1 contain the same values for Z_2 and A). This is in contradiction with $\mathrm{Z}_2 \cup \{\mathrm{A}\} \mapsto \mathrm{Z}_1 - \mathrm{Z}_2$. Then, if d_i is the number of distinct A values in some partition $T_i^x \subseteq T$, we can easily show that $\sum_{i=0}^{n} d_i$ is the number of distinct A values in partition V. □

Observe that if Eq. (3) holds for any subset $\mathrm{Z}_2 \subset \mathrm{Z}_1$, A is summarizable with respect to grouping set Z_1 and COUNT_DISTINCT using function SUM (Definition 10).

Example 17. Let $T_1 = \mathcal{A}gg_{\mathrm{PRODUCT\_LIST}}(\mathrm{COUNT\_DISTINCT}(\mathrm{PROD\_SKU}) \mid \mathrm{Z}_1)$ where $\mathrm{Z}_1 = \{\mathrm{BRAND}, \mathrm{COUNTRY}\}$. Attribute PROD_SKU is not summarizable with respect to grouping set Z_1 and function COUNT_DISTINCT using SUM. However, if $\mathrm{PROD\_SKU} \mapsto \mathrm{COUNTRY}$ holds in table PRODUCT_LIST, then for $\mathrm{Z}_2 = \{\mathrm{BRAND}\}$ and $\mathrm{A} = \mathrm{PROD\_SKU}$, we have $\{\mathrm{PROD\_SKU}, \mathrm{BRAND}\} \mapsto \{\mathrm{BRAND}, \mathrm{COUNTRY}\}$. Therefore, PROD_SKU is summarizable with respect to Z_1 and COUNT_DISTINCT using function SUM with partitioning attribute $\mathrm{Z}_2 = \{\mathrm{BRAND}\}$.

4.2 Controlling attribute summarizability using aggregable properties

Given the result of an aggregate query $T_1 = \mathcal{A}gg_T(\mathrm{F}(\mathrm{A}) \mid \mathrm{Z}_1)$ over some attribute A, we want to control the possible aggregations of attribute $\mathrm{F}(\mathrm{A})$ depending on the summarizability of A. We use aggregable properties for that purpose, as shown on Figure 10. We want to automatically compute the subset of dimension attributes $\mathrm{X}' \subseteq \mathrm{Z}_1$ of T_1 such that $\mathrm{agg}_{\mathrm{F}(\mathrm{A})}(\mathrm{G}, \mathrm{X}')$ holds in T_1, for some function G that is applicable to $\mathrm{F}(\mathrm{A})$, and which guarantees the summarizability of A in T_1 for any Z_2 such that $\mathrm{Z}_1 - \mathrm{X}' \subseteq \mathrm{Z}_2$. Our rationale is therefore to refine the propagation rules introduced in Section 3.3 to take into account the summarizability correctness criteria.

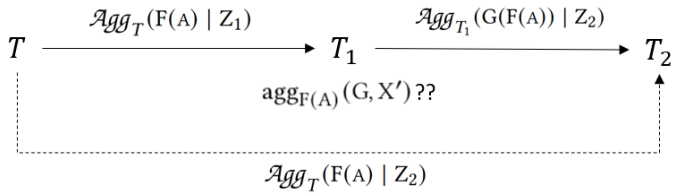


Fig. 10. Aggregable property that controls the summarizability of A in T with respect to grouping set Z_1

We next define the notion of summarizability preserving aggregable property which formalizes Figure 10.

Definition 12 (Summarizability preserving aggregable property)**.** Let $T(\mathrm{S})$ be an analytic table and $T_1 = \mathcal{A}gg_T(\mathrm{F(A)} \mid \mathrm{Z}_1)$ be the result of a valid aggregate query. We say that $\mathrm{agg}_{\mathrm{F(A)}}(\mathrm{G}, \mathrm{X}')$ *preserves the summarizability of* A *with respect to grouping set* Z_1 if for any subset Z_2 such that $\mathrm{Z}_1 - \mathrm{X}' \subseteq \mathrm{Z}_2$, attribute A is summarizable in T with respect to grouping set Z_2 and function F using G.

The next proposition uses Proposition 5 and Proposition 6 to refine the previous propagation rule for aggregation in Table 19 so that summarizability preserving aggregable properties are inferred. The refined rule for the case when $\mathrm{A}' = \mathrm{F}(A)$ is displayed in Table 23.

Table 23. Propagation rule for aggregate operation preserving summarizability

Query on $T(\mathrm{S})$	New propagation rule for inferring the aggregable properties of new measure attribute A′ in the result $T_r(S_r)$	*User action*
$\mathcal{A}gg_T(\mathrm{F(A)} \mid \mathrm{Y})$	new measure attribute $\mathrm{A}' = \mathrm{F(A)}$ and $\mathrm{agg}_{\mathrm{A}}(\mathrm{F}, \mathrm{X})$ holds in T: if G can be applied on A′ (Table 16) and F is distributive using G then $\mathrm{agg}_{\mathrm{F(A)}}(\mathrm{G}, \mathrm{X}')$ holds in T_r with $\mathrm{X}' = \mathrm{X} \cap \mathrm{Y}$, X'_d = fact identifier in T_r and $\mathrm{X}'_f = \emptyset$. if F = COUNT_DISTINCT and X′ is a *maximal* subset of $\mathrm{X} \cap \mathrm{Y}$ such that $(\mathrm{Y} - \mathrm{X}') \cup \{A\} \mapsto \mathrm{Y}$ holds in T then $\mathrm{agg}_{\mathrm{F(A)}}(\mathrm{SUM}, \mathrm{X}')$ holds in T_r with X'_d = fact identifier in T_r and $\mathrm{X}'_f = \emptyset$.	*Minimize* X'_d *Complete* X'_f *Minimize* X'_d *Complete* X'_f

Proposition 7 (Propagation of aggregable properties with summarizability preservation)**.** Let $T(\mathrm{S})$ be an analytic table with dimension attributes $\mathrm{S}_D \subseteq \mathrm{S}$ and let $T_r = \mathcal{A}gg_T(\mathrm{F(A)} \mid \mathrm{Y})$ be the result of a valid aggregate query. Then the aggregable properties inferred by the rule of Table 23 when $\mathrm{A}' = F(A)$ preserve the summarizability of A with respect to grouping set Y.

Proof. By Definition 12, we have to show that for any subset $\mathrm{Z}_2 \subseteq \mathrm{Y}$ such that $\mathrm{Y} - \mathrm{X}' \subseteq \mathrm{Z}_2$, attribute A is summarizable in T with respect to Z_2 and function F using G. We examine both cases of Proposition 7:

- G can be applied on A′ as defined in Table 16 and F is distributive using G: Since F is distributive using function G, it is also distributive on attribute A with partitioning attributes Y. Then, by Proposition 5, A is summarizable with respect to Y and F using function G, and Equation (1) in Definition 10 holds for any subset $\mathrm{Z}_2 \subseteq \mathrm{Y}$.

- F = COUNT_DISTINCT and X' is a *maximal* subset of $X \cap Y$ such that $(Y - X') \cup \{A\} \mapsto Y$ holds in T: By Proposition 6, it is sufficient to show that $Z_2 \cup \{A\} \mapsto Y$ for all Z_2 where $Y - X' \subseteq Z_2 \subset X \cap Y$. Since $(Y - X') \cup \{A\} \mapsto Y$ and $Y - X' \subseteq Z_2$ we also have $Z_2 \cup \{A\} \mapsto Y$.

□

For the second condition in Table 23, observe that there might exist several maximal subsets of attributes X'_i. The process to compute these subsets is quite simple. Each maximal subset X'_i corresponds to a *minimal subset* of attributes $K_i = Y - X'_i \subset Y$ such that $K_i \cup \{A\}$ determines all attributes of Y. These sets K_i can easily be computed using the attribute graphs of the corresponding dimensions to obtain $X'_i = (Y \cap X) - K_i$.

Example 18. Aggregable property $\text{agg}_{\text{PROD\_SKU}}(\text{COUNT} \mid X)$ holds in table PRODUCT_LIST (Table 8e) for $X = \{\text{BRAND, COUNTRY, YEAR}\}$. Let $T_r = \mathcal{A}gg_{\text{PRODUCT\_LIST}}(\text{COUNT}(\text{PROD\_SKU}) \mid Y)$, where $Y = \{\text{BRAND, COUNTRY, YEAR}\}$.

By the first condition in Table 23 and distributivity of COUNT using SUM, the aggregable property $\text{agg}_{\text{COUNT(PROD\_SKU)}}(\text{SUM} \mid X')$, where $X' = X \cap Y = \{\text{BRAND, COUNTRY, YEAR}\}$, preserves the summarizability of PROD_SKU with respect to grouping set Y.

Example 19. Property $\text{agg}_{\text{PROD\_SKU}}(\text{COUNT\_DISTINCT} \mid X)$ holds in table PRODUCT_LIST for $X = \{\text{BRAND, COUNTRY, YEAR}\}$. Let $T_r = \mathcal{A}gg_{\text{PRODUCT\_LIST}}(\text{COUNT\_DISTINCT}(\text{PROD\_SKU}) \mid Y)$ where $Y = \{\text{BRAND, COUNTRY, YEAR}\}$. By the second condition in Table 23, we must compute the maximum subset X' of $X \cap Y$ such that $(Y - X') \cup \{A\} \mapsto Y$. We use the method explained before. By the attribute graphs of dimension *TIME* and *PROD*, the only LFD which holds among the attributes of Y is BRAND $\mapsto$ COUNTRY. Thus, there is a single minimal set $K = \{\text{BRAND, YEAR}\}$ such that $K \cup \{\text{PROD\_SKU}\}$ determines all attributes in Y. We obtain that $X' = Y - K = \{\text{COUNTRY}\}$. Hence, $\text{agg}_{\text{COUNT\_DISTINCT(PROD\_SKU)}}(\text{SUM} \mid \{\text{COUNTRY}\})$ preserves the summarizability of PROD_SKU with respect to grouping set Y.

Aggregable properties provide "explanations" for end users of which aggregate queries preserve the summarizability condition of the aggregated attribute in a given stage of the data analysis session. In the previous example, the aggregable property $\text{agg}_{\text{COUNT\_DISTINCT(PROD\_SKU)}}(\text{SUM} \mid \{\text{COUNTRY}\})$ of T_r explains that table T_r can be used to count the number of distinct products per brand *and* year by taking the sum of $\text{COUNT\_DISTINCT}(\text{PROD\_SKU})$ along COUNTRY. However, T_r cannot be used to obtain the number of distinct products by brand *or* by year. In this case, the user must "backtrack" in the interactive data analysis session to the table PRODUCT_LIST to obtain this number.

4.3 Generalized attribute summarizability

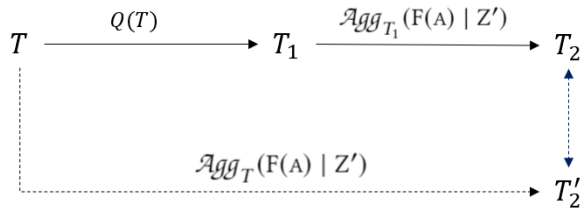
In the previous sections, as illustrated in Figure 9, we defined an attribute A in some table T to be summarizable with respect to some aggregate query $Q(T) = \mathcal{A}gg_T(F(A) \mid Z_1)$ and function F using G, if for any query $Q'(T) = \mathcal{A}gg_T(F(A) \mid Z_2)$ aggregating A along a subset Z_2 of Z_1, there exists an *equivalent* aggregate query $Q''(T_1) = \mathcal{A}gg_{T_1}(G(F(A)) \mid Z_2)$ on the result T_1 of $Q(T)$. Definition 13 generalizes this notion to any analytic query Q as follows.

Definition 13 (Generalized summarizable attribute)**.** Let $T(S)$ be an analytic table that is the input of an analytic query Q returning a table $T_1(S_1)$. Let A be an aggregable attribute of both T and T_1 and Z be a subset of the dimension attributes of $S \cap S_1$. If for any two valid aggregate queries $Q'(T) = \mathcal{A}gg_T(F(A) \mid Z')$ and $Q'(T_1) = \mathcal{A}gg_{T_1}(F(A) \mid Z')$

such that $\mathrm{Z} \subseteq \mathrm{Z}'$, and any two tuples $t_1 \in Q'(T)$ and $t_2 \in Q'(T_1)$, we have:

$$t_1.\mathrm{Z}' \equiv t_2.\mathrm{Z}' \Rightarrow t_1.\mathrm{F}(\mathrm{A}) \equiv t_2.\mathrm{F}(\mathrm{A})$$

then A is said to be *G-summarizable* in T with respect to query Q, grouping set Z and function F.



$$\forall\, Z', t_1 \in T_2', t_2 \in T_2\colon\ (Z \subseteq Z' \wedge t_1.Z' = T_2.Z') \Rightarrow (t_1.F(A) = t_2.F(A))$$

Fig. 11. G-summarizable attribute A in T with respect to Q, grouping set Z and function F

The above definition is illustrated in Figure 11. We make a few observations. First, T_2 and T_2' are not necessarily equal, *i.e.* T_2 might contain tuples that are not in T_2' and vice versa. Second, in Section 3.3, we established the propagation rules to compute the aggregable properties on A that hold in T_1 for F, which are used in the definition to determine which aggregate queries Q' on T_1 are valid. Third, an implicit assumption is that A is an attribute that exists in both T and T_1. Therefore, in the case of an aggregate or a pivot query Q, A must be a dimension attribute (since measure attributes of T do not exist anymore in T_1 - they have either been aggregated, pivoted or eliminated). Finally, grouping set Z defines a set of attributes that must be contained in Z′ and implicitly restricts the attributes along which aggregation can be done in Q'.

Before formalizing sufficient conditions for G-summarizability, we present a few examples.

Example 20. Consider a fact table T (in Table 24) defined over two dimension D_1 and D_2 with dimension attributes $\mathrm{A}_1, \mathrm{A}_2, \mathrm{A}_3$ from dimension D_1 (where $\mathrm{A}_1 \preccurlyeq A_2 \preccurlyeq A_3$) and $\mathrm{B}_1, \mathrm{B}_2$ from dimension D_2 (where $B_1 \preccurlyeq B_2$). We shall say that within table T, A_3 is the *highest* attribute of D_1 while B_2 is the *highest* attribute of D_2.

Table 24. G-summarizability in T with respect to a filter query

T	A_1	A_2	A_3	B_1	B_2	M
t_0	a_1	b_1	c_1	f_1	e_1	x_1
t_1	a_2	b_2	c_1	f_1	e_1	x_2
t_2	a_3	b_1	c_1	f_2	e_1	x_3
t_3	a_2	b_1	c_2	f_2	e_1	x_4

$Q_1(T)$	A_1	A_2	A_3	B_1	B_2	M
t_0	a_1	b_1	c_1	f_1	e_1	x_1
t_1	a_2	b_2	c_1	f_1	e_1	x_2
t_2	a_3	b_1	c_1	f_2	e_1	x_3

$Q_2(T)$	A_1	A_2	A_3	B_1	B_2	M
t_2	a_3	b_1	c_1	f_2	e_1	x_3
t_3	a_2	b_1	c_2	f_2	e_1	x_4

Take query $Q_1(T) = \mathit{Filter}_T(\{\mathrm{A}_3 = c_1\} \mid \{\mathrm{A}_3\})$ whose result table T_1 is displayed in Table 24. If we take a grouping set $\mathrm{Z} = \{\mathrm{A}_3\}$, then for each partition $T^p = \sigma_{\mathrm{A}_3=p}(T)$ of T we either have $Q_1(T^p) = T^x$ or $Q_1(T^p) = \emptyset$. In our example, we have: $Q_1(T^{c_1}) = T^{c_1}$ and $Q_1(T^{c_2}) = \emptyset$. Thus, for any subset Z′ of dimension attributes of T containing A_3, we either have $\Pi_{\mathrm{Z}'}(Q_1(T^p)) = \Pi_{\mathrm{Z}'}(T^p)$ or $\Pi_{\mathrm{Z}'}(Q_1(T^p))$ is empty, where Π is a projection without duplicate

elimination. Therefore, any valid aggregation query with grouping attributes Z′ containing A_3 returns, for each partition of T_1 defined by Z′, the same result as for the corresponding partition of T defined by Z′. Hence, any attribute A is G-summarizable in T with respect to query Q_1, grouping set $Z = \{A_3\}$ and any function F applicable to A in T. Observe that this is not the case for any other grouping set Z that does not contain attribute A_3. For instance, if $Z = \{A_2\}$ then for partition T^{b_1}, $Q_1(T^{b_1}) \neq T^{b_1}$.

However, the previous reasoning does not apply if T is filtered on a measure attribute M, like in query $Q_2(T) = \mathcal{F}ilter_T(\{M \in \{x_3, x_4\}\} \mid \{M\})$. Since a measure attribute cannot belong to the grouping set of an aggregate query, we cannot define a partitioning set $Z = \{M\}$. Then, to identify a set of dimension attributes Z such that all attributes A are G-summarizable in T with respect to query Q_2, grouping set Z and an applicable function F, we must find a set of dimension attributes Z that partitions T into $Q_2(T)$ and $T - Q_2(T)$. In our example, $Z = \{B_1\}$ would be a possible solution that cannot be easily found.

Example 21. Consider now the fact tables T and T' defined over the same dimensions D_1 and D_2 as before, as shown below in Table 25. Take the left-merge query $Q(T, T') = T \rtimes_Y T' = T_1$ where $Y = \{A_1, A_2, B_1, B_2\}$. Any attribute A of T is G-summarizable in T with respect to Q, grouping set $Z = \emptyset$, and any function F applicable to A, because the duplicate preserving projection of T_1 on the attributes of T is equal to table T.

Let us look at the G-summarizability of attributes in T' with respect to Q and F. First, grouping set $Z = \{A_2, B_2\}$ containing the "highest" attributes in Y defines two partitions of T'. We can see that partition T'^{b_1,e_1} is different from the duplicate-preserving projection of $T_1^{b_1,e_1}$ on the attributes of T' (tuples t_3, t_4 of T' have no corresponding tuples in T_1). So, any valid aggregation query over a partitioning on Z would violate the G-summarizability property. Indeed, the only grouping set Z for which we have the equality of non-empty partitions is $Z = \{A_1, A_2, B_1, B_2\}$. We shall see later that if a valid aggregation can be expressed over Y in T' then Z can be equal to Y.

Table 25. G-summarizability in T with respect to a left-merge query

T	A_1	A_2	B_1	B_2	M
t_0	a_1	b_1	f_2	e_1	x_1
t_1	a_4	b_2	f_4	e_2	x_2

T'	A_1	A_2	A_3	B_1	B_2	M′
t_3	a_1	b_1	c_1	f_1	e_1	y_1
t_4	a_2	b_1	c_1	f_2	e_1	y_3
t_5	a_1	b_1	c_1	f_2	e_3	y_4
t_6	a_1	b_1	c_1	f_2	e_1	y_5

$T \rtimes T'$	A_1	A_2	A_3	B_1	B_2	M	M′
t_{10}	a_1	b_1	c_1	f_2	e_1	x_1	y_5
t_{11}	a_4	b_2	-	f_4	e_2	x_2	-

Now take the same left-merge query Q as before applied to the tables displayed in Table 26. For grouping set $Z = \{A_2, B_2\}$, the partition of T' with values (b_1, e_1) has a corresponding identical partition in T_1 after a duplicate-preserving projection on the attributes of T'. The partition of T' with values (b_1, e_3) has a corresponding empty partition in T_1. However, the partition of T_1 with values (b_2, e_1) has one extra tuple with respect to the corresponding partition in T' because tuple t_7 is matched by two tuples of T and its attribute values appear duplicated in T_1 (in tuples t_{12} and t_{13}). Hence, for attributes of T', G-summarizability in T' with respect to Q and grouping set $Z = \{A_2, B_2\}$ must be restricted to functions that are insensitive to duplicates (i.e., COUNT_DISTINCT, MIN, MAX).

Finally, any attribute of T is G-summarizable in T with respect to Q and grouping set $Z = \emptyset$ because the duplicate preserving projection of T_1 on the attributes of T is equal to T. Thus, T_1 is aggregable over any partitioning of attributes of T.

Table 26. G-summarizability in T' with respect to a left-merge query

T	A1	A2	A3	B1	B2	M
t_0	a_1	b_1	c_1	f_1	e_1	x_1
t_1	a_2	b_1	c_1	f_1	e_1	x_2
t_2	a_3	b_2	c_1	f_2	e_1	x_3
t_3	a_3	b_2	c_2	f_2	e_1	x_4
t_4	a_4	b_2	c_1	f_2	e_1	x_5

T'	A1	A2	B1	B2	M′
t_5	a_1	b_1	f_1	e_1	y_1
t_6	a_2	b_1	f_1	e_1	y_2
t_7	a_3	b_2	f_2	e_1	y_3
t_8	a_2	b_1	f_3	e_3	y_4

$T \rtimes T'$	A1	A2	A3	B1	B2	M	M′
t_{10}	a_1	b_1	c_1	f_1	e_1	x_1	y_1
t_{11}	a_2	b_1	c_1	f_1	e_1	x_2	y_2
t_{12}	a_3	b_2	c_1	f_2	e_1	x_3	y_3
t_{13}	a_3	b_2	c_2	f_2	e_1	x_4	y_3
t_{14}	a_4	b_2	c_1	f_2	e_1	x_5	-

Proposition 8 (Queries satisfying G-summarizability)**.** Let Q be a unary or binary analytic query with some input table $T(S)$ returning a table $T_1(\mathrm{S}_1)$ and S_D be the dimension attributes in $S \cap \mathrm{S}_1$. Let Z be a subset of S_D, and A′ be an attribute in $S \cap \mathrm{S}_1$ such that $\mathrm{agg}_{\mathrm{A}'}(\mathrm{F}, \mathrm{X})$ and $\mathrm{agg}_{\mathrm{A}'}(\mathrm{F}, \mathrm{X}_1)$ hold in T and T_1 respectively. Then, *the attribute* A′ *is G-summarizable in* T *with respect to query* Q*, grouping set* Z*, and function* F in the following cases:
Unary queries:

(1) $Q = \mathit{Filter}_T(P \mid \mathrm{Y})$, $\mathrm{Y} \subseteq S_D$, $\mathrm{A}' \in S$ and $\mathrm{Z} = S_D - \mathrm{X}_1 \cup \mathrm{Y}$.
(2) $Q = \mathit{Project}_T(\mathrm{Y}, f(\mathrm{Z}') \rightarrow \mathrm{M})$, $\mathrm{A}' \in \mathrm{Y}$ and $\mathrm{Z} = S_D - \mathrm{X}_1$.
(3) $Q = \mathit{Agg}_T(\mathrm{G}(\mathrm{A}) \mid \mathrm{Y})$, $\mathrm{A}' \in \mathrm{Y}$, $\mathrm{Z} = \mathrm{Y} - \mathrm{X}_1$ and $\mathrm{F} \in \{\mathrm{MIN}, \mathrm{MAX}, \mathrm{COUNT\_DISTINCT}\}$.
(4) $Q = \mathit{Pivot}_T(\mathrm{A} \mid \mathrm{Y})$, $\mathrm{A}' \in S - \mathrm{Y} - \{\mathrm{A}\}$, $\mathrm{Z} = S_D - \mathrm{X}_1 - \mathrm{Y} - \{\mathrm{A}\}$) and $\mathrm{F} \in \{\mathrm{MIN}, \mathrm{MAX}, \mathrm{COUNT\_DISTINCT}\}$.

Merge queries: In the following, let $\mathrm{Y}^{top} \subseteq \mathrm{Y}$ denote the subset of "highest" attributes in the set of join attributes Y.

(1) $Q = T \rtimes_{\mathrm{Y}} T'$, $\mathrm{A}' \in \mathrm{S}$, $\mathrm{Z} = \mathrm{S}_D - \mathrm{X}_1$ and if $\mathrm{Y} \not\rightarrow S_r$ then $\mathrm{F} \in \{\mathrm{MIN}, \mathrm{MAX}, \mathrm{COUNT\_DISTINCT}\}$.
(2) $Q = T \bowtie\!\!\!\!\subset_{\mathrm{Y}} T'$ or $Q = T \rtimes\!\!\!\!\subset_{\mathrm{Y}} T'$:
   (a) If $\mathrm{A}' \in \mathrm{Y}$ and for all non-empty partitions $T'^y = \sigma_{\mathrm{Y}^{top}=y}(T')$ of T', the corresponding partition $T^y = \sigma_{\mathrm{Y}^{top}=y}(T)$ of T is empty or $\pi_{\mathrm{Y}}(T^y)$ is *equal* to $\pi_{\mathrm{Y}}(T'^y)$, then $\mathrm{Z} = \mathrm{S}_D - \mathrm{X}_1 \cup \mathrm{Y}^{top}$
   (b) If $\mathrm{A}' \in \mathrm{S}-\mathrm{Y}$ and for all non-empty partitions $T'^y = \sigma_{\mathrm{Y}=y}(T')$ of T' and corresponding partitions $T^y = \sigma_{\mathrm{Y}=y}(T)$ of T, $\pi_{\mathrm{Y}}(T^y)$ is a *subset of* $\pi_{\mathrm{Y}}(T'^y)$, then $\mathrm{Z} = \mathrm{S}_D - \mathrm{X}_1$.
   (c) Otherwise, $\mathrm{Z} = (\mathrm{S}_D - \mathrm{X}_1) \cup \mathrm{Y}$.
   In addition, for all cases, if $\mathrm{Y} \not\rightarrow S_r$ then $\mathrm{F} \in \{\mathrm{COUNT\_DISTINCT}, \mathrm{MIN}, \mathrm{MAX}\}$.
(3) $Q = T \bowtie_{\mathrm{Y}} T'$, $\mathrm{A}' \in \mathrm{S}$: if for all non-empty partitions $T'^y = \sigma_{\mathrm{Y}^{top}=y}(T')$ and corresponding partitions $T^y = \sigma_{\mathrm{Y}^{top}=y}(T)$, $\pi_{\mathrm{Y}}(T'^y)$ contains $\pi_{\mathrm{Y}}(T^y)$ then $\mathrm{Z} = \mathrm{S}_D - \mathrm{X}_1 \cup \mathrm{Y}^{top}$ else $\mathrm{Z} = \mathrm{S}_D - \mathrm{X}_1 \cup \mathrm{Y}$.
   In addition, if $\mathrm{Y} \not\rightarrow S_r$ then $\mathrm{F} \in \{\mathrm{COUNT\_DISTINCT}, \mathrm{MIN}, \mathrm{MAX}\}$.

Set queries: In the following, let $\mathrm{Y}^{top} \subseteq S_D$ denote the set of "highest" attributes in the set of dimension attributes $S_D \subseteq S$ of T and T'.

(1) $Q = T \cup T'$, $\mathrm{A}' \in \mathrm{S}$ and if $\pi_{\mathrm{Y}^{top}}(T) \cap \pi_{\mathrm{Y}^{top}}(T') = \emptyset$, then $\mathrm{Z} = S_D - \mathrm{X}_1 \cup \mathrm{Y}^{top}$.
(2) $Q = T - T'$, $\mathrm{A}' \in \mathrm{S}$ and if all partitions $\sigma_{\mathrm{Y}^{top}=y}(T)$ are equal to or disjoint with $\sigma_{\mathrm{Y}^{top}=y}(T')$, then $\mathrm{Z} = S_D - \mathrm{X}_1 \cup \mathrm{Y}^{top}$.

Proof. *Unary queries:* We shall use symbol Π to denote the duplicate preserving projection and π do denote duplicate eliminating projection. For each case of a unary query Q on some table T producing a table T_1, we first prove that, for any partition $T^x = \sigma_{\mathrm{Z}=x}(T)$ of T and $T_1^x = \sigma_{\mathrm{Z}=x}(T_1)$ of T_1, we have: (1) $\pi_{\mathrm{Z},\mathrm{A}'}(T_1^x) = \pi_{\mathrm{Z},\mathrm{A}'}(T^x)$ (both partitions are equal modulo duplicates), or (2) T^x is empty, or (3) T_1^x is empty. We call the previous condition the *G-summarizability*

condition on T and T_1 for Z *and* A'. If this condition holds, we can show that A' is G-summarizale in T with respect to Q, grouping set Z and F as follows:

- In case (1), if $\mathrm{Z} \mapsto \mathrm{A}'$, then $\Pi_{\mathrm{Z},\mathrm{A}'}(T_1^x) = \Pi_{\mathrm{Z},\mathrm{A}'}(T^x)$ (both partitions are identical including duplicates) and it is obvious that any query, that aggregates A' using F grouped by Z' containing all attributes in Z, produces the same result on T and T_1 and the conditions for G-summarizability are fulfilled. Otherwise, if $\mathrm{Z} \not\mapsto \mathrm{A}'$, F must be restricted to aggregation functions that are not sensitive to duplicates ($\mathrm{F} \in \{\mathrm{COUNT\_DISTINCT}, \mathrm{MIN}, \mathrm{MAX}\}$).
- In cases (2) and (3), the aggregated value does not exist, respectively, in the input table T or the result table T_1. This is also sufficient for satisfying the G-summarizability property.

Then, since $\mathrm{agg}_{\mathrm{A}'}(\mathrm{F}, \mathrm{X}_1)$ holds in T_1, any query that aggregates A' using F grouped by Z' containing all dimension attributes in $S_r \cap S_D - \mathrm{X}_1$ is valid. Thus it is sufficient to show that Z contains all dimension attributes in $S_r \cap S_D - \mathrm{X}_1$. We call this condition the *aggregable property condition.*

(1) Analytic filter $T_1 = \mathit{Filter}_T(P \mid \mathrm{Y})$: By the condition $\mathrm{Y} \subseteq S_D$, all attributes in Y are dimension attributes. By $\mathrm{Z} = S_D - \mathrm{X}_1 \cup \mathrm{Y}$ we have $\mathrm{Y} \subseteq \mathrm{Z}$. Then for any non-empty partition $T^x = \sigma_{\mathrm{Z}=x}(T)$ of T we can show that if P is true for some tuple in T^x, it is true for *all* tuples in T^x: Z contains all attributes of filtering predicate P and we can show that for all $\mathrm{Z} = x$ either $\mathrm{Z} = x \Rightarrow P(\mathrm{Y})$ or $\mathrm{Z} = x \Rightarrow \neg P(\mathrm{Y})$. Therefore, the corresponding partition $T_1^x = \sigma_{\mathrm{Z}=x}(\mathit{Filter}_T(P \mid \mathrm{Y})) = \mathit{Filter}_T(P \wedge \mathrm{Z} = x \mid \mathrm{Z})$ is either empty or equal to T^x. Hence, for any T^x and attribute A' in S, we either have $\Pi_{\mathrm{Z},\mathrm{A}'}(T_1^x) = \Pi_{\mathrm{Z},\mathrm{A}'}(T^x)$ (both partitions are identical with duplicates) or T_1^x is empty. Finally, the aggregable property condition holds: $S_r \cap S_D - \mathrm{X}_1 = S_D - \mathrm{X}_1 \subseteq \mathrm{Z} = S_D - \mathrm{X}_1 \cup \mathrm{Y}$.

(2) Analytic projection $T_1 = \mathit{Project}_T(\mathrm{Y}, f(\mathrm{Z}') \rightarrow \mathrm{M})$: By definition of analytic projection, for any subset $\mathrm{X} \subseteq \mathrm{Y}$, we have $\Pi_{\mathrm{X}}(T_1) = \Pi_{\mathrm{X}}(T)$. Since $\mathrm{Z} = S_D - \mathrm{X}_1 \subseteq S_D \subseteq \mathrm{Y}$ and $\mathrm{A}' \in \mathrm{Y}$, we have $\Pi_{\mathrm{Z},\mathrm{A}'}(T_1) = \Pi_{\mathrm{Z},\mathrm{A}'}(T)$. Finally, the aggregable property condition holds: $S_r \cap S_D - \mathrm{X}_1 = S_D - \mathrm{X}_1 \subseteq \mathrm{Z} = S_D - \mathrm{X}_1$.

(3) Analytic aggregate $T_1 = \mathit{Agg}_T(\mathrm{G}(\mathrm{A}) \mid \mathrm{Y})$: By definition of aggregate queries, for any $\mathrm{X} \subseteq \mathrm{Y}$ and partition T^x, we have $\pi_{\mathrm{X}}(T_1^x) = \pi_{\mathrm{X},\mathrm{A}'}(T^x)$ (duplicate eliminating projection). Since $\mathrm{Z} \subseteq \mathrm{Y}$ and $\mathrm{A}' \in \mathrm{Y}$, we then have $\pi_{\mathrm{Z},\mathrm{A}}(T_1^x) = \pi_{\mathrm{Z},\mathrm{A}}(T^x)$. However, T^x generally contains several tuples that are merged into a single tuple in T_1. Therefore, F must be restricted to functions that are not sensitive to duplicates ($\mathrm{F} \in \{\mathrm{COUNT\_DISTINCT}, \mathrm{MIN}, \mathrm{MAX}\}$). Finally, the aggregable property condition holds: $S_r \cap S_D - \mathrm{X}_1 = \mathrm{Y} - \mathrm{X}_1 \subseteq \mathrm{Z} = \mathrm{Y} - \mathrm{X}_1$.

(4) Analytic pivot $T_1 = \mathit{Pivot}_T(\mathrm{A}' \mid \mathrm{Y})$: We apply similar arguments as for aggregate queries on the remaining attributes $\mathrm{Y}' = S - \mathrm{Y} - \{\mathrm{A}\}$ in T_1. By definition of pivot queries, for any subset $\mathrm{X} \subseteq \mathrm{Y}'$ and partition T^x, we have $\pi_{\mathrm{X}}(T_1^x) = \pi_{\mathrm{X}}(T^x)$ (duplicate eliminating projection) and $\pi_{\mathrm{Z},\mathrm{A}'}(T_1^x) = \pi_{\mathrm{Z},\mathrm{A}'}(T^x)$ in particular for $\mathrm{Z} \subseteq \mathrm{Y}'$ and $\mathrm{A}' \in \mathrm{Y}'$. However, in the general case, T^x contains several tuples that are merged into a single tuple in T_1 and F must be restricted to functions that are not sensitive to duplicates ($\mathrm{F} \in \{\mathrm{COUNT\_DISTINCT}, \mathrm{MIN}, \mathrm{MAX}\}$). Finally, the aggregable property condition holds: $S_r \cap S_D - \mathrm{X}_1 = S_D - \mathrm{Y} - \mathrm{X}_1 - \{\mathrm{A}\} \subseteq \mathrm{Z} = S_D - \mathrm{X}_1 - \mathrm{Y} - \{\mathrm{A}\}$.

Merge queries: For merge queries Q over two tables T and T' producing a table T_1, we also check the G-summarizability condition and the aggregable property condition as for unary queries:

- For any couple of partitions $T^x = \sigma_{\mathrm{Z}=x}(T)$ and $T_1^x = \sigma_{\mathrm{Z}=x}(T_1)$, either at least one of the two partitions is empty or $\pi_{\mathrm{S}}(T^x)$ is equal to $\pi_{\mathrm{S}}(T_1^x)$.
- Z contains all dimension attributes in $S_r \cap S_D - \mathrm{X}_1$.

(1) Left-merge query $T(\mathrm{S}) \ltimes_{\mathrm{Y}} T'(\mathrm{S}')$: By definition, $\pi_{\mathrm{S}}(T) = \pi_{\mathrm{S}}(T_1)$ (each tuple of T produces one or more tuples in T_1 and vice versa). Thus, for any $\mathrm{Z} \cup \{\mathrm{A}'\} \subseteq \mathrm{S}_d$ we also have $\pi_{\mathrm{Z},\mathrm{A}'}(T) = \pi_{\mathrm{Z},\mathrm{A}'}(T_1)$. By definition, a tuple in T

can only appear twice in $\Pi_{\mathrm{Z}}(T_1)$ if $\mathrm{Y} \not\mapsto \mathrm{S}'$. If that is the case, F must be restricted to functions that are not sensitive to duplicates: $\mathrm{F} \in \{\mathrm{COUNT\_DISTINCT}, \mathrm{MIN}, \mathrm{MAX}\}$. Finally, the aggregable property condition holds: $S_r \cap S_D - \mathrm{X}_1 = S_D - \mathrm{X}_1 \subseteq \mathrm{Z} = S_D - \mathrm{X}_1$.

(2) Right-merge query $T(\mathrm{S}) \bowtie\!\!\!\subset_{\mathrm{Y}} T'(\mathrm{S}')$ or full-merge query $Q = T(\mathrm{S}) \supset\!\!\!\bowtie\!\!\!\subset_{\mathrm{Y}} T'(\mathrm{S}')$:

(a) $\mathrm{A}' \in \mathrm{Y}$: We assume that for all non-empty partitions $T'^{y} = \sigma_{\mathrm{Y}^{top}=y}(T')$ of T', the corresponding partition $T^{y} = \sigma_{\mathrm{Y}^{top}=y}(T)$ is either empty or $\pi_Y(T^{y})$ is equal to $\pi_{\mathrm{Y}}(\sigma_{\mathrm{Y}^{top}=y}(T'))$. From this assumption and the definition of right-outer join, it directly follows that for any non-empty partition $T_1^{y} = \sigma_{\mathrm{Y}^{top}=y}(T_1)$ of T_1 and corresponding partition T^{y}, $T^{y} = \emptyset$ or $\pi_{\mathrm{S}}(T^{y}) = \pi_{\mathrm{S}}(T_1^{y})$. Then, for all $\mathrm{X} \supseteq \mathrm{Y}^{top}$, all non-empty partitions $\pi_{\mathrm{X}}(\sigma_{\mathrm{X}=x}(T))$ are equal to $\pi_{\mathrm{X}}(\sigma_{\mathrm{X}=x}(T_1))$ and since, by definition of Z, $\mathrm{Z} \cup \{\mathrm{A}'\} \supseteq \mathrm{Y}^{top}$, the previous condition also holds for $\mathrm{X} = \mathrm{Z} \cup \{\mathrm{A}'\}$. Therefore the G-summarizability condition holds on T and T_1 for Z and A′. Finally, the aggregable property condition also holds: $S_r \cap S_D - \mathrm{X}_1 = S_D - \mathrm{X}_1 \subseteq \mathrm{Z} = S_D - \mathrm{X}_1 \cup \mathrm{Y}^{top}$.

(b) $\mathrm{A}' \in \mathrm{S} - \mathrm{Y}$: We assume that for all non-empty partitions, $T'^{y} = \sigma_{\mathrm{Y}^{top}=y}(T')$, and corresponding partitions $T^{y} = \sigma_{\mathrm{Y}^{top}=y}(T)$, $\pi_Y(T^{y})$ is a subset of $\pi_{\mathrm{Y}}(\sigma_{\mathrm{Y}^{top}=y}(T'))$. From this assumption and the definition of right-outer join, it directly follows that for any non-empty partition $T_1^{y} = \sigma_{\mathrm{Y}^{top}=y}(T_1)$ and corresponding partition T^{y}, $\pi_{\mathrm{S}-\mathrm{Y}}(T_1^{y}) - \pi_{\mathrm{S}-\mathrm{Y}}(T^{y})$ only contains *null* values. Then, for all $\mathrm{X} \subseteq \mathrm{S}_D$ and $\mathrm{A}' \in \mathrm{S} - \mathrm{Y}$, all non-empty partitions $\pi_{\mathrm{X},\mathrm{A}'}(\sigma_{\mathrm{X}=x}(T))$ are equal to $\pi_{\mathrm{X}}(\sigma_{\mathrm{X}=x \wedge \mathrm{A}' \neq null}(T_1))$ and since, by definition of Z, $\mathrm{Z} \subseteq \mathrm{S}_D$, the previous condition also holds for $\mathrm{X} = \mathrm{Z} \cup \{\mathrm{A}'\}$. Therefore, the G-summarizability condition holds on T and T_1 for Z and A′. Finally, the aggregable property condition also holds: $S_r \cap S_D - \mathrm{X}_1 = S_D - \mathrm{X}_1 \subseteq \mathrm{Z} = S_D - \mathrm{X}_1 \cup \mathrm{Y}$.

(c) Otherwise: From the definition of right-merge and full-merge, it directly follows that for any non-empty partition $T_1^{y} = \sigma_{\mathrm{Y}=y}(T_1)$, the corresponding partition T^{y} is either empty or equal to $\pi_{\mathrm{S}}(T_1^{y})$. Then, for all $\mathrm{X} \supseteq \mathrm{Y}$, all non-empty partitions $\pi_{\mathrm{X}}(\sigma_{\mathrm{X}=x}(T))$ are equal to $\pi_{\mathrm{X}}(\sigma_{\mathrm{X}=x}(T_1))$ and since, by definition of Z, $\mathrm{Z} \cup \{\mathrm{A}'\} \supseteq \mathrm{Y}$, the previous condition also holds for $\mathrm{X} = \mathrm{Z} \cup \{\mathrm{A}'\}$. Therefore, the G-summarizability condition holds for T and T_1 for Z and A′. Finally, the aggregable property condition also holds: $S_r \cap S_D - \mathrm{X}_1 = S_D - \mathrm{X}_1 \subseteq \mathrm{Z} = S_D - \mathrm{X}_1 \cup \mathrm{Y}$.

(3) $Q = T(\mathrm{S}) \bowtie_{\mathrm{Y}} T'(\mathrm{S}')$:

(a) "if" part: We assume that for all non-empty partitions $T'^{y} = \sigma_{\mathrm{Y}^{top}=y}(T')$, the corresponding partition $T^{y} = \sigma_{\mathrm{Y}^{top}=y}(T)$ is contained in $\pi_{\mathrm{Y}}(T'^{y})$. From this assumption and the definition of inner join, it directly follows that any non-empty partition $T_1^{y} = \sigma_{\mathrm{Y}^{top}=y}(T_1)$ in the result is equal to the corresponding partition T^{y}: $\pi_{\mathrm{S}}(T^{y}) = \pi_{\mathrm{S}}(T_1^{y})$. Then, for all $\mathrm{X} \supseteq \mathrm{Y}^{top}$, all non-empty partitions $\pi_{\mathrm{X}}(\sigma_{\mathrm{X}=x}(T))$ are equal to $\pi_{\mathrm{X}}(\sigma_{\mathrm{X}=x}(T_1))$ and since, by definition of Z, $\mathrm{Z} \cup \{\mathrm{A}'\} \supseteq \mathrm{Y}^{top}$, the previous condition also holds for $X = \mathrm{Z} \cup \{\mathrm{A}'\}$. Therefore the G-summarizability condition holds on T and T_1 for grouping set Z and attribute A′. Finally, the aggregable property condition also holds: $S_r \cap S_D - \mathrm{X}_1 = S_D - \mathrm{X}_1 \subseteq \mathrm{Z} = S_D - \mathrm{X}_1 \cup \mathrm{Y}^{top}$.

(b) "else" part: From the definition of inner join, it directly follows that for any non-empty partition $T_1^{y} = \sigma_{\mathrm{Y}=y}(T_1)$, the corresponding partition T^{y} is equal to $\pi_{\mathrm{S}}(T_1^{y})$. Then, for all $\mathrm{X} \supseteq \mathrm{Y}$, all non-empty partitions $\pi_{\mathrm{X}}(\sigma_{\mathrm{X}=x}(T_1))$ are equal to $\pi_{\mathrm{X}}(\sigma_{\mathrm{X}=x}(T))$ and since, by definition of Z, $\mathrm{Z} \cup \{\mathrm{A}'\} \supseteq \mathrm{Y}$, the previous condition also holds for $\mathrm{X} = \mathrm{Z} \cup \{\mathrm{A}'\}$. Therefore the G-summarizability condition holds on T and T_1 for groupîng set Z and A′. Finally, the aggregable property condition also holds: $S_r \cap S_D - \mathrm{X}_1 = S_D - \mathrm{X}_1 \subseteq \mathrm{Z} = S_D - \mathrm{X}_1 \cup \mathrm{Y}$.

Set queries: In the following, let $\mathrm{Y}^{top} \subseteq S_D$ denote the set of "highest" attributes in the set of dimension attributes $S_D \subseteq S$ of T and T'.

(1) $\mathcal{Q} = T(\mathrm{S}) \cup T'(\mathrm{S})$: By the assumption $\pi_{\mathrm{Y}^{top}}(T) \cap \pi_{\mathrm{Y}^{top}}(T') = \emptyset$ and the definition of union, it follows that for any non-empty partition $T_1^y = \sigma_{\mathrm{Y}^{top}=y}(T_1)$ in the result, the corresponding partition T^y is either empty or T^y is equal to T_1^y. Then, for all $\mathrm{X} \supseteq \mathrm{Y}^{top}$, all non-empty partitions $\pi_{\mathrm{X}}(\sigma_{\mathrm{X}=x}(T))$ are equal to $\pi_{\mathrm{X}}(\sigma_{\mathrm{X}=x}(T_1))$ and since, by definition of Z, $\mathrm{Z} \supseteq \mathrm{Y}^{top}$, the previous condition also holds for $\mathrm{Z} = \mathrm{S}_D - \mathrm{X}_1 \cup \mathrm{Y}^{top}$. Finally, the aggregable property condition also holds: $S_r \cap S_D - \mathrm{X}_1 = S_D - \mathrm{X}_1 \subseteq \mathrm{Z} = S_D - \mathrm{X}_1 \cup \mathrm{Y}^{top}$.

(2) $\mathcal{Q} = T(\mathrm{S}) - T'(\mathrm{S})$: By assumption, all partitions $\sigma_{\mathrm{Y}^{top}=y}(T)$ are equal to or disjoint with $\sigma_{\mathrm{Y}^{top}=y}(T')$. Then, by the definition of set-difference, it follows that for any non-empty partition $T_1^y = \sigma_{\mathrm{Y}^{top}=y}(T_1)$ in the result, the corresponding partition T^y is either empty or T^y is equal to T_1^y. Then, for all $\mathrm{X} \supseteq \mathrm{Y}^{top}$, all non-empty partitions $\pi_{\mathrm{X}}(\sigma_{\mathrm{X}=x}(T))$ are equal to $\pi_{\mathrm{X}}(\sigma_{\mathrm{X}=x}(T_1))$ and since, by definition of Z, $\mathrm{Z} \supseteq \mathrm{Y}^{top}$, the previous condition also holds for $\mathrm{X} = \mathrm{Z} \cup \{\mathrm{A}'\}$. Therefore, the G-summarizability condition holds on T and T_1 for Z and A′. Finally, the aggregable property condition also holds: $S_r \cap S_D - \mathrm{X}_1 = S_D - \mathrm{X}_1 \subseteq \mathrm{Z} = S_D - \mathrm{X}_1 \cup \mathrm{Y}^{top}$.

□

Observe that for merge queries and set queries we choose the "highest" dimension Y^{top} as candidates for checking the G-summarizability conditions. In fact, we might check this condition for any subset Y' of attributes from Y or S_D instead of Y^{top}, and identify the minimal candidates for which these conditions hold. There are two main reasons for only choosing Y^{top}. First, checking the G-summarizability condition for a subset of attributes mainly corresponds to comparing the size of partitions in two different tables obtained by two aggregate queries. This basic operation is costly and the systematic exploration of all attribute subsets Y' might, even with efficient pruning techniques, take too much time in an interactive data exploration session. Secondly, the choice of the highest attributes Y^{top} is based on the realistic hypothesis that the majority of analytic queries aggregate values along these attributes and other lower attributes.

4.4 Controlling G-summarizability using aggregable properties

The following proposition refines the propagation rules for aggregable properties of Section 3.3, using the results of Proposition 8, to guarantee the G-summarizability of attributes.

Proposition 9 (aggregable properties with G-summarizability for unary queries)**.** Let $\mathcal{Q}$ be a unary analytic query with some input table $T(S)$ returning a table $T_r(\mathrm{S}')$, and S_D be all the dimension attributes of $S \cap \mathrm{S}'$. Let A′ be an attribute in $S \cap \mathrm{S}'$ such that $\mathrm{agg}_{A'}(\mathrm{F}, \mathrm{X})$ holds in T. Then, in the cases of queries $\mathcal{Q}$ of Table 27, the aggregable property $\mathrm{agg}_{A'}(\mathrm{F}, \mathrm{X}')$ holds in T_r and is such that for all Z where $\mathrm{S}_D - \mathrm{X}' \subseteq \mathrm{Z}$, A′ is G-summarizable in T with respect to query $\mathcal{Q}$, grouping set Z, and function F.

Proof. The proof mainly consists in defining the "new" X' as the "complement" of Z as defined in Proposition 8 where X' is replaced by its definition in Table 19. For example, for filter queries, since $\mathrm{Z} = (S_D - \mathrm{X}') \cup \mathrm{Y}$ and $\mathrm{X}' = \mathrm{X}$, we obtain $\mathrm{Z} = S_D - (\mathrm{X} - \mathrm{Y})$ and its complement $\mathrm{X}' = S_r - \mathrm{Z} = \mathrm{X} - \mathrm{Y}$. For aggregation queries, $\mathrm{Z} = \mathrm{Y} - \mathrm{X}_1$ and $\mathrm{X}_1 = \mathrm{Y} \cap \mathrm{X}$ and we obtain $\mathrm{Z} = \mathrm{Y} - (\mathrm{Y} \cap \mathrm{X})$ and its complement $\mathrm{X}' = \mathrm{Y} - \mathrm{Z} = \mathrm{Y} \cap \mathrm{X}$. □

We make the following observations on the rules of Table 27. First, the rule for Project is unchanged with respect to Table 19. Second, when Q is an aggregate query $\mathcal{Q} = \mathcal{A}gg_T(\mathrm{G}(\mathrm{B}) \mid \mathrm{Y})$, the aggregable property for attribute G(B) is computed using the rule of Table 19 to guarantee the summarizability of attribute B. For attributes of Y, the only

Table 27. Propagation rules for unary operations on $T(S)$ preserving G-summarizability

Query on T(S)	Propagation rule for inferring the aggregable properties for attributes $\mathrm{A}' \in S \cap S_r$ of the result $T_r(S_r)$	*User action*
$\mathit{Filter}_T(P \mid \mathrm{Y})$	attribute $\mathrm{A}' \in S_r$, $\mathrm{Y} \subseteq S_D$ and $\mathrm{agg}_{\mathrm{A}'}(\mathrm{F}, \mathrm{X})$ holds in T: $\mathrm{agg}_{\mathrm{A}'}(\mathrm{F}, \mathrm{X}')$ holds in T_r with $\mathrm{X}' = \mathrm{X} - \mathrm{Y}$, X'_d = fact identifier and $\mathrm{X}'_f = \mathrm{X}_f - \mathrm{Y}$	*Minimize* X'_d
$\mathit{Project}_T(\mathrm{Y}, f(\mathrm{Z}) \rightarrow \mathrm{M})$	dimension attribute $\mathrm{A}' \in \mathrm{Y}$ and $\mathrm{agg}_{\mathrm{A}'}(\mathrm{F}, \mathrm{X})$ holds in T: $\mathrm{agg}_{\mathrm{A}'}(\mathrm{F}, \mathrm{X}')$ holds in T_r with $\mathrm{X}' = \mathrm{X}$, $\mathrm{X}'_d = \mathrm{X}_d$ and $\mathrm{X}'_f = \mathrm{X}_f$.	*None*
$\mathit{Pivot}_T(\mathrm{A} \mid \mathrm{Y})$	attribute $\mathrm{A}' \in S_r - \{\mathrm{A}\}$ and $\mathrm{agg}_{\mathrm{A}'}(\mathrm{F}, \mathrm{X})$ holds in T: if $\mathrm{X}_d \cap \mathrm{Y} = \emptyset$ then $\mathrm{agg}_{\mathrm{A}'}(\mathrm{F}, \mathrm{X}')$ holds in T_r with $\mathrm{X}' = \mathrm{X} - \mathrm{Y}$, $\mathrm{X}'_d = \mathrm{X}_d$ and $\mathrm{X}'_f = \mathrm{X}_f - \mathrm{Y}$	*None*
	else $\mathrm{agg}_{\mathrm{A}'}(\mathrm{F}, \mathrm{X}')$ holds in T_r with X' as defined by the rules of Table 17 with X'_d = fact identifier and $\mathrm{X}'_f = \mathrm{X}_f - \mathrm{Y}$ and $\mathrm{F} \in \{\mathrm{MIN}, \mathrm{MAX}, \mathrm{COUNT\_DISTINCT}\}$	*Minimize* X'_d
$\mathit{Agg}_T(\mathrm{G}(\mathrm{A}) \mid \mathrm{Y})$	dimension attribute $\mathrm{A}' \in \mathrm{Y}$ and $\mathrm{agg}_{\mathrm{A}'}(\mathrm{F}, \mathrm{X})$ holds in T: $\mathrm{agg}_{\mathrm{A}'}(\mathrm{F}, \mathrm{X}')$ holds in T_r with $\mathrm{X}' = \mathrm{X} \cap \mathrm{Y}$ and $\mathrm{X}'_f = \mathrm{X}_f \cap \mathrm{Y}$. and $\mathrm{F} \in \{\mathrm{MIN}, \mathrm{MAX}, \mathrm{COUNT\_DISTINCT}\}$	*None*

refinement to the rule in Table 19 is to restrict the scope of F. The same observation applies to the refined propagation rule for a pivot query.

Proposition 10 (aggregable properties with G-summarizability for binary queries)**.** Let $T(S)$ and $T'(S')$ be two analytic tables with dimension attributes $S_D \subseteq S$ and $S'_D \subseteq S'$ respectively, $S_D^{top} p$ denote the highest attributes in S_D and Y^{top} denote the highest attributes in $\mathrm{Y} = S_D \cap S'_D$. Let $T_r(S_r)$ be the result of a binary query between T and T' and $\mathrm{A}' \in S_r \cap S$ be an attribute of T with aggregable property $\mathrm{agg}_{\mathrm{A}'}(\mathrm{F}, \mathrm{X})$ holding in T. Then, for all queries Q satisfying the conditions of Table 28, the aggregable property $\mathrm{agg}_{\mathrm{A}}(\mathrm{F}, \mathrm{X}')$ holds in T_r and is such that for all Z where $\mathrm{S}_D - \mathrm{X}' \subseteq \mathrm{Z}$, A' is G-summarizable in T with respect to query Q, grouping set Z, and function F.

Proof. As for unary queries, the proof mainly consists in defining the "new" X' as the "complement" of Z as defined in Proposition 8 where X' is replaced by its definition in Table 19. For dimension attributes A', we can show that we always obtain a new X' which is equal to the old X' defined in Table 19. For example, for right-merge and full-merge queries, if A' is a dimension attribute in S_D, $\mathrm{X}' = \mathrm{X} \cup S'_D - \mathrm{Y} - \mathrm{X}'_f$ and in the first case where $\mathrm{Z} = (S_D - \mathrm{X}') \cup \mathrm{Y}^{top}$, we obtain $\mathrm{Z} = S_D - (\mathrm{X} \cup S'_D - \mathrm{Y} - \mathrm{X}'_f) \cup \mathrm{Y}^{top} = S_D - (\mathrm{X} \cup S'_D - \mathrm{Y} - \mathrm{X}'_f - \mathrm{Y}^{top}) = S_D - (\mathrm{X} \cup S'_D - \mathrm{Y} - \mathrm{X}'_f)$ which we also obtain in the second case: $\mathrm{Z} = (S_D - \mathrm{X}') \cup \mathrm{Y} = S_D - (\mathrm{X} \cup S'_D - \mathrm{Y} - \mathrm{X}'_f - \mathrm{Y}) = S_D - (\mathrm{X} \cup S'_D - \mathrm{Y} - \mathrm{X}'_f)$.

For measure attributes, $\mathrm{Z} = \mathrm{S}_D - \mathrm{X}'$ and $\mathrm{X}' = \mathrm{X}$, we obtain the new $\mathrm{X}' = \mathrm{X}$. □

We make the following observations on the rules of Table 28 and Table 29. First, all rules refine the conditions and actions of the propagation rules of Table 21 by taking into account the restrictions described in Proposition 8. Second, in the case of a right-merge, full-merge, union and difference query, it is possible to search for any subset $\mathrm{Y}' \subseteq \mathrm{Y}$ instead of Y^{top} for which the conditions on Y^{top} hold. If no such Y' is found then the set of attributes Y must be

removed from X' (we illustrated that in Example 21). Third, note that the propagation rule for right-merge assumes that A' is an exclusive attribute of T. If A' is also in T' then its aggregable property is computed using the propagation rule for left-merge.

Table 28. Propagation rules for merge operations with G-summarizability

Merge query on $T(S)$ and $T'(S')$	Propagation rule for inferring the aggregable properties of attributes $\mathrm{A}' \in S$ in the result $T_r(\mathrm{S}_r)$	*User action*
$T_r = T \rtimes_{\mathrm{Y}} T'$	if $\mathrm{Y} \not\mapsto \mathrm{S}'$ then $\mathrm{F} \in \{\mathrm{COUNT\_DISTINCT}, \mathrm{MIN}, \mathrm{MAX}\}$	
	dimension attribute $\mathrm{A}' \in S_D$ and $\mathrm{agg}_{\mathrm{A}'}(\mathrm{F}, \mathrm{X})$ holds in T: $\mathrm{agg}_{\mathrm{A}'}(\mathrm{F}, \mathrm{X}')$ holds in T_r with $\mathrm{X}' = \mathrm{X} \cup (S'_D - \mathrm{X}'_f)$ and $\mathrm{X}'_f = \mathrm{X}_f$	*Complete* X'_f
	measure attribute $\mathrm{A}' \in S - S_D$ and $\mathrm{agg}_{\mathrm{A}'}(\mathrm{F}, \mathrm{X})$ holds in T : $\mathrm{agg}_{\mathrm{A}'}(\mathrm{F}, \mathrm{X}')$ holds in T_r with $\mathrm{X}' = \mathrm{X}, \mathrm{X}'_d = \mathrm{X}_d$ and $\mathrm{X}'_f = \mathrm{X}_f$.	*Complete* X'_f

Table 28. Propagation rules for merge operations with G-summarizability

Merge query on $T(S)$ and $T'(S')$	Propagation rule for inferring the aggregable properties of attributes $A' \in S$ in the result $T_r(S_r)$	*User action*
	if $Y \not\mapsto S'$ then $F \in \{\text{COUNT\_DISTINCT}, \text{MIN}, \text{MAX}\}$	
$T_r = T \bowtie_Y T'$ $T_r = T \rtimes_Y T'$	dimension attribute $A' \in Y$ and $\text{agg}_{A'}(F, X)$ holds in T: if $\pi_Y(\sigma_{Y^{top}=y}(T)) = \pi_Y(\sigma_{Y^{top}=y}(T'))$ or $\pi_Y(\sigma_{Y^{top}=y}(T)) = \emptyset$ for all non-empty partitions $\sigma_{Y^{top}=y}(T')$: then $\text{agg}_{A'}(F, X')$ holds in T_r with $X' = X \cup (S'_D - X'_f) - Y^{top}$ and $X'_f = X_f$. else $\text{agg}_{A'}(F, X')$ holds in T_r with $X' = X \cup (S'_D - X'_f) - Y$ and $X'_f = X_f$.	*Complete* X'_f
	dimension attribute $A' \in S_D - Y$ and $\text{agg}_{A'}(F, X)$ holds in T: if $\pi_Y(\sigma_{Y^{top}=y}(T)) \subseteq \pi_Y(\sigma_{Y^{top}=y}(T'))$ or $\pi_Y(\sigma_{Y^{top}=y}(T)) = \emptyset$ for all non-empty $\sigma_{Y^{top}=y}(T') \neq \emptyset$: then $\text{agg}_{A'}(F, X')$ holds in T_r with $X' = X \cup (S'_D - X'_f)$ else $\text{agg}_{A'}(F, X')$ holds in T_r with $X' = X \cup (S'_D - X'_f) - Y$ and $X'_f = X_f$.	*Complete* X'_f
	measure attribute $A' \in S - S_D$ and $\text{agg}_{A'}(F, X)$ holds in T : if $\pi_Y(\sigma_{Y^{top}=y}(T)) \subseteq \pi_Y(\sigma_{Y^{top}=y}(T'))$ or $\pi_Y(\sigma_{Y^{top}=y}(T)) = \emptyset$ for all non-empty $\sigma_{Y^{top}=y}(T') \neq \emptyset$: then $\text{agg}_{A'}(F, X')$ holds in T_r with $X' = X$, $X'_d = X_d$ and $X'_f = X_f$.	*Complete* X'_f
	else $\text{agg}_{A'}(F, X')$ holds in T_r with $X' = X - Y$ and $X'_f = X_f$.	Recompute X'_d
	if $Y \not\mapsto S'$ then $F \in \{\text{COUNT\_DISTINCT}, \text{MIN}, \text{MAX}\}$	
$T_r = T \bowtie_Y T'$	dimension attribute $A' \in S_D$ and $\text{agg}_{A'}(F, X)$ holds in T: if $\pi_Y(\sigma_{Y^{top}=y}(T)) \subseteq \pi_Y(\sigma_{Y^{top}=y}(T'))$ or $\pi_Y(\sigma_{Y^{top}=y}(T)) = \emptyset$ for all non-empty partitions $\sigma_{Y^{top}=y}(T') \neq \emptyset$: then $\text{agg}_{A'}(F, X')$ holds in T_r with $X' = X \cup (S'_D - X'_f) - Y^{top}$ and $X'_f = X_f$. else $\text{agg}_{A'}(F, X')$ holds in T_r with $X' = X \cup (S'_D - X'_f) - Y$ and $X'_f = X_f$.	*Complete* X'_f

Table 28. Propagation rules for merge operations with G-summarizability

Merge query on $T(S)$ and $T'(S')$	Propagation rule for inferring the aggregable properties of attributes $\mathrm{A}' \in S$ in the result $T_r(\mathrm{S}_r)$	*User action*
	measure attribute $\mathrm{A}' \in S - S_D$ and $\mathrm{agg}_{\mathrm{A}'}(\mathrm{F}, \mathrm{X})$ holds in T : if $\pi_{\mathrm{Y}}(\sigma_{\mathrm{Y}^{top}=y}(T)) \subseteq \pi_{\mathrm{Y}}(\sigma_{\mathrm{Y}^{top}=y}(T'))$ or $\pi_{\mathrm{Y}}(\sigma_{\mathrm{Y}^{top}=y}(T)) = \emptyset$ for all non-empty partitions $\sigma_{\mathrm{Y}^{top}=y}(T') \neq \emptyset$: then $\mathrm{agg}_{\mathrm{A}'}(\mathrm{F}, \mathrm{X}')$ holds in T_r with $\mathrm{X}' = \mathrm{X} - \mathrm{Y}^{top}$ and $\mathrm{X}'_f = \mathrm{X}_f$. else $\mathrm{agg}_{\mathrm{A}'}(\mathrm{F}, \mathrm{X}')$ holds in T_r with $\mathrm{X}' = \mathrm{X} - \mathrm{Y}$ and $\mathrm{X}'_f = \mathrm{X}_f$	*Recompute* X'_d *Complete* X'_f

4.5 Wrapping up results on summarizability

To wrap up our results on summarizability and G-summarizability, we illustrate them using the motivating example presented in the introduction of this paper. We then discuss some directions for future work around the generation of explanations associated with the result of an analytic query.

Table 29. Propagation rules for set operations with G-summarizability

Set query on $T(S)$ and $T'(S')$	Propagation rule for inferring the aggregable properties of attributes $\mathrm{A}' \in S$ in the result $T_r(\mathrm{S}_r)$	*User action*
$T_r = T \cup T'$	dimension attribute $\mathrm{A}' \in S_D$ and $\mathrm{agg}_{\mathrm{A}'}(\mathrm{F}, \mathrm{X})$ holds in T and T': if $\pi_{\mathrm{S}_D^{top}}(T) \cap \pi_{\mathrm{Y}^{top}}(T') = \emptyset$ then $\mathrm{agg}_{\mathrm{A}'}(\mathrm{F}, \mathrm{X}')$ holds in T_r with $\mathrm{X}' = \mathrm{X} - \mathrm{Y}^{top}$ and $\mathrm{X}'_f = \mathrm{X}_f$	*Recompute* X'_d
	measure attribute $\mathrm{A}' \in \mathrm{S}$ and $\mathrm{agg}_{\mathrm{A}'}(\mathrm{F}, \mathrm{X})$ holds in T and T': if $\pi_{\mathrm{Y}^{top}}(T) \cap \pi_{\mathrm{Y}^{top}}(T') = \emptyset$ then if $\mathrm{X}_d \mapsto \mathrm{A}'$ holds in T_r then $\mathrm{agg}_{\mathrm{A}'}(\mathrm{F}, \mathrm{X}')$ holds in T_r with $\mathrm{X}' = \mathrm{X} - \mathrm{Y}^{top}$ and $\mathrm{X}'_f = \mathrm{X}_f$.	*Recompute* X'_d
	else $\mathrm{agg}_{\mathrm{A}'}(\mathrm{F}, \mathrm{X}')$ holds in T_r with $\mathrm{X}' = \mathrm{X} - \mathrm{Y}^{top}$, $\mathrm{X}'_d = \mathrm{X}_d$ and $\mathrm{X}'_f = \mathrm{X}_f$	*Minimize* X'_d
$T_r = T - T'$	$\mathrm{A}' \in S_r$ and $\mathrm{agg}_{\mathrm{A}'}(\mathrm{F}, \mathrm{X})$ holds in T and T' : if all partitions $\sigma_{\mathrm{Y}^{top}}(T)$ are equal to or disjoint with $\sigma_{\mathrm{Y}^{top}}(T')$ then $\mathrm{agg}_{\mathrm{A}'}(\mathrm{F}, \mathrm{X}')$ holds in T_r with $\mathrm{X}' = \mathrm{X} - \mathrm{Y}^{top}$ and $\mathrm{X}'_f = \mathrm{X}_f$	Recompute X'_d *None*

Example 22. Consider the example of interactive data analysis session of Figure 2 on the tables in Table 2. Table T3 is obtained by a filter query $\mathit{Filter}_{\mathrm{STORE\_SALES}}(P|\mathrm{COUNTRY,YEAR})$ and attribute AMOUNT in STORE_SALES is aggregable along all dimension attributes except attribute YEAR. By the G-summarizability rule for filter queries

for attribute AMOUNT in T3, we obtain the aggregable property $\text{agg}_{\text{AMOUNT}}(\text{SUM}, \text{X}_3)$ where X_3 now contains all attributes of STORE_SALES except COUNTRY and YEAR. Since the table T4 is obtained by summarizing AMOUNT by Z = CITY, STATE, COUNTRY, YEAR, the aggregate operation over T3 leading to T4 is *correct* with respect to the G-summarizability of AMOUNT.

Next, the table T5 is obtained by a merge query adding the attribute POP from table *DEM* to table T4. For attribute POP in T5, we have $\text{agg}_{\text{POP}}(\text{F}, \text{X}_5)$, where F and X_5 are defined according to Table 28 for right merge (*DEM* is the "outer" merge table). We have join attributes Y = {CITY, STATE, COUNTRY, YEAR} with Y^{top} = {COUNTRY, YEAR}. However, $\pi_{\text{Y}}(\sigma_{\text{COUNTRY='}USA\text{'}\wedge\text{YEAR=2018}}(\text{T4})) \subset \pi_{Y}(\sigma_{\text{COUNTRY='}USA\text{'}\wedge\text{YEAR=2018}}(\text{DEM}))$ (city of 'Palo Alto' is missing in T4). Thus, we have X_5 = Y in the above aggregable property. Consequently, the aggregation of SUM(POP) along CITY is **incorrect** with respect to the G-summarizability property.

Table *DEM′* in Table 6a is obtained by aggregating the population in table *DEM* along attribute CITY. In table *DEM*, we have $\text{agg}_{\text{POP}}(\text{SUM}, \text{X})$ with X = {CITY, STATE, COUNTRY}. So the aggregation along CITY leading to table *DEM′* is *correct*, and by Proposition 7, $\text{agg}_{\text{POP}}(\text{SUM}, \text{X}')$ holds in *DEM′* with $\text{X}' = \text{X} \cap \text{Y}$ = {STATE, COUNTRY} (function SUM is distributive). Next, in the merge result of T4 with DEM′, we have $\text{agg}_{\text{POP}}(\text{F}, \text{X}_4)$, where F and X_4 are defined by the rule for outer merge (*DEM′* is the outer merge table). We have Y = {STATE, COUNTRY, YEAR} and Y^{top} = {COUNTRY, YEAR}. We also have $\pi_{Y}(\text{DEM} \ltimes_{\text{Y}^{top}} \text{T4}) = \pi_{Y}(\text{DEM})$, and since $\text{agg}_{\text{POP}}(\text{SUM}, \text{X}')$ holds in DEM′, we get $\text{X}_4 = X' - \text{Y}^{top}$ = {STATE}. Finally, since Y does not literally determine the schema of T4, F must be restricted to one of {COUNT_DISTINCT, MIN, MAX}.

The question that naturally arises is what options should be provided to the end user when G-summarizability is violated (i.e., the conditions given by aggregable properties are not satisfied). A first option is to reject an incorrect aggregate query with respect to G-summarizability and return the grouping set of the aggregable property as an explanation. This is the option we have described in this paper. However, another option could be to accept the aggregate query provided that some metadata is added to the resulting table to enable a non-ambiguous and correct interpretation of the tuples in that table. We explain the idea in the next example and leave it for future work.

Example 23. Consider again the interactive session of Figure 2 and suppose that the first filter query on STORE_SALES is: STATE ≠ *null* (expecting that this eliminates all European countries) and YEAR = '2018'. Next, suppose that the aggregate operation over T3 sums AMOUNT for each partition defined by attributes COUNTRY and YEAR. It will be difficult for an end user to figure out that an incorrect aggregate value has been computed for country 'USA' if the user ignores that 'Washington DC' has no state. With our current proposition, this aggregate operation will be rejected since the grouping set does not include attribute STATE (we would have $\text{agg}_{\text{AMOUNT}}(\text{SUM}, \{\text{CITY}, \text{COUNTRY}\})$ in T3). However, to disambiguate the result of the aggregate operation, it would be sufficient to "attach" the filter condition STORE_SALES.STATE ≠ *null* as metadata to table T4 to indicate that the amount for stores in those states has not been accounted for. It is then possible for the end user to query those stores from table STORE_SALES to visualize them and decide if table T4 is satisfactory.

The same principle applies to the result of the left-merge operation between T4 and DEM. As we have seen in the previous example, we have $\text{agg}_{\text{POP}}(\text{SUM}, \text{X}_5)$ in T5 and the aggregate operation on T5 is rejected. However, we could accept the operation and simply mark that measure attribute SUM(POP) now depends on dimension *SALESORG*. This would indicate that the population is summed for the cities in *SALESORG*, that is, the cities that have stores.

The idea is therefore to make each analytic table, resulting from an interactive data analysis session, "self-explanatory" with respect to its aggregated attributes.

5 RELATED WORK

In this section, we focus on previous works that propose conditions on the schema of a fact table, or on the parameters of an aggregate query expressed over that fact table, to determine if the aggregate query returns a correct result with respect to some summarizability definition. Previous papers on summarizability use heterogeneous notations and concepts and are sometimes difficult to read because they lack some details or hide some assumptions. To facilitate comparisons with our work, we reformulated each previous proposition using the notations introduced in this paper.

In our detailed analysis, we establish the following:

(1) Our data model is more general than the data models considered by previous work.
(2) In the case of a sequence of two aggregate queries, Q_1 followed by Q_2, our sufficient conditions to determine if Q_2 is correct subsume the conditions proposed by previous work.
(3) To our best knowledge, no previous work addressed the case of a sequence made of an arbitrary analytic query followed by an aggregate query, which is addressed by our notion of G-summarizability.

5.1 Summarizability of a query over a statistical object

The notion of *summarizability* was initially defined by Rafanelli and Shoshani [43] for statistical databases and later refined in their seminal paper [30]. In their context, *base data*, also referred to as "micro-data", describe all the details about the objects or individuals over which a summarization operation can be applied to produce a so-called *statistical object*, also referred to as "macro-data". There are a few constraints. In the base data, an *object of interest* must be identified (e.g., a product, a customer, a store) using some attributes, all other attributes being viewed as "descriptors" of the object. A statistical object is a table defined by a summarized attribute (i.e., an attribute of the base data on which a summarization function is applied) and a set of "category" attributes defining the partitions of the base table on which the summarization function is applied. Using the terminology defined in Section 2, base data can be modelled as a non-analytic table and summarization operations are aggregate queries which ignore partitions with null values in their partition identifiers. We shall keep the expression "summarization operation" to distinguish it from our analytic aggregate operation that handles null values as regular values. A statistical object can be modeled as a fact table that results from a summarization operation over the base data where category attributes represent dimensions and the summarized attribute is a measure.

The fact tables that represent statistical objects in [30] are however more restricted than the fact tables enabled by our data model. Firstly, dimensions are restricted to *strict* hierarchies, that is, each attribute has at most one parent attribute in the hierarchy type, each attribute value of a dimension has at most one parent attribute value in the hierarchy of the dimension, and hierarchy types must have a single bottom and top level attribute. Secondly, all dimensions in a fact table must be independent (that is, no attribute in some dimension functionally depends on an attribute of another dimension). Finally, all facts in a fact table have the same dimensions, i.e., the measure attribute does not depend on a subset of the dimensions.

An important distinction, with respect to multidimensional data models, is that there is no notion of managed dimensions in [30, 43]. The notion of dimension hierarchy is purely local to a statistical object and depends on the functional dependencies that are supposed to hold in the base data on which the object is built. If these dependencies change, the dimensions hierarchies are adjusted to fit the strictness constraint explained before.

In [30, 43], summarizability is defined as the property of a summarized attribute $\mathrm{F}(\mathrm{A})$ of a fact table (statistical object) T which guarantees that a summarization operation G over $\mathrm{F}(\mathrm{A})$produces a *correct* result. Suppose we have a

base table T_0 and a fact table $T(S)$, which results from a summarization operation applying aggregation function F on attribute A along attributes X:

$$T = \mathcal{A}gg^*_{T_0}(\mathrm{F}(A) \mid \mathrm{X})$$

where $\mathcal{A}gg^*$ denotes a summarization operation that does not consider partitions where an attribute of X has a null value. Let $\mathcal{Q}$ be a summarization query over T:

$$\mathcal{Q} = \mathcal{A}gg^*_T(\mathrm{G}(\mathrm{F}(\mathrm{A}))|\mathrm{Z})$$

where G is a function applicable to the summarized attribute F(A) of T, and Z is a set of dimension attributes in X. Then, the summarization query $\mathcal{Q}$ is said to be *correct*, if the following condition holds:

$$\mathcal{A}gg^*_{T_0}(F(\mathrm{A})|\mathrm{Z}) = \mathcal{A}gg^*_T(G(F(\mathrm{A}))|\mathrm{Z})$$

To guarantee the correctness of summarization query $\mathcal{Q}$, [30] defines three necessary properties on the summarization query $\mathcal{Q}$ and the dimensions D of T_0. Let $\mathrm{X}_D \subseteq \mathrm{X}$ be the set of dimension attributes for dimension D in T, $\mathrm{X}_D^{bot} \in X_D$ be the bottom level attribute of D in T and D^{bot} be the bottom level attribute of D in T_0 (then, $D^{bot} = X_D^{bot}$ or $D^{bot} \preccurlyeq^* X_D^{bot}$). The properties are:

(1) *Disjointness.* For each dimension D along which summarization is done in $\mathcal{Q}$, at least one of the following conditions must hold: (a) $\mathrm{X}_D - Z = \{D^{bot}\}$ consists of the bottom level attribute of D and the partitions of T_0 using D^{bot} are disjoint with respect to the identifier attributes of the object of interest in T_0; (b) every value in T_0 of a dimension attribute A_1 that is a below an attribute A_2 in $\mathrm{X}_D - Z$ in D ($\mathrm{A}_1 \preccurlyeq^* \mathrm{A}_2$) must map to a single value of its parent attribute (many-to-one mapping);
(2) *Completeness with respect to* $F(\mathrm{A})$. For each dimension D of T, the domain of each attribute in X_D is *complete* in T_0, if both of the following conditions hold: (a) all values of the identifier attributes of the object of interest which are required by $F(\mathrm{A})$ appear in T_0, and (b) one of the two following conditions holds: if $\mathrm{X}_D - Z = \{D_{bot}\}$ consists of the bottom level attribute of D in T_0 then the value of this attribute cannot be *null* in T_0. Otherwise, every value of a dimension attribute that is a child of X_D^{bot} must map to a parent value in X_D^{bot} within T_0.
(3) *Applicable summary function.* The summary function G is "applicable" to the summarized attribute $\mathrm{F}(A)$ with respect to all dimensions along which summarization is done in $\mathcal{Q}$.

We now analyze each one of the previous conditions and relate it to our work.

Disjointness: In the original formulation of [30], the disjointness property requires that the dimension attributes along which summarization is done form disjoint subsets over the "objects of interest" defined in the base table. Two different disjointness conditions are then given, depending on whether the dimension attribute is a bottom attribute of the dimension or not. The goal of the disjointness property is mainly to avoid double counting by overlapping subsets. In our work, we address the disjointness property by defining aggregable properties and propagation rules. Each aggregable property describes for a given attribute A in fact table T_0 along which attributes it can be aggregated using some function F. The summarizability preserving propagation rules then produce all aggregable properties of $\mathrm{F}(A)$ in T after the aggregation operation on T_0 is done (see Proposition 2). Summarizability is defined using the notion of function distributivity and literal functional dependencies. There is no need to choose an object of interest in the base data for defining the scope of summarizability, and any attribute can be summarized.

Completeness: In completeness condition (2.a) of [30], it is not clear how the set of "all possible values" for the identifier attributes is determined to assess completeness. Thus, we used the interpretation that the possible values are those listed in some reference directory. Furthermore, condition (2.a) applies to the identifier values required for computing $F(A)$ which means that the user who is formulating query Q must decide whether completeness is needed or whether the values listed in T are sufficient to compute a summary attribute. We shall see in Example 25 that this condition (2.a) is useless for checking summarizability. The Item (2.b) of the completeness condition is required because summarization operations cannot deal with attributes of Z that have null values. In our work, we do not have such a restriction since our SQL aggregate operations handle null values as regular values.

Function applicability: The third condition focuses on testing the compatibility between the type of dimensions and the type of measures used in T. The following types of measures can be used by the designer of a fact table: *stock* (i.e., a simple value at a particular point in time), *flow* (i.e., cumulative values over a period of time) and *value-per-unit* (i.e, determined value for a fixed time). Dimensions can be of type temporal or non-temporal. When a measure attribute A is aggregated using a given function over some dimension D, the types of A and D should be compatible with respect to that function. In our work, applicable functions are captured by the more general notion of aggregable property, which leaves the method to decide which function is applicable to an attribute open. Furthermore, we use propagation rules and default rules to infer the functions that are applicable to an attribute that has been aggregated. Thus, a user is not forced to define the type of the measure attributes in T since aggregable properties for these attributes will be automatically computed using propagation rule (see Proposition 2).

The following two examples illustrate the conditions of [30] and emphasize the differences with our work.

Table 30. Table PRODUCT_LIST

PROD_SKU	BRAND	COUNTRY	YEAR	QTY
cz-tshirt-s	Coco Cola	USA	2017	5 000
cz-tshirt-s	Coco Cola	USA	2018	7 000
cz-tshirt-s	Zora	Spain	2017	5 000
cz-tshirt-s	Zora	Spain	2018	7 000
coco-can-33cl	Coco Cola	USA	2017	10 000

Example 24. Consider the base table PRODUCT_LIST (PROD_SKU, COUNTRY, BRAND, YEAR, QTY), whose instance is displayed on Table 30, and where the "object of interest" is a product identified by PROD_SKU. To comply with the constraint of strict dimensions defined by [30], attribute PROD_SKU must belong to a separate dimension (it determines no other attribute), attributes BRAND, COUNTRY belong to a dimension *MKT_PROD*, and attribute YEAR belongs to a dimension *TIME*.

Next, suppose that we define a statistical object PRODUCT_SUM (BRAND, YEAR, NB_PROD_SKU) built using a summarization query with function F = COUNT_DISTINCT:

$$\text{PRODUCT\_SUM} = \mathcal{A}gg^{*}_{PRODUCT\_LIST}(\text{COUNT\_DISTINCT}(\text{PROD\_SKU})|\text{BRAND, YEAR})$$

The result is displayed on Table 31 (we renamed the summarized attribute as NB_PROD_SKU). The user should associate the summarized attribute NB_PROD_SKU with a type *flow* since the number of distinct products sold is cumulative over time.

Table 31. Table PRODUCT_SUM

BRAND	YEAR	NB_PROD_SKU
Coco Cola	2017	2
Coco Cola	2018	1
Zora	2017	1
Zora	2018	1

The first summarization query Q_1 aggregates NB_PROD_SKU along YEAR using function G = SUM to count the number of distinct products by brand :

$$Q_1 = \mathcal{A}gg^*_{PRODUCT\_SUM}(\text{SUM}(\text{NB\_PROD\_SKU})|\text{BRAND})$$

Since YEAR is a bottom attribute of dimension $D = TIME$ in T_0, Item (1.a) of the disjointness condition must be tested over T_0. It fails because the product "cz-tshirt-s" belongs to two different partitions of T_0 = PRODUCT_LIST by YEAR. Thus, query Q_1 is incorrect. Indeed, the number of distinct products by brand reported by query Q_1 (e.g., value 2 for brand Zora) would be different from the number directly computed from PRODUCT_LIST (e.g., 1 for brand Zora).

The second summarization query Q_2 aggregates NB_PROD_SKU along BRAND:

$$Q_2 = \mathcal{A}gg^*_{PRODUCT\_SUM}(\text{SUM}(\text{NB\_PROD\_SKU})|\text{YEAR})$$

Since BRAND is again a bottom attribute of dimension *MKT_PROD* in T_0, Item (1.a) of the disjointness condition must be tested. It fails since the product "cz-tshirt-s" maps to different partitions of T_0 by BRAND. Thus, query Q_2 is also incorrect. Again, the number of distinct products by year reported by query Q_2 (e.g., value 2 for year 2018) would be different from the number directly computed from PRODUCT_LIST (e.g., 1 for year 2018).

By comparison with our work, let's assume that the aggregable property $\text{agg}_{\text{PROD\_SKU}}(\text{COUNT\_DISTINCT} \mid \text{Z})$, where Z = {BRAND, COUNTRY, YEAR}, has been validated by the designer of table PRODUCT_LIST. Then, by Proposition 7, we infer that the aggregable property controlling summarizability, $\text{agg}_{\text{NB\_PROD\_SKU}}(\text{SUM} \mid \emptyset)$, holds in PRODUCT_SUM. Therefore, we also detect that both Q_1 and Q_2 are incorrect. However, our detection does not require running any query over the base data, unlike the disjointness condition of [30]. We only use the knowledge of the attribute graphs of the dimensions and of the aggregable properties defined on fact tables.

Example 25. Suppose that we have a base object, represented by table STORE_SALES introduced earlier (see Table 8d), in which stores are the objects of interest (identified by STORE_ID). A statistical object, modeled by fact table STORE_SALES_YEARLY (displayed in Table 32a), is computed using the following summarization query:

$$\text{STORE\_SALES\_YEARLY} = \mathcal{A}gg^*_{STORE\_SALES}(\text{SUM}(AMOUNT)|\text{CITY, STATE, COUNTRY, YEAR})$$

We assume that the summarized attribute SUM(AMOUNT) is renamed as AMOUNT and has been associated with type *flow*, i.e., it can be summed along any dimension. To fulfill the constraints on dimensions, there will be four dimension hierarchies respectively formed of the following attributes: {CITY}, {STATE}, {COUNTRY}, and {YEAR}.

Consider the following summarization query Q_3 whose result is displayed in Table 32b:

$$Q_3 = \mathcal{A}gg^*_{STORE\_SALES}(\text{SUM}(AMOUNT)|\text{COUNTRY, YEAR})$$

Table 32. Results of summarization queries

(a) STORE_SALES_YEARLY

CITY	STATE	COUNTRY	YEAR	AMOUNT
Dublin	Ohio	USA	2017	3.2
Dublin	california	USA	2017	5.3
Dublin	Ohio	USA	2018	8.2
Dublin	California	USA	2018	6.3

(b) Result of Q_3(STORE_SALES_YEARLY)

COUNTRY	YEAR	SUM(AMOUNT)
USA	2017	8.4
USA	2018	14.5

The disjointness condition is obviously satisfied since a store in table STORE_SALES belongs to a single partition by CITY and a single partition by STATE. Now, assume that the directory of all possible values for STORE_ID is given by the dimension table *SALESORG* of Table 8d. Then, the completeness condition (2.a) is violated since store *Ca*_02 is missing in the list of values of STORE_ID in STORE_SALES. However, from a strict summarizability point of view, it is easy to see that the result of Q_3 is the same as the result of the same aggregation executed over table STORE_SALES. So completeness constraint (2.a) is useless for checking summarizability.

Completeness constraint (2.b) is also violated since bottom attribute STATE, has a *null* value in STORE_SALES for city 'Paris'. Hence, there is no tuple for 'Paris' in STORE_SALES_YEARLY and therefore also no tuple for country 'France' in the result of Q_3. The summarizability property is thus clearly violated since the result of Q_3 applied to STORE_SALES returns a different result containing the tuple $(France, 2017, 45.1)$. In our work, STORE_SALES_YEARLY would contain a tuple for city 'Paris' with a null value for STATE, and therefore a tuple for country 'France' in the result of Q_3.

5.2 Summarizability of attributes in fact tables

In previous works on multidimensional databases, summarizability is expressed over fact tables in a way similar to our Definition 10 illustrated by Figure 9. In this context, we first review the work of [39, 40] that uses a multidimensional data model close to ours, and generalizes the summarizability conditions of [30]. Note that other solutions propose alternative analytic data models and methods to modify the representation of dimensions to enforce summarizability. In the following, we do not consider these solutions for which a survey can be found in [34]. Thus, like in our work, we concentrate on summarizability models which characterize correct compositions of two aggregation queries over fact tables with fixed dimensions.

Given a summarization query $Q = \mathcal{A}gg^*_T(F(\mathrm{A})|\mathrm{X})$ over a fact table $T(\mathrm{S})$, we define *summarizability* as a property of an attribute A with respect to grouping set X and function F. More exactly, [40] considers an attribute A to be summarizable with respect to X and F if for any subset $Z \subset \mathrm{X}$ of dimension attributes, the following condition holds:

$$\mathcal{A}gg^*_T(\mathrm{F}(\mathrm{A})|\mathrm{Z}) = \mathcal{A}gg^*_Q(\mathrm{F}(\mathrm{F}(\mathrm{A}))|\mathrm{Z}) \tag{4}$$

Note that this definition is more restrictive than our notion of summarizability in Definition 10, since it imposes that the same function F is used in the two queries. The following conditions provided by [40] are sufficient for ensuring Equation (4). Let X_D denote the set of dimension attributes for dimension D in X:

(1) Function F is *applicable* to A,
(2) Function F is *distributive* over the domain of values in A.

(3) For every dimension D, all the value mappings between the bottom level attribute of D in T and any attribute of X_D are many to one.
(4) For every dimension D, every value of an attribute in T that is a sub-level of an attribute of X_D must be non null.

We now comment each condition and draw comparisons with our work. In the first condition, [40] assumes that we know the functions that are applicable to A. Three types of aggregation functions are distinguished: (1) functions, which are applicable to data that can be added together (e.g., SUM, COUNT, AVG), (2) functions, which are applicable to data that can be used for average calculations (e.g., COUNT, AVG, MIN, MAX), and (3) functions which are applicable to data that can only be counted. However, unlike our work, the notion of aggregation type does not consider the dimensions along which a summarization can be performed. The second condition uses a definition of distributive function that is more restrictive than our Definition 11 because it requires that F is such that for any two sets, V_1 and V_2, $F(V_1, \cup V_2) = F(F(V_1) \cup F(V_2))$. Thus, functions like COUNT are discarded. The third condition is similar to the disjointness condition of [30] which we already compared with our work. The fourth condition is equivalent to the second completeness condition (2.b) of [30] but does not cover the first completeness condition (2.a). As mentioned before, aggregation functions in our work handle null values in dimensions as regular values, and can ignore condition (2.b) to guarantee summarizability.

5.3 Multidimensional normal forms

Several works proposed multidimensional normal forms for analytic tables that provide guarantees for the correctness of summarization queries [27, 28]. These normal forms can be used to design dimension and fact tables over which correct summarization queries can be easily detected and evaluated. In the sequel, we are not interested in the design aspects but we examine the definitions of these normal forms as a way to formulate summarizability conditions.

We first introduce a few concepts and vocabulary. A dimension attribute (also called a dimension level) which can have a *null* value is called *optional* and otherwise called *mandatory*. Dimensions must have a single bottom level type and an implicit top level type, called *ALL*. The hierarchy in each dimension is strict using "functional dependencies with nulls" (NFD), that is, each attribute A_i has at most one parent attribute A_j and each arc (A_i, A_j) in the attribute graph of the dimension is labelled with a 1 (Section 2.2).

The novelty of [27, 28] in comparison with [40] is the following. A dimension is also associated with a (possibly empty) set of *context dependencies*: let A_i and A_j be two dimension attributes of a dimension D such that A_i is optional, $A_j \neq ALL$, and $A_i \preccurlyeq A_j$. If $c \in dom(A_j)$ and $c \neq null$, then (A_i, A_j, c) is a context dependency for D stating that for every tuple T of the dimension table, $t.A_j = c \Leftrightarrow t.A_i \neq null$. Intuitively, the interpretation of a context dependency is that A_j plays the role of a discriminating attribute in the hierarchy and value c is the discriminating value to indicate when the optional attribute A_i has a non-null value. Note that the use of an equivalence ($\Leftrightarrow$), in the above formula, is quite strong since it forces the existence of a *single* discriminating value.

In [27, 28], a fact table is defined over a set of dimensions at the finest level of detail, that is, all dimension attributes of a dimension are included in the schema of the fact table, and each measure in the fact table is determined (with NFD constraints) by the set of bottom level dimension attributes in each dimension. Thus, fact tables are similar to the "micro data" of [30]. Note that although fact tables in [40] can represent facts at a coarser granularity, their summarizability conditions impose that the bottom level dimension attributes of each dimension determine (with NFD constraints) all measure attributes. In [27], *summarizability constraints* express the dimension hierarchy along which a measure can be aggregated using some aggregation function. However, no formal treatment of summarizability constraints is provided,

unlike our use of aggregable properties for analyzing aggregate queries and propagating summarizability constraints to query results. In [28], the same categorization of measure attributes as [30] is used.

In [28], two different multidimensional normal forms are presented. The first one is called *Multidimensional Normal Form (MNF)* and provides conditions similar to the work of [30] and [40]. The second one, called *Generalized Multidimensional Normal Form (GMNF)*, provides more general conditions which have been slightly extended in [27].

In the following we will describe the GMNF as defined by [27]. Let T be a fact table defined over a set of dimensions with a measure (summary) attribute A. Then T is in GMNF if all of the following conditions are satisfied:

(1) For each dimension D of T:
  (a) for every optional dimension attribute A_i of D, there exists a context dependency (A_i, A_j, c) in D;
  (b) the values of the bottom level dimension attribute in T are complete.
(2) All dimensions are mutually independent, i.e., there exists no NFD between any two dimension attributes of two distinct dimensions.
(3) The set of (unique) bottom level attributes of all dimensions functionally determines (FD) attribute A.

We comment these conditions. Condition 1a and condition 3 enforce the third and fourth conditions of [40] presented in Section 5.2. Indeed, if all dimension attributes are mandatory, they cannot have null values and all dependencies between dimension attributes become functional dependencies. Condition 1b is analogous to the completeness condition of [30]. Here again, the means to test this requirement are left unspecified and seem to require some external knowledge. Finally, condition 2 ensures that dimensions do not share dimension attributes and is not really needed for guaranteeing summarizability. Note that by conditions 2 and 3 the bottom level dimension attributes in the schema of T form a primary key in T and there is no other primary key for A. The novelty with respect to the previous models is brought by 1a. It constrains the semantics of every optional dimension attribute A_i so that there exists at an upper level an attribute A_j that plays the role of discriminator for A_i. Note that A_j can itself be an optional attribute, in which case there will again be a context dependency (A_j, A_k, c') in D. Eventually, the discriminator attribute will be a mandatory attribute since by definition of context dependency, the upper level attribute cannot be *ALL*.

We can now reformulate the previous GMNF conditions on dimensions as summarizability conditions on measure attributes and aggregate queries. Let T be a fact table in GMNF, and $Q = \mathcal{A}gg^*_T(\mathrm{F}(\mathrm{A})|\mathrm{X})$ be a summarization query over T. Then, A is *summarizable* with respect to function F and grouping set X, *i.e.*, for all $\mathrm{Z} \subseteq \mathrm{X}$: $\mathcal{A}gg^*_T(\mathrm{F}(\mathrm{A})|\mathrm{Z}) = \mathcal{A}gg^*_Q(\mathrm{F}(\mathrm{F}(\mathrm{A}))|\mathrm{Z}))$ if the following conditions hold:

(1) F is applicable to A with respect to any subset of X in the result of Q.
(2) One of the two conditions hold:
  (a) X does not contain any optional dimension attribute, or
  (b) if X contains an optional attribute A_i then, assuming that (A_i, A_j, c) is the associated context dependency, a filter condition: $A_j = c$ must be applied on T before the summarization query is applied

Here, the condition 1 is determined using either the categorization of A [28] or the summarizability constraints on A [27], the latter being more precise as we stated earlier. The condition 2 states that summarizability holds provided that a (filtered) subset of the fact table is considered, and this subset is given by the context dependencies of the dimensions over which the fact table is defined.

Example 26. Consider a product dimension, *PROD_NEW*, whose attribute graph is depicted in Figure 12. A new optional attribute VIDEO_RES provides the video resolution for products with screens where VIDEO_RES $\preccurlyeq$ CATEGORY and

Table 33. PROD_NEW_SALES

PROD_SKU	SUBCATEGORY	CATEGORY	VIDEO_RES	BRAND	AMOUNT
p_01	Video projector	video	1920x1080	Epson	42
p_02	TV	video	3840x2160	Philips	58
p_05	TV	video	3840x2160	Samsung	90
p_03	Radio	audio	-	Philips	45
p_04	CD-player	audio	-	Samsung	5

PROD_SKU $\preccurlyeq$ VIDEO_RES. Consider the fact table PROD_NEW_SALES over *PROD_NEW* whose instance is displayed in Table 33.

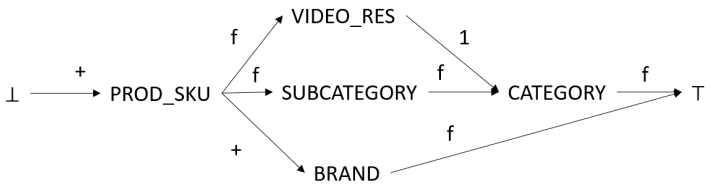


Fig. 12. Attribute graph of *PROD_NEW*

Assume that the designer of the dimension table *PROD_NEW* defined a context dependency : (VIDEO_RES, CATEGORY, 'video'). Then, table PROD_NEW_SALES is in GMNF if we assume that all products are listed in the table. It is easy to see that conditions 2 and 3 are satisfied on the attribute graph. Condition 1 is also satisfied because the only optional attribute VIDEO_RES has an upper level discriminating attribute with value 'video'.

Consider a summarization query $Q = \mathcal{A}gg^*_{\text{PROD\_NEW\_SALES}}(SUM(\text{AMOUNT}) \mid X)$. Then SUM is still applicable to the resulting attribute SUM(AMOUNT). If X = {CATEGORY, BRAND} then AMOUNT is summarizable with respect to SUM and X because X only contains mandatory attributes. If X = {VIDEO_RES, BRAND} then, by summarizability condition 2b, PROD_NEW_SALES must be first filtered with a filter: CATEGORY = 'Video' before applying Q. Afterwards, AMOUNT is summarizable with respect to SUM and X. Otherwise, because partitions with identifiers containing null values are ignored, a second query taking the sum along VIDEO_RES would generate an incorrect value for Philips (58 instead of 103) and Samsung (90 instead of 95).

We now compare GMNF with our work in the same context. Going back to Example 26, attribute AMOUNT is *literally* determined by the minimal subset of dimension attributes {PROD_SKU}, which also determines all other dimension attributes of PROD_NEW_SALES. Since SUM is applicable to AMOUNT, by Definition 9 on Page 19, $\text{agg}_{\text{AMOUNT}}(\text{SUM}, Z)$ holds in PROD_NEW_SALES, where Z is the set of all dimension attributes of PROD_NEW_SALES. Consider the query Q_1 of Example 26. By Proposition 5, attribute AMOUNT is summarizable with respect to SUM and X = {CATEGORY, BRAND} since X $\subset Z$ and SUM is distributive. Now if X = {VIDEO_RES, BRAND}, since X $\subset Z$, attribute AMOUNT is also summarizable with respect to SUM and X, without requiring any pre-filtering of PROD_NEW_SALES. The reason is our usage of SQL aggregation operations that considers null values as regular values. Indeed, it is easy to see that the summarizability condition is satisfied by looking at the result of Q, displayed in Table 34. To conclude the comparison, observe that fact table STORE_SALES of Table 8d is not in GMNF since condition 1 is violated. The optional STATE

has a null value for different countries and it is not possible to create a *single* context dependency for attribute STATE using either attribute COUNTRY or CONTINENT. Thus, in our work, by accepting partition identifiers with null values, we handle cases of summarizability that are rejected by the conditions based on GMNF.

Table 34. Query result of Q_2

VIDEO_RES	BRAND	AMOUNT
1920x1080	Epson	42
3840x2160	Philips	58
3840x2160	Samsung	90
-	Philips	45
-	Samsung	5

5.4 Reasoning over constraints on dimensions

In [20], the summarizability constraints on dimensions generalize the idea of context dependencies introduced in [27, 28]. The multidimensional data model is restricted as follows. All dimension hierarchies have one top level attribute called ALL and possibly multiple bottom level attributes. As in [27, 28, 40], a dimension attribute can have multiple parent dimension attributes in the hierarchy (such dimensions are called "heterogeneous"), and there can be both, mandatory and optional dimension attributes. Every child-parent attribute mapping should be functional (i.e., every value only maps to one parent value). This is equivalent to the existence of an NFD dependency from any attribute A_i to attribute A_j where $\mathrm{A}_i \preccurlyeq A_j$. As in [27, 28], fact tables are defined over dimensions at the finest level of detail, that is, the schema of the fact table includes the bottom level attributes of the dimensions. Measure attributes are determined by *all* the dimensions and can only be aggregated using distributive functions (defined as in Definition 11). Dimensions are also supposed to be mutually independent in a fact table. Summarizability is defined as a property of dimensions and any fact table built over summarizable dimensions has summarizable measures. Let D be a dimension, X be a subset of dimension attributes in D, and B a dimension attribute in D such that $\mathrm{A}_i \preccurlyeq \mathrm{B}$ for some attribute $\mathrm{A}_i \in \mathrm{X}$. Attribute B is *summarizable from* X *in* D if and only if for every fact table T defined over D, every measure attribute M of T, every set $\mathrm{X}' \subset \mathrm{X}$, and every distributive aggregate function F using G that is applicable to M, we have:

$$\mathcal{A}gg^*_T(F(M)|\mathrm{B}) = \mathcal{A}gg^*_{T'}(G(F(M))|\mathrm{B}) \text{ where } T' = \mathcal{A}gg^*_T(\mathrm{F(M)}|\mathrm{X}' \cup \mathrm{B}) \tag{5}$$

The above definition of summarizability relates to summarizability conditions as follows. In Definition 10, we consider a fact table $T' = \mathcal{A}gg^*_T(\mathrm{F(M)} \mid \mathrm{X})$ resulting from a summarization query over a fact table T, where F is a distributive function using G. Then we consider that *M is summarizable with respect to* X *and* F when $\mathcal{A}gg^*_T(\mathrm{F(M)}|\mathrm{Z}) = \mathcal{A}gg^*_{T'}(\mathrm{G(F(M))}|\mathrm{Z})$ for any $Z \subset X$. In this case, query $Q = \mathcal{A}gg^*_{T'}(\mathrm{G(F}(M)|\mathrm{Z})$ is considered to be correct by [20]. In the above definition, Condition 5 must hold for any grouping set $\mathrm{X}' \subset \mathrm{X}$ in the query defining T', but Z is restricted to B. Therefore, the condition to determine if query Q is correct, is to enforce that *every attribute* B *of any dimension* D *in* Z *is summarizable from* X_D, *where* X_D *is the set of dimension attributes of* D *in* X (Equation (5) must hold for all attributes B of any dimension D and any subset of attributes $\mathrm{X}' \subset \mathrm{X}_D$).

[20] show that the summarizability Condition 5 can also be expressed independently of the fact tables which refer to a given dimension: attribute B is summarizable from X in dimension D if and only if for every bottom level attribute $\mathrm{A}_\perp$ of D, the following equality holds, where π denotes the relational duplicate elimination projection and $\bowtie$ denotes the

null-eliminating join:

$$\pi_{A_\perp,B}(D) = \bigcup_{A_i \in X} (\pi_{A_\perp,A_i}(D) \bowtie \pi_{A_i,B}(D)) \tag{6}$$

Example 27. Consider the product dimension *PROD_NEW* in Table 35. Using Equation (6), we can show that attribute CATEGORY is summarizable from X = {SUBCATEGORY} because $\pi_{\text{PROD\_SKU,CATEGORY}}(PROD\_NEW)$ is equal to the join $\pi_{\text{PROD\_SKU,SUBCATEGORY}}(PROD\_NEW) \bowtie \pi_{\text{SUBCATEGORY,CATEGORY}}(PROD\_NEW)$. However, attribute CATEGORY is not summarizable from X = {VIDEO_RES} because the natural join between $\pi_{\text{PROD\_SKU,VIDEO\_RES}}(PROD\_NEW)$ and $\pi_{\text{VIDEO\_RES,CATEGORY}}(PROD\_NEW)$ eliminates products 'p_03' and 'p_04'.

Table 35. *PROD_NEW*

PROD_SKU	SUBCATEGORY	CATEGORY	VIDEO_RES	BRAND
p_01	Video projector	Video	1920x1080	Epson
p_02	TV	Video	3840x2160	Philips
p_05	TV	Video	3840x2160	Samsung
p_03	Radio	Audio	-	Philips
p_04	CD-player	Audio	-	Samsung

To efficiently check summarizability, [20] proposes to transform the summarizability problem into the problem of verifying the satisfaction of a set of dimension constraints by some dimension D. Let A_i be a dimension attribute of a dimension D and $\langle \text{A}_i, \text{A}_{i+1}, ..., \text{A}_{i+n} \rangle$ denote a path in D such that $\text{A}_k \preccurlyeq \text{A}_{k+1}$, $i \leq k < i+n$. Then, the following dimension constraints can be defined on A_k and D:

(1) $D \models \langle \text{A}_i, \text{A}_{i+1}, ..., \text{A}_j \rangle$ means that for every attribute value v of A_i, there exists a tuple t in D such that $t.\text{A}_i = v$ and $t.\text{A}_{i+1}$, ..., $t.\text{A}_j$ are non-null (all values of A_i roll-up to a value A_j through a value of A_{i+1} ...). We shall say that A_i *rolls up to* A_j.

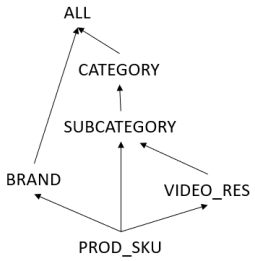
(2) $D \models \langle \text{A}_i, ..., \text{A}_j = k \rangle$ means that for every attribute value v of A_i, there exists a tuple t in D such that $t.\text{A}_i = v$ and $t.\text{A}_j$ is not null if and only if $t.\text{A}_j = k$.

Constraints can then be composed using the usual Boolean logical connectives. Now, assume that a set of constraints are specified on the schema of D. To determine if an attribute B is summarizable from X = $\{A_1, ..., A_n\}$ in D, one must determine if, for each bottom level attribute $\text{A}_\perp$ of D, the following constraint is satisfied :

$$D \models \langle A_\perp, ..., B \rangle \implies (\langle A_\perp, ..., A_1, ..., B \rangle \oplus ... \oplus (\langle A_\perp, ..., A_n, ..., B \rangle) \tag{7}$$

where $\oplus$ denotes an exclusive disjunction (XOR). Intuitively, if all values of $\text{A}_\perp$ roll-up to a value of B then all these values either roll-up through values of A_1 or (exclusive) through values of A_2 ... or through values of A_n. We use the following examples to illustrate the use of constraints to determine summarizability.

Example 28. Suppose that in dimension *PROD_NEW*, VIDEO_RES has now SUBCATEGORY as parent in the hierarchy type (Figure 13a). Then the two constraints (a) and (b) shown in Figure 13b are expressed on *PROD*. Note that the disjunction constraint (a) is more expressive than the context dependency of [27, 28] because it is not restricted to a single value.



(a) Hierarchy type of *PROD_NEW*

Rules	
(a)	⟨PROD_SKU, VIDEO_RES, SUBCATEGORY = 'TV' ⟩ ⊕ ⟨PROD_SKU, VIDEO_RES, SUBCATEGORY = 'Video projector' ⟩
(b)	⟨A_i, A_j⟩, for all other edges (A_i, A_j)
Semantics	
(a)	A value of PROD_SKU rolls up to VIDEO_RES and SUBCATEGORY only for the 'Video Projector' and 'TV' values of SUBCATEGORY
(b)	All other attributes directly roll up to their parent attribute

(b) Constraints on *PROD_NEW*

Fig. 13. The dimension schema of *PROD_NEW*

The attribute CATEGORY is summarizable from X = {SUBCATEGORY} because the following constraint is satisfied (all products roll up to a category through some subcategory):

$$PROD \models \langle \text{PROD\_SKU}, \ldots, \text{CATEGORY} \rangle \Rightarrow \langle \text{PROD\_SKU}, \text{SUBCATEGORY}, \text{CATEGORY} \rangle \quad (8)$$

Using the constraints in (b), we can compose ⟨PROD_SKU, SUBCATEGORY⟩ and ⟨SUBCATEGORY, CATEGORY⟩ to yield the final constraint. Thus, if the table PROD_NEW_SALES is first aggregated with a grouping set {SUBCATEGORY CATEGORY}, then a query that further aggregates this result with a grouping set {CATEGORY} is correct.

However, attribute SUBCATEGORY is not summarizable from X = {VIDEO_RES} because the following constraint cannot be satisfied:

$$PROD \models \langle \text{PROD\_SKU}, \text{SUBCATEGORY} \rangle \Rightarrow \langle \text{PROD\_SKU}, \text{VIDEO\_RES}, \text{SUBCATEGORY} \rangle \quad (9)$$

Thus, if table PROD_NEW_SALES is first aggregated with a grouping set {SUBCATEGORY, VIDEO_RES, CATEGORY}, then a query Q that further aggregates this result with a grouping set {SUBCATEGORY, CATEGORY} is incorrect. If the tuples of PROD_NEW_SALES with SUBCATEGORY attribute values 'TV' or 'Video projector' are filtered out then the summarizability constraint can be satisfied and previous query Q will be correct.

It is clear that the data model and constraints proposed by [20] subsume the data model with context dependencies of [27, 28]. We already showed that our summarizability conditions are more expressive than context dependencies by considering *null* values as regular values for aggregation. The same arguments apply to dimension constraints. In addition, [20] has the following limitations with respect to our work. First, any non-null value of a mandatory attribute must map to a single parent value in the hierarchy. This discards the use of dimension tables like *SALESORG* in Table 8b. Second, measure attributes cannot depend on a subset of the dimensions of a fact table. This discards the use of fact tables resulting from interactive user queries like the result of the left-merge of T2 with DEM′in Table 6b, presented in the Introduction section. Finally, unlike other works, the notion of applicability of an aggregation function to an attribute is not covered.

6 CONCLUSIONS

In this article, we introduce a new framework for controlling the correctness of aggregation operations during sessions of interactive analytic queries. Our framework adopts an *attribute-centric view*, whereby *aggregable properties* of attributes

are used to describe and control the interaction between measures, dimensions and aggregation functions. As a first advantage, aggregable properties enable the designers of analytic tables to describe the wide variety of semantic properties of measure attributes with respect to their dimensions defined in previous work [19, 26, 30, 36, 40, 50]. Another advantage of aggregable properties is their ability to guarantee that aggregate queries over some attributes can only be expressed if these attributes are summarizable. We provide two definitions of summarizable attributes. Our first definition covers the case when an aggregate query is defined over the result of another aggregate query; it subsumes the definitions of previous work on summarizability [20, 27, 28, 30, 40]. Our second definition introduces the new notion of *G-summarizability* which applies in the case of an aggregate query defined over the result of an arbitrary analytic queries. The two definitions are complementary. Our main technical results are the definition of *propagation* rules that automatically compute the aggregable properties of attributes in the result of an analytic query knowing the aggregable properties of the attributes in the operand tables of the query. We progressively refine our propagation rules to handle the semantic properties of measures, and the summarizability and G-summarizability properties of attributes.

There is a number of perspectives for future work. First, aggregable properties could be extended to handle other correctness issues of aggregate queries. Currently, we rely on literal functional dependencies (LFD) for analyzing the summarizability properties of query results. Simpson's paradox [55] is an example of incorrect causal interpretation of aggregated attribute values where a statistical observation like ratio or bias on the measures from several partitions might disappear or be inverted on the aggregated measures over these partitions. Aggregable properties could be extended to guard against this kind of statistical errors by exploiting existing causal dependencies between table attributes [38] (representing features) in addition to LFD.

Another direction of research is the "explainability" of aggregated values in an analytic table resulting from an interactive data analysis session. Aggregable properties provide explanations for the decision to forbid an incorrect aggregate query on a table. They also help the user to backtrack in her session to find an intermediate result over which a desired aggregation can be expressed. However, as shown in Example 23, a priori non-summarizable aggregate queries could be accepted provided that adequate annotations are added to the result table so that aggregated values can be properly interpreted. Generating such minimal annotations and propagating them is still open.

Finally, although a large part of our data model has already been prototyped in SAP HANA [44], a significant effort is needed to integrate our framework into self-service data preparation tools and analytic database systems.

Another possible direction is the implementation of our framework as an independent software component, *e.g.* Python library, which could be part of data preparation pipelines for Deep Learning applications (Python notebooks).

REFERENCES

[1] Julien Aligon, Enrico Gallinucci, Matteo Golfarelli, Patrick Marcel, and Stefano Rizzi. 2015. A collaborative filtering approach for recommending OLAP sessions. *Decision Support Systems* 69 (2015), 20–30.

[2] Paolo Atzeni and Nicola M. Morfuni. 1984. Functional Dependencies in Relations with Null Values. *Inf. Process. Lett.* 18, 4 (1984), 233–238. https://doi.org/10.1016/0020-0190(84)90117-0

[3] Antonio Badia and Daniel Lemire. 2014. Functional dependencies with null markers. *Comput. J.* 58, 5 (2014), 1160–1168.

[4] Jens Bleiholder and Felix Naumann. 2009. Data Fusion. *ACM Comput. Surv.* 41, 1, Article 1 (Jan. 2009), 41 pages. https://doi.org/10.1145/1456650.1456651

[5] Cristiana Bolchini, Elisa Quintarelli, and Letizia Tanca. 2012. Context Support for Designing Analytical Queries. In *Methodologies and Technologies for Networked Enterprises - ArtDeco: Adaptive Infrastructures for Decentralised Organisations*. Springer, 277–289.

[6] Robert Brunel, Jan Finis, Gerald Franz, Norman May, Alfons Kemper, Thomas Neumann, and Franz Faerber. 2015. Supporting hierarchical data in SAP HANA. In *Data Engineering (ICDE), 2015 IEEE 31st International Conference on*. IEEE, 1280–1291.

[7] Robert Brunel, Norman May, and Alfons Kemper. 2016. Index-assisted hierarchical computations in main-memory RDBMS. *Proceedings of the VLDB Endowment* 9, 12 (2016), 1065–1076. https://doi.org/10.14778/2994509.2994524

[8] Mónica Caniupán, Loreto Bravo, and Carlos A. Hurtado. 2012. Repairing inconsistent dimensions in data warehouses. *Data & Knowledge Engineering* 79-80 (2012), 17–39. https://doi.org/10.1016/j.datak.2012.04.002

[9] Gloria Chatzopoulou, Magdalini Eirinaki, and Neoklis Polyzotis. 2009. Query recommendations for interactive database exploration. In *International Conference on Scientific and Statistical Database Management (SSDBM)*. Springer, 3–18.

[10] Marco Console, Paolo Guagliardo, and Leonid Libkin. 2020. Fragments of bag relational algebra: Expressiveness and certain answers. *Information Systems* (2020), 101604.

[11] datadog [n.d.]. Datadog: Cloud Monitoring as a Web Service Software. https://www.datadoghq.com/.

[12] Marina Drosou and Evaggelia Pitoura. 2013. YmalDB: exploring relational databases via result-driven recommendations. *VLDB J.* 22, 6 (2013), 849–874. https://doi.org/10.1007/s00778-013-0311-4

[13] Magdalini Eirinaki, Suju Abraham, Neoklis Polyzotis, and Naushin Shaikh. 2014. Querie: Collaborative database exploration. *IEEE Transactions on Knowledge and Data Engineering (TKDE)* 26, 7 (2014), 1778–1790.

[14] Hector Garcia-Molina, Jeffrey D. Ullman, and Jennifer Widom. 2008. *Database Systems: The Complete Book* (2 ed.). Prentice Hall Press, USA.

[15] J. Gray, A. Bosworth, A. Lyaman, and H. Pirahesh. 1996. Data cube: a relational aggregation operator generalizing GROUP-BY, CROSS-TAB, and SUB-TOTALS. In *Proceedings of the Twelfth International Conference on Data Engineering* (New Orleans, LA, USA). IEEE Comput. Soc. Press, 152–159. https://doi.org/10.1109/ICDE.1996.492099

[16] Paolo Guagliardo and Leonid Libkin. 2017. A formal semantics of SQL queries, its validation, and applications. *Proceedings of the VLDB Endowment* 11, 1 (Sept. 2017), 27–39. https://doi.org/10.14778/3151113.3151116

[17] F Binti Hamzah, C Lau, Hafeez Nazri, Dominic Vincent Ligot, Guanhua Lee, Cheng Liang Tan, MKBM Shaib, Ummi Hasanah Binti Zaidon, Adina Binti Abdullah, Ming Hong Chung, et al. 2020. CoronaTracker: worldwide COVID-19 outbreak data analysis and prediction. *Bull World Health Organ* 1, 32 (2020).

[18] John Horner and Il-Yeol Song. 2005. A Taxonomy of Inaccurate Summaries and Their Management in OLAP Systems. In *International Conference on Conceptual Modeling*. Vol. 3716. Springer, 433–448.

[19] John Horner, Il-Yeol Song, and Peter P. Chen. 2004. An analysis of additivity in OLAP systems. In *Proceedings of the 7th ACM international workshop on Data warehousing and OLAP*. ACM, 83–91.

[20] Carlos A. Hurtado, Claudio Gutierrez, and Alberto O. Mendelzon. 2005. Capturing summarizability with integrity constraints in OLAP. *ACM Transactions on Database Systems* 30, 3 (2005), 854–886.

[21] Christian S Jensen, Torben Bach Pedersen, and Christian Thomsen. 2010. Multidimensional databases and data warehousing. *Synthesis Lectures on Data Management* 2, 1 (2010), 1–111.

[22] Paulo Jesus, Carlos Baquero, and Paulo Sérgio Almeida. 2011. A Survey of Distributed Data Aggregation Algorithms. *CoRR* abs/1110.0725 (2011). arXiv:1110.0725 http://arxiv.org/abs/1110.0725

[23] Manas Joglekar, Hector Garcia-Molina, and Aditya G. Parameswaran. 2019. Interactive Data Exploration with Smart Drill-Down. *IEEE Transactions on Knowledge and Data Engineering (TKDE)* 31, 1 (2019), 46–60.

[24] Nodira Khoussainova, YongChul Kwon, Magdalena Balazinska, and Dan Suciu. 2010. SnipSuggest: Context-Aware Autocompletion for SQL. *Proc. VLDB Endow.* 4, 1 (2010), 22–33. https://doi.org/10.14778/1880172.1880175

[25] kibana [n.d.]. Kibana: Your window into the Elastic Stack Software. https://www.elastic.co/kibana.

[26] Ralph Kimball and Margy Ross. 2013. *The Data Warehouse Toolkit*. John Wiley & Sons.

[27] Jens Lechtenbörger and Gottfried Vossen. 2003. Multidimensional normal forms for data warehouse design. *Information Systems* 28, 5 (2003), 415–434.

[28] Wolfgang Lehner, Jens Albrecht, and Hartmut Wedekind. 1998. Normal forms for multidimensional databases. In *Scientific and Statistical Database Management, 1998. Proceedings. Tenth International Conference on*. IEEE, 63–72.

[29] Hans-Joachim Lenz and Bernhard Thalheim. 2009. A Formal Framework of Aggregation for the OLAP-OLTP Model. *J. UCS* 15, 1 (2009), 273–303. https://doi.org/10.3217/jucs-015-01-0273

[30] H.-J. Lenz and Arie Shoshani. 1997. Summarizability in OLAP and statistical data bases. In *Proceedings of the Ninth International Conference on Scientific and Statistical Database Management (SSDBM '97)*. 132–143.

[31] H.-J. Lenz and Bernhard Thalheim. 2001. OLAP databases and aggregation functions. In *Scientific and Statistical Database Management, 2001. SSDBM 2001. Proceedings. Thirteenth International Conference on*. IEEE, 91–100.

[32] Rutian Liu, Eric Simon, Bernd Amann, and Stéphane Gançarski. 2020. Discovering and merging related analytic datasets. *Information Systems* 91 (2020), 101495.

[33] Patrick Marcel and Elsa Negre. 2011. A survey of query recommendation techniques for data warehouse exploration.. In *Journées Francophones sur les Entrepôts de Données et l'Analyse en ligne (EDA)*. 119–134.

[34] Jose-Norberto Mazón, Jens Lechtenbörger, and Juan Trujillo. 2009. A survey on summarizability issues in multidimensional modeling. *Data & Knowledge Engineering* 68, 12 (2009), 1452–1469.

[35] Microsoft. [n.d.]. Azure Blob storage: Massively scalable and secure object storage for cloud-native workloads, archives, data lakes, high-performance computing, and machine learning Software. https://azure.microsoft.com/en-us/services/storage/blobs/.

[36] Tapio Niemi, Marko Niinimäki, Peter Thanisch, and Jyrki Nummenmaa. 2014. Detecting summarizability in OLAP. *Data & Knowledge Engineering* 89 (2014), 1–20.

[37] paxata [n.d.]. Paxata | Self-Service Data Preparation for Data Analytics. https://www.paxata.com/.
[38] Judea Pearl, Madelyn Gleamour, and Nicholas Jewell. 2016. *Causal Inferences in Statistics*. Wiley.
[39] Torben Bach Pedersen, Christian S. Jensen, and Curtis E. Dyreson. 1999. Extending Practical Pre-Aggregation in On-Line Analytical Processing. In *Proceedings of the 25th International Conference on Very Large Data Bases (VLDB '99)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 663–674.
[40] Torben Bach Pedersen, Christian S. Jensen, and Curtis E. Dyreson. 2001. A foundation for capturing and querying complex multidimensional data. *Information Systems* 26, 5 (2001), 383–423. https://doi.org/10.1016/S0306-4379(01)00023-0
[41] powerbi [n.d.]. Power BI | Interactive Data Visualization BI Tools. https://powerbi.microsoft.com/en-us/.
[42] qlik [n.d.]. Data Analytics for Modern Business Intelligence | Qlik. https://www.qlik.com/us.
[43] Maurizio Rafanelli and Arie Shoshani. 1990. Storm: A statistical object representation model. In *International Conference on Scientific and Statistical Database Management*. Vol. 420. Springer, 14–29. https://doi.org/10.1007/3-540-52342-1_18
[44] Liu Rutian. 2020. *Semantic Services for Assisting Users to Augment Data in the Context of Analytic Data Sources*. PhD Thesis. Sorbonne Université.
[45] s3 [n.d.]. Amazon S3 Object storage built to store and retrieve any amount of data from anywhere Software. https://aws.amazon.com/s3.
[46] sapdata [n.d.]. SAP Agile Data Preparation and Transformation Solution. https://www.sap.com/products/data-preparation.html.
[47] Sunita Sarawagi, Rakesh Agrawal, and Nimrod Megiddo. 1998. Discovery-driven exploration of OLAP data cubes. In *International Conference on Extending Database Technology (EDBT)*. 168–182.
[48] Manish Singh, Michael J Cafarella, and HV Jagadish. 2016. DBExplorer: Exploratory Search in Databases.. In *International Conference on Extending Database Technology (EDBT)*. 89–100.
[49] splunk [n.d.]. Splunk: The Data-to-Everything™ Platform Software. https://www.splunk.com/.
[50] Stanley Smith Stevens. 1946. On the theory of scales of measurement. *Science* 103, 2684 (1946), 677–680.
[51] tableau [n.d.]. Tableau: Business Intelligence and Analytics Software. https://www.tableau.com/.
[52] Peter Thanisch, Tapio Niemi, Jyrki Nummenmaa, and Marko Niinimäki. 2019. Detecting measurement issues in SQL arithmetic expressions and aggregations. *Data & Knowledge Engineering* 122 (2019), 116–129. https://doi.org/10.1016/j.datak.2019.06.001
[53] Trifacta. [n.d.]. Data Wrangling Tools & Software | Trifacta. https://www.trifacta.com/.
[54] SQL tutorial. [n.d.]. The SQL Rollup operator. https://www.sqltutorial.org/sql-rollup/.
[55] Clifford H Wagner. 1982. Simpson's paradox in real life. *The American Statistician* 36, 1 (1982), 46–48.
[56] Wikipedia. [n.d.]. Olap Cube. https://en.wikipedia.org/wiki/OLAP_cube.
[57] Alice Zheng and Amanda Casari. 2018. Feature Engineering for Machine Learning. (2018), 217.