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 Business Intelligence (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 that 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 that 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, multidimensional data model, interactive query sessions

ACM Reference format:
Eric Simon, Bernd Amann, Rutian Liu, and Stéphane Gançarski. 2023. Controlling the Correctness of Aggregation Operations During Sessions of Interactive Analytic Queries. ACM J. Data Inform. Quality 15, 2, Article 12 (June 2023), 41 pages.
https://doi.org/10.1145/3575812
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 [47], Elasticsearch/Kibana [25], and Datadog [13]), trend analysis in finance or social networks, or the conduct of epidemiological studies in healthcare [19]. In a world where an overwhelming amount of raw data are collected and stored at an affordable price in cloud object stores (e.g., Amazon S3 [44] and Azure Blob Storage [4]), properly aggregated and cleaned data are the data foundation layer on which augmented analytics are built with the help of machine learning pipelines.

Traditionally, the creation of analytic datasets has been governed by a central Information Technology (IT) authority. Nowadays, the emergence of self-service data preparation and Business Intelligence (BI) tools (e.g., References [36, 41, 42, 45, 50, 54]) empowers business users and data scientists to directly create their own analytic datasets. With these tools, data analysis becomes an interactive and iterative process whereby a user can select one or more analytic datasets, issue a data analysis action (translated into a query), obtain a new analytic dataset, and decide which action to perform next. Users actions include operations such as filtering, data transformation, match up, and aggregation. Eventually, a user may decide to share with other users the final analytic dataset thus obtained in the form of a reusable view or a materialized dataset.

In this new environment, it is important to stress that the analytic datasets created by business users do not necessarily follow the standard modeling rules adopted for the design of data warehouses or data marts. The core problem addressed by our research is that data experts, who directly manipulate such 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 [23]. The dimension table REGION (Table 1(b)) describes a list of cities. These cities are referenced by the fact table DEMographics, which contains three dimension attributes, CITY, STATE, and COUNTRY, from dimension REGION and one attribute YEAR from another dimension table TIME (not shown). Attributes POP and UNEMP are measure attributes that respectively represent the population and the unemployment rate in that city. Note that dimension table REGION does not have any surrogate primary key (the dimension identifier is formed by CITY, STATE, COUNTRY), and its hierarchy is non-strict (e.g., a CITY value can have multiple STATE values) and non-covering (e.g., for some cities, STATE is a non-applicable value) [23]. While contrasting with some data warehouse modeling practices, such dimensions capture real-life situations and are considered by previous work on multidimensional databases [3, 22, 27].

Interactive session 1: Suppose that a user selects the DEM dataset and wants to count (without duplicates) the number of cities (CITY) per state and country. This action is translated into an Structures Query Language (SQL) aggregate query on table DEM by doing a COUNT DISTINCT(CITY) group by STATE and COUNTRY whose result table T1 is displayed in Table 2(a) (the count has been renamed into NB_CITIES). Next, suppose that the business user aggregates further NB_CITIES by COUNTRY using function SUM, yielding a new table T2 displayed in Table 2(b). The values of SUM(NB_CITIES) in T2 are, however, hard to interpret: For country “USA,” if we refer to the original table DEM, then “5” is neither the number of cities by country with duplicates (which is “7”) nor the number of cities without duplicates (which is “4”). Thus, if the user expects to obtain the number of cities by country without duplicates, then the result in T2 is incorrect. Now, suppose that
Table 1. Fact and Dimension Tables for Demographics

(a) Fact table DEM (Demographics)

| CITY      | STATE | COUNTRY | YEAR | POP  | UNEMP (%) |
|-----------|-------|---------|------|------|-----------|
| Dublin    | CA    | USA     | 2017 | 61   | 3.1       |
| Palo Alto | CA    | USA     | 2017 | 67   | 2.1       |
| Dublin    | CA    | USA     | 2018 | 63   | 3.0       |
| Palo Alto | CA    | USA     | 2018 | 66   | 2.0       |
| San Jose  | CA    | USA     | 2018 | 1,028| 2.2       |
| Dublin    | OH    | USA     | 2018 | 44   | 3.7       |
| Washington D.C | — | USA | 2018 | 672  | 6.2       |
| Dublin    | —     | Ireland | 2018 | 1,348 | 6.71     |
| Dublin    | —     | Belarus | 2018 | 0.354 |          |

(b) Dimension table REGION

| CITY      | STATE | COUNTRY | REGION     |
|-----------|-------|---------|------------|
| Dublin    | CA    | USA     | North America |
| Palo Alto | CA    | USA     | North America |
| San Jose  | CA    | USA     | North America |
| Dublin    | OH    | USA     | North America |
| Washington D.C | — | USA | North America |
| Dublin    | —     | Ireland | Europe     |
| Dublin    | —     | Belarus | Europe     |

Table 2. Aggregate Query Results

(a) Table $T_1$

| NB_CITIES | STATE | COUNTRY |
|-----------|-------|---------|
| 1          | OH    | USA     |
| 3          | CA    | USA     |
| 1          | —     | USA     |
| 1          | —     | Ireland |
| 1          | —     | Belarus |

(b) Table $T_2$

| SUM(NB_CITIES) | COUNTRY |
|----------------|---------|
| 5              | USA     |
| 1              | Ireland |
| 1              | Belarus |

(c) Table $T'_2$

| SUM(NB_CITIES) | COUNTRY |
|----------------|---------|
| 7              | USA     |
| 1              | Ireland |
| 1              | Belarus |

Fig. 1. Interactive data analysis session 2 yielding SALES_DEM_USA.

the first query in the interactive session is counting cities with duplicates (doing a COUNT(CITY)), and the second query was summing NB_CITIES as before. The result table $T'_2$ displayed in Table 2(c) would be correct this time, because it is equivalent to the count with duplicates of cities by country directly obtained from DEM. Detecting incorrect steps in a sequence of aggregation queries is not trivial for non-expert business users (and even for experts), and, in this article, we propose to compute and maintain additional metadata on $T_1$ for detecting incorrect aggregations like doing a SUM(NB_CITIES) by country as in $T_2$.

Interactive session 2: Another interactive session using the datasets in Table 3 is shown in Figure 1. Suppose that a user first selects 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.
Table 3. Fact and Dimension Tables for Store Sales

(a) Dimension table SALESORG

| STORE_ID | CITY    | STATE    | COUNTRY |
|----------|---------|----------|---------|
| Ca_01    | Dublin  | California | USA    |
| Ca_02    | Dublin  | California | USA    |
| Sa_01    | San Jose| California | USA    |
| Oh_01    | Dublin  | Ohio      | USA    |
| Wa_01    | Washington DC | — | USA |
| Wa_02    | Washington DC | — | USA |
| Du_01    | Dublin  | —         | Ireland|

(b) Fact table STORE_SALES

| STORE_ID | CITY    | STATE    | COUNTRY | YEAR | AMOUNT |
|----------|---------|----------|---------|------|--------|
| Ca_01    | Dublin  | California | USA    | 2018 | 5.3    |
| Ca_02    | Dublin  | California | USA    | 2018 | 1.4    |
| Ca_01    | Dublin  | California | USA    | 2017 | 3.5    |
| Sa_01    | San Jose| California | USA    | 2018 | 22.8   |
| Oh_o1    | Dublin  | Ohio      | USA    | 2018 | 1.2    |
| Wa_o1    | Washington DC | — | USA    | 2018 | 16.1   |
| Wa_o2    | Washington DC | — | USA    | 2018 | 27.6   |
| Du_o1    | Dublin  | —         | Ireland| 2018 | 7.8    |

Table 4. Result T4 in Session 2 of Figure 1

| 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        |

Next, the user filters STORE_SALES on COUNTRY = ’USA’ and YEAR = ‘2018’, yielding a table named T3 (not displayed) and then aggregates SUM(AMOUNT) for each partition of CITY, STATE, COUNTRY, and YEAR, yielding a table named T4 displayed in Table 4. We consider 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.

Next, the user augments the schema of T4 with the measure attribute POP of table DEM, yielding a new table named T5 (Table 5(a)). This latter action, called a left-merge or a drill across [26, 53] query, is translated into an SQL left outer join query between T4 and DEM on attributes CITY, STATE, COUNTRY, and YEAR. For each attribute, the join predicate tests if the two values are equal or both values are null. In table T5, the dimension attributes CITY, STATE, and COUNTRY are defined in the intersection of both dimensions REGION and SALESORG on their common attributes. Note that each row of T5 has correct values for both POP and SUM(AMOUNT) if we compare to the values that would be directly obtained from DEM and STORE_SALES, respectively.

Finally, the user sums attributes SUM(AMOUNT) and POP of T5 by STATE, COUNTRY, and YEAR, yielding the final result SALES_DEM_USA displayed in Table 5(b). There, the value of SUM(POP) can be misleading, because it does not represent 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 in the “California” state. As before, our proposition is to compute proper metadata on T5 to detect that the aggregation along CITY of SUM(POP) should not be allowed.
Table 5. Results T5 and SALES_DEM_USA in Session of Figure 1

(a) Result T5 in session of Figure 1

| CITY        | STATE | COUNTRY | YEAR | SUM(AMOUNT) | POP  |
|-------------|-------|---------|------|-------------|------|
| Dublin      | CA    | USA     | 2018 | 6.7         | 61   |
| San Jose    | CA    | USA     | 2018 | 22.8        | 1,028|
| Dublin      | OH    | 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) |
|-------|---------|------|-------------|----------|
| CA    | USA     | 2018 | 29.5        | 1,089    |
| OH    | USA     | 2018 | 1.2         | 44       |
| —     | USA     | 2018 | 43.7        | 672      |

Fig. 2. Final flow of interactive queries yielding a correct instance of SALES_DEM_USA.

**Variant of interactive session 2:** To obtain the correct population of each state, suppose that the user moves one step back in the previous interactive session (before the left-merge operation), and sums POP by STATE, COUNTRY, and YEAR, in table DEM yielding a new table DEM’ (in Table 6(a)). Note that aggregations of measure POP in DEM must only be done within partitions defined by YEAR and possibly other attributes to avoid the double counting of the population of “Palo Alto” and “Dublin” in California. Here again, our proposition is to extend the definition of table DEM with suitable metadata to control which aggregation operation can be done on POP and along which dimension attributes. Next, the user performs a left-merge operation of DEM’ with T4 yielding the result T5’ displayed in Table 6(b) and sums attributes SUM(POP) and SUM(AMOUNT) by STATE, COUNTRY, and YEAR. This summation again leads to an incorrect result, since the population of California would be double counted. The proper explanation is that tuples from DEM’ match multiple tuples of T4. Our proposition is to compute metadata on the result of the left-merge of T4 and DEM’ to detect that the summation operation should be rejected.

Finally, to overcome this problem the user backtracks again in the interactive session to table T3 and sums AMOUNT by STATE, COUNTRY, and YEAR, yielding a new table T4’. Here, each tuple in T4’ is considered to be correct, because it would also be a tuple in the result of the same aggregate query computed over STORE_SALES. After merging T4’ with DEM’, the resulting table SALES_DEM_USA as displayed in Table 7 is finally correct. The interactive session that produced this result is displayed in Figure 2.

1.2 Related Work

The occurrence of an incorrect result after a sequence of two aggregations, with respect to the correctness criteria explained in the example of Interactive Session 1, is known as a summarizability problem. In the original definition of the problem by Lenz and Shoshani [29], an initial fact table represents micro-data, and a summarization query over that table yields another fact table representing macro-data by performing an aggregation operation F(A) for each partition of the table...
defined by a grouping set of attributes $X$. In Interactive Session 1, table DEM represents the micro-data and table T1 in Table 2(a) represents the macro-data obtained after summarizing attribute CITY using function COUNT_DISTINCT for each partition defined by the grouping set \{STATE, COUNTRY\}. In essence, the problem of summarizability is to determine whether, for some new 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, then the summarization query over $F(A)$ of the macro-data is said to be correct. In our example, the query that summarizes NB_CITIES in T1 using function SUM is incorrect: There is no summarization query over CITY in DEM using function COUNT_DISTINCT that returns the same result.

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 some (or all) aggregation queries over a previously aggregated fact table are guaranteed to be correct (see Reference [33] and Reference [9] for a survey of these solutions). For instance, some works propose to repair the hierarchy levels [3, 22] or the hierarchy instances [9, 40] of dimensions, e.g., to avoid non-strict and non-covering hierarchies. However, these solutions do not satisfy our premises that not all dimensions are centrally governed and that power business users do not necessarily conform to modeling restrictions when they manipulate and create analytic datasets. A second group of constraint-based solutions [22, 27–29, 40] defines summarizability constraints over the schemas and instances of dimensions 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. For instance, in Interactive Session 1, using such constraints, we shall say that attribute CITY in T1 is not summarizable with respect to grouping set \{STATE, COUNTRY\} and function COUNT_DISTINCT using function SUM. Our research contributions belong to this second group of solutions that, instead of constraining

### Table 6. Results after First Backtracking in Session of Figure 1

(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    |
| —        | Belarus | 2018 | 0,354    |

(b) Result of T5′: 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      |
the datasets that can be used by business users in interactive sessions, check some summarizability constraints to avoid executing an aggregate query that may return incorrect results.

The detailed analysis, reported in the long version of this article [46], of the existing constraint-based solutions to the summarizability problem, reveals the following main limitations. First, within the hierarchy of a dimension, any non-null value of an attribute must map to a single parent attribute value [22, 27–29, 40]. This discards the use of non-strict [23] dimension hierarchies like SALESORG (in Table 3(a)), wherein a city can have multiple states. Second, measure attributes in a fact table must depend on all the identifiers of the dimensions over which the fact table is defined [22, 27–29]. This discards the use of fact tables such as TS’ in Table 6(b), where measure attribute SUM(Pop), which represents the population of a state, depends only on STATE, COUNTRY and YEAR. Such a table makes sense though for a business user who wants to compute a ratio between the SUM(AMOUNT) of a city and the SUM(Pop) of the corresponding state. Third, it is generally assumed that dimension hierarchies are “covering” and “balanced” [29, 40], which means that null values representing non-applicable values are forbidden in dimension tables. This discards the use of tables like SALESORG or dimensions capturing multiple classifications. Non-covering hierarchies are allowed in References [22, 27, 28], but the proposed solutions require to reason on dimension constraints whose number depends on the number of non-covering paths in the dimension hierarchies, which can be large.

In addition to the above limitations, a major weakness of all existing methods on summarizability is that they 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 our example of Interactive Session 2. In real life scenarios though, data mash-ups (supported by merge queries) are popular, because they allow the combination of analytic data that is 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). It is therefore important to control the correctness of aggregation queries that occur after any common query expressible in a self-service BI tool.

Other complementary research efforts address the problem of correctly evaluating aggregation operations. The issues raised by the semantics of SQL aggregation functions, when they are applied over an empty set of values or a set of values containing null values, are discussed in Reference [12, 18, 30, 31] where alternative semantics are proposed. In our work, we assume that the actions expressed by a business user during an interactive session are translated (whenever possible) by the BI tool into a SQL query that uses the standard SQL semantics of null values. Thus, we do not try to “fix” the semantics of SQL with nulls. Our rationale is that most BI tools adopt this translation pattern into SQL queries. Our solution is then designed to support these practical cases.

The evaluation of aggregation queries over imprecise or uncertain multidimensional data is addressed in References [8, 14, 39]. Imprecision is due to missing dimension attributes values (e.g., for low hierarchy levels) while uncertain values encode a range of possible values. In Reference [8], uncertain values only occur in measure attributes together with a belief in the likelihood of each possible value. In Reference [14], uncertain values can occur in any attribute and contain a lower and upper bound. These papers use specific criteria to define the correctness of aggregations (e.g., consistency and faithfulness in Reference [8]) that differ from the correctness criteria based on summarizability considered in this article. Hence, they propose specific algorithms to compute a correct result for aggregation operations. In our work, we do not propose any specific modeling of uncertain values. Missing values can occur in fact tables and are processed as null values using standard SQL aggregation operations.

Other works propose methods to 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 [20, 43, 51]. This happens when measure attributes hold
heterogeneous values such as euro and dollar values. We are not addressing these consistency issues that are complementary to the correctness issues targeted by this article.

1.3 Research Contributions

In this article, we present a comprehensive set of conditions to control the correctness of aggregation queries, with respect to a summarizability criteria, within an interactive data analysis session. We consider 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.

The analytic data model we introduce in Section 2 of this article is quite expressive. It accepts arbitrary dimension hierarchies that can be non-strict, non-covering, or unbalanced while fact tables have no restriction other than referring to existing values in dimension tables. The relaxation of common data modeling contraints found in OnLine Analytical Processing (OLAP) databases matches our intention to cope with the practical cases of analytic datasets manipulated by business users in self-service BI tools. 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. New fact tables are defined as the result of interactive queries over previously defined dimension and fact tables, as shown before in our examples of interactive sessions.

At the core of our approach is the definition of specific metadata, called aggregable properties, which 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. These properties generalize the notion of additivity category of a measure used in References [21, 26, 34] and extend properties that we previously introduced in Reference [32]. Default rules assist the designer of a table to define aggregable properties when the table is created as a view from source data (i.e., from non-analytic tables). Using these properties, it is possible to automatically control which aggregations are possible on an analytic table.

The central technical contribution of this article is the definition of propagation rules, which automatically compute the aggregable properties for a table resulting from an interactive analytic query, thereby allowing us to control the correctness of aggregate queries at any stage of an interactive data analysis session. More specifically, we make the following research contributions:

1. In Section 3, we introduce aggregable properties to express the semantic properties of measures, as previously defined in References [21, 26, 29, 34, 40, 49]. For instance, in our example of table DEM, the aggregable properties of measure UNEMP can express that only MIN or MAX aggregation functions can be applied, because UNEMP represents a ratio that cannot be summed or averaged along any dimension. Default rules are introduced to minimize the effort of end users for defining aggregable properties on analytic tables built from source data (i.e., non-analytic data). In the previous works cited above, users have to explicitly define which attributes of each query result are aggregable and how. Hence, our first contribution is a set of propagation rules that automatically compute aggregable properties for the results of interactive analytic queries.

2. In Section 4.1, we formally define summarizability conditions for attributes, and in Section 4.2, we present a propagation rule to compute the aggregable properties of attributes in the result of an aggregate query. These properties control the correctness (with respect to summarizability) of subsequent aggregations over these attributes. Thus, a query that is allowed (by its aggregable properties) to aggregate an attribute using some function with respect to a grouping set is guaranteed to be correct. The conditions provided by
our aggregable properties subsume the sufficient conditions defined in previous work on summarizability \[22, 27–29, 40\], because they can properly characterize the result of some sequence of aggregate queries as being correct with respect to summarizability when previous work would view this result as being wrongly incorrect.

(3) Finally, in Section 4.3, we introduce the new notion of \(G\) (generalized) 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 present propagation rules in Section 4.4 to compute aggregable properties that control that attributes can be aggregated only if they are \(G\)-summarizable. No previous work on summarizability has addressed this problem.

2 MULTIDIMENSIONAL DATA MODEL AND ANALYTIC QUERIES

In this section, we first present our multidimensional data model, composed of dimension and fact tables. We then present the types of analytic queries that can be expressed on our data model. We use conventional relational database notations \[16\]. Each table \(T\) is a finite multiset of tuples over a set of domains of values \(S = \{A_1, \ldots, A_n\}\), called attributes, where each domain may contain a null marker (a.k.a. null value). We call \(S\) the schema of \(T\).

2.1 Dimension and Fact Tables

We consider datasets in which data are 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. Initially, fact tables are built from non-analytic tables and dimension tables. Then, business users can build new fact tables using analytic queries that are presented later.

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 or more dimensions and one measure attribute.

Throughout this article, we use some examples of dimension tables identifying and describing stores (dimension \(SALESORG\), shown in Table 3(a)), products (dimension \(PROD\), shown in Figure 3), and dates (dimension \(TIME\), not shown). We also use a fact table capturing store sales (\(STORE\_SALES\) in Table 3(b)), and a fact table (\(PRODUCT\_LIST\), shown in Figure 3) describing 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 multidimensional data model that organizes a set of dimension attributes \(X\) into an attribute hierarchy noted \((X, \preceq)\). Notably, 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 \(A = (X, \preceq)\) is a set of values \(N\) and a partial order \(\preceq\), where \(N\) contains for each attribute \(X_i \in X\) a non-empty subset of values \(N_i \subseteq N\) such that each order relation \(v_i \preceq v_j\) preserves the ancestor/descendant relation \(\preceq^*\) between the corresponding attributes \(X_i\) and \(X_j\), i.e., \(v_i \in N_i, v_j \in N_j \Rightarrow X_i \preceq^* X_j\). We also assume that \((N, \preceq)\) is transitively reduced, i.e., there is no pair of values that is connected by an order relation (\(\preceq\)) and a sequence of two or more order relations.

Example 1. The attribute hierarchy for dimension \(PROD\) (displayed in Figure 3) is defined by \(PROD\_SKU \preceq BRAND \preceq COUNTRY\) and \(PROD\_SKU \preceq SUBCATEGORY \preceq CATEGORY\), in which \(PROD\_SKU\) is a
bottom level attribute and CATEGORY and COUNTRY are two top level attributes. Within the hierarchy instance of \((\text{PROD}, \preceq)\) displayed in Figure 3, we have ‘coco-can-25cl’ \(\preceq\) ‘Coco Cola’, ‘coco-can-25cl’ \(\preceq\) ‘Zora’, and ‘coco-can-25cl’ \(\preceq\) ‘Soft Drinks’.

**Definition 1 (Dimension Table).** A dimension table \(D\) over some attribute hierarchy \(\mathcal{A} = (S, \preceq)\) is a table \(D(S)\) 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.

There are practical cases of dimension data, called non-covering hierarchies [23] or structurally heterogeneous dimensions [22], wherein some hierarchy level can be optional (e.g., level STATE in dimension SALESORG). In addition, some hierarchies can be unbalanced [23], i.e., values for low hierarchy levels can be omitted. For example, in dimension table REGION (Table 1(b)), all hierarchy levels below REGION would have no values for region “Antarctica.” For these hierarchies, we use null values that represent non-applicable values in dimension attributes and treat them as regular values using the same literal equality semantics as in SQL unique constraints (see e.g., Reference [16]): Two attribute values \(t_1.A\) and \(t_2.A\) are literally equal, denoted by \(t_1.A \equiv t_2.A\), iff \(t_1.A = t_2.A\) or both values are null values We say that two tuples \(t_1\) and \(t_2\) match on a given set of attributes \(X\) iff \(t_1.X \equiv t_2.X\). No other type of null values can occur in a dimension table. That is, we require that dimension tables do not have imprecise data (missing or unknown attribute values). A dimension attribute that has null values is called optional and otherwise mandatory.

Literal equality naturally extends to sets of attributes and leads to the notion of **Literal Functional Dependencies (LFD)** [5, 56]. Let \(X\) and \(Y\) be two sets of attributes in a schema \(S\), an LFD \(X \rightarrow Y\) holds for some table \(T\) over \(S\) iff for any two tuples \(t_1, t_2\) of \(T\), when \(t_1.X = t_2.X\) then \(t_1.Y = t_2.Y\). In Reference [32], we introduced the notion of attribute graph that specifies all possible “valid” hierarchy instances of a dimension in a simple and natural way, including LFDs between any two attributes. We also provided efficient algorithms to check if a literal dependency \(U \rightarrow B\) holds for a set of dimension attributes \(U\) and a dimension attribute \(B\) and to compute the minimum set of dimension attributes (called dimension identifier) that literally determines all other attributes of the dimension. This inference mechanism plays a central role for the definition, validation, and propagation of aggregable properties.

**Example 2.** In dimension SALESORG, the lower and upper bound attributes are respectively STORE_ID and COUNTRY, and the minimal LFDs are an LFD from STORE_ID to every other
attribute and $\text{STATE} \mapsto \text{COUNTRY}$. In table $\text{PROD}$, the minimal LFDs are $\text{PROD}_{\text{SKU}} \mapsto \text{SUBCATEGORY}$, $\text{SUBCATEGORY} \mapsto \text{CATEGORY}$, and $\text{BRAND} \mapsto \text{COUNTRY}$.

**Definition 2 (Fact Table).** A fact table over a set of dimensions $D_1, \ldots, D_n$ is a table $T(S)$ 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 representing one or more measures. For each tuple $t$ of $T$ (called a fact), if $X'_i \subseteq X_i$ are all the attributes having non-null values in $t$, then there exists at least one tuple in $D_i$ that matches $t$ on attributes $X'_i$.

We make a few remarks on this definition of fact tables. First, we assume that subsets $X_i$ of attribute names are pairwise disjoint (using appropriate attribute renaming when duplicate names occur). Second, by duplicate free, we mean that there cannot be two facts that have the same values on all their dimension attributes. Third, unlike dimension tables, any attribute of a fact table can have null values with the meaning of imprecise values (i.e., missing or unknown value). Even if we assume that initially fact tables have no missing value (i.e., no null value in any mandatory dimension attribute or any measure attribute), merge operations can introduce missing values in the resulting (fact) table due to the semantics of outer-joins, as we shall see later.

The introduction of missing values has two consequences. First, according to our definition, for each fact and each dimension in a fact table, the non-null values of the dimension attributes must match at least one tuple in the corresponding dimension table. Second, we consider that an LFD $X \mapsto A$ holds in a fact table, where $X$ is a set of dimension attributes and $A$ is a measure attribute, if it holds for all the facts where $A \neq \text{null}$. To distinguish this interpretation of an LFD for measure attributes from the interpretation for dimension attributes, we write $X \mapsto^* A$ when $A$ is a measure attribute and $X \mapsto A$ when $A$ is a dimension attribute.

As a last remark, we do not address in this article the problem of modeling and processing imprecise or uncertain information in fact tables, as done in previous works like References [8, 14, 39, 40].

### 2.2 Analytic Queries

In this article, we consider analytic queries consisting of unary operations (filter, project, aggregation, pivot) and binary operations (union, difference, merge), which includes the most common data transformation operations supported by self-service data preparation and BI tools. We assume that all operations, except pivot, are translated into SQL queries, because this is the approach followed by most self-service tools. In this section, we define the semantics of these operations, which are tailored to the case of analytic tables, with an attention to their manipulation of null values.

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

In the definition, $P$ can be any well-formed Boolean predicate using negation, conjunction, and disjunction over any subset of attributes in $S$. We consider that $P$ is a Boolean function that is defined for tuples with null value attributes: Except for literal equality (and inequality), any other comparison of an attribute value with a null value evaluates to false and Boolean connectives follow the SQL logic of nulls. Thus, for a null valued attribute $\text{STATE}$, both predicates $\text{STATE} = \text{null}$ and $\text{STATE} \neq \text{‘California’}$ evaluate to true, while predicate $\text{STATE} \leq \text{‘California’}$ evaluates to false. Note that the inequality predicate $\text{STATE} \neq \text{‘California’}$ is expressed in SQL as $\text{STATE} \neq \text{‘California’} \text{ OR STATE IS NULL}$.

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) [35].
Example 3. Here are examples of filter queries on table $T(A_1, A_2, M)$: $\text{Filter}_{T}([A_1 = 'a_1'] \mid \{A_1\})$ and $\text{Filter}_{T}([A_2 \neq 'b_2'] \mid \{A_2\})$.

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

Definition 4 (Analytic Projection Query). Let $T(S)$ be an analytic table with dimension attributes $S_D \subseteq S$. Let $Y$ be a subset of $S$ such that $S_D \subseteq Y \subseteq S$. Let $f(Z) \rightarrow A$ be an optional expression where $f(Z)$ is an expression involving a set of attributes $Z \subseteq S$, 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 $Q(T) = \text{Project}_{T}(Y, f(Z) \rightarrow A)$ (respectively, $Q(T) = \text{Project}_{T}(Y)$ ) an analytic projection that returns a table $T_r$ with schema $Y \cup \{A\}$ (respectively, $Y$), such that for every tuple $t$ of $T$, there exists a unique tuple $t'$ in $T_r$ such that $t'.B \equiv t.B$ for every $B \in Y$, and $t'.A \equiv f(t.Z)$.

When the values of some attributes in $Z$ are null, the evaluation of $f(Z)$ follows the SQL semantics: If an argument of an arithmetic or string operator is null, then the result is null. Thus, calculated attributes can have null values with the meaning of unknown. An analytic projection over an analytic table $T(S)$ is a special case of an extended projection [16]. 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.

Example 4. With table $T(A_1, A_2, M, N)$, projection query $\text{Project}_{T}([A_1, A_2, M])$ simply removes measure attribute $M$, whereas projection $\text{Project}_{T}([A_1, A_2], (M + N) \rightarrow M')$ creates a new attribute $M'$ that 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 that partition analytic tables along a subset of dimension attributes and aggregate the values of a certain attribute in each partition.

Definition 5 (Analytic Aggregate Query). Let $T(S)$ be an analytic table with dimension attributes $S_D \subseteq S$, $A$ be an aggregable attribute in $S$ and $F$ be an aggregation function. We denote by $Q(T) = \text{Agg}_{T}(F(A) \mid X)$ where $X \subseteq S_D$ 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 evaluation of function $F$ over a set of values for $A$ follows the SQL semantics: Null values are eliminated, and $F$ is evaluated over the remaining values.

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

Example 5. With table $T(A_1, A_2, A_3, M, N)$, examples of aggregate queries are $\text{Agg}_{T}$(SUM($M$) $\mid \{A_2\}$) and $\text{Agg}_{T}$(COUNT($M$) $\mid \{A_1, A_3\}$).

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, 58].

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 $Q(T) = \text{Pivot}_{T}(A \mid X)$, where $X \subset S_D$, an analytic
pivot query that 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 $v_i$ (including null marker) in the domain of $T.X$. The value $t.A$ of each tuple $t \in T$ such that $t.X \equiv v_i$ is a value in the attribute $A_{\_v_i}$ of the unique tuple $t'$ in $T_r$ such that $t.(S_D - X) \equiv t'.(S_D - X)$.

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

**Example 6.** Consider the table $T(S)$ in Table 8(a). The result of pivot query $Q_7 = \text{Pivot}_T(M | A_1)$ that pivots attribute $M$ over $A_1$ is shown in Table 8(b). 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 $Q_8 = \text{Pivot}_T(M | A_3)$ is shown in Table 8(c).

Analytic merge queries combine the tuples of two analytic tables and correspond to natural outer-join operations defined in the extended relational algebra with null values. We allow the merge of two fact tables on their common dimension attributes (which have the same names in the two fact tables), and we accept that the common attributes belong to different dimensions in each fact table likewise [33]. We gave an example of such a merge operation in the Interactive Session 2 of Section 1.

**Definition 7 (Analytic Left-merge Query).** Let $Q_r = \pi_X(T)$ where $T = T \bowtie P_1 \land \ldots \land P_k \ T', T(S)$ and $T'(S')$ are two analytic tables, $\bowtie$ is a left-outer join operator, $P_i$ are equi-join predicates over a set of (common) dimension attributes $Y$, and $\pi_X$ is the duplicate elimination relational projection over a set of attributes $X$ defined below. Then $Q_r$ is a left-merge merge analytic query if the following conditions hold:

1. For each $A_i \in Y$, $\exists P_i$ such that $P_i = (T.A_i \equiv T'.A_i)$.
2. If for each subset $Z$ of $Y$ that belongs to both a dimension $D_1$ in $T$ and a dimension $D_2$ in $T'$ ($D_1 \neq D_2$) the attribute hierarchy $(Z, \preceq)$, and the closure of $Z$ with respect to LFIDs, are identical in $D_1$ and $D_2$, then $X = S \cup S'$ else $X = S \cup S'$ (denotes disjoint union, i.e., union after renaming conflicting attributes).

In the following, we will abbreviate $Q_r = \pi_X(T \bowtie P_1 \land \ldots \land P_k \ T')$ by $Q_r = T \bowtie_Y T'$, where $Y$ is the set of join attributes, call it a left-merge query, and refer to the result of $Q_r$ as a merge table.

Item 1 manages the join predicates in the merge query in the presence of nulls. Literal equality $P_i = (T.A_i \equiv T'.A_i)$ is translated into $P_i = (T.A_i = T'.A_i)$ OR $T.A_i$ is null AND $T'.A_i$ is null. The merge table preserves all rows in $T$ (with possible row duplication).

Item 2 checks that, when two different dimensions are joined on their common attributes, the structures and the properties of their respective hierarchies for the joined attributes are identical. This check is done using the attribute graphs of the two dimensions, as shown in Reference [32]. When the check fails, the merge query keeps the join attributes separately for each table.

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

(a) Input table $T$

| $Q_7$ | $A_2$ | $A_3$ | $M_{\_a_1}$ | $M_{\_a_2}$ |
|------|-------|-------|-------------|-------------|
| $b_1$ | $c_1$ | $x_1$ | $x_3$ |
| $b_2$ | $-x_2$ | $-$ | $x_4$ |

(b) $\text{Pivot}_T(M | A_1)$

| $Q_8$ | $A_1$ | $A_2$ | $M_{\_c_1}$ | $M_{\_null}$ |
|------|-------|-------|-------------|-------------|
| $a_1$ | $b_1$ | $x_1$ | $-$ |
| $a_2$ | $b_1$ | $-x_2$ | $-$ |
| $a_2$ | $b_2$ | $x_3$ | $-$ |
| $a_2$ | $b_2$ | $-x_4$ | $-$ |

(c) $\text{Pivot}_T(M | A_3)$

| $Q_{17}$ | $A_2$ | $A_3$ | $M_{\_a_1}$ | $M_{\_a_2}$ |
|---------|-------|-------|-------------|-------------|
| $b_1$ | $c_1$ | $x_1$ | $x_3$ |
| $b_2$ | $-x_2$ | $-$ | $x_4$ |
Left-merge queries are generalized to a full-outer join (▷◁) between two tables, called an analytic full merge query, or restricted to a natural join (⇒), called an analytic strict merge query. Right-merge queries \( Q(T, T') = T \bowtie_Y T' \) are equivalent to the symmetric left-merge queries \( Q(T', T) = T' \bowtie_Y T \) on the switched tables.

Analytic merge queries can support OLAP operations such as a “drill-across” between two fact tables through common “conformed” [23, 26] or “compatible” dimensions [53].

**Example 7.** Consider the fact tables \( T(A_1, A_2, A_3, M) \) and \( T'(A_2, A_3, N) \), respectively, defined over dimension \( D_1 \) (where \( A_1 \preceq A_2 \preceq A_3 \)) and dimension \( D_2 \) (where \( A_2 \preceq A_3 \)). Suppose that in both dimensions we have \( A_2 \Leftrightarrow A_3 \); then, by Item 2, they appear only once in the merge table, and the result of a “drill-across” (left merge) \( T \bowtie T' \) is a table \( T_r(A_1, A_2, A_3, M, N) \). If \( A_2 \preceq A_3 \) holds in \( D_1 \) but not in \( D_2 \), then all the dimension attributes of \( T \) and \( T' \) will be kept separately in the result of the merge. In the latter case, \( D_1 \) and \( D_2 \) are said to be incompatible in Reference [53].

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 tuples of a table are unique on their 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). Analytic difference and union are defined as follows:

- \( T - T' = \{ t | t \in T \land \nexists t' \in T' : t.Y = t'.Y \} \) keeps all tuples in \( T \) that do not match any tuple in \( T' \) on attributes \( Y \).

- \( T \cup T' = \{ t | t \in T \lor (t \in T' \land \nexists t' \in T : t.Y = t'.Y) \} \) keeps all tuples in \( T \) and all tuples in \( T' \) that do not match any tuple in \( T \) on attributes \( Y \).

Observe that union \( T \cup T' \) is a duplicate-free operation that eliminates each tuple in \( T' \) that has identical dimension values to a tuple in \( T \) and is therefore not commutative.

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 Reference [33] for a survey). Focusing on function \( \text{SUM} \), References [21, 26, 34] 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* 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.

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 \( \text{agg}_A(F, X) \) the *aggregable property* of \( A \) and state that property \( \text{agg}_A(F, X) \) holds in \( T \) if \( X \) is the maximal set of attributes along which \( A \) is aggregable using \( F \) in \( T \).

**Definition 9 (Aggregable Property).** Let \( S_D \) be the set of dimension attributes in an analytic table \( T(S) \), \( A \) be an attribute in \( S \) and \( F \) be an aggregation function.
Let \( X_f \subseteq S_D \) be the set of 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, then let \( X_d \subseteq S_D \) be a minimal subset of dimension attributes such that \( X_d \rightarrow^* A \) holds in the subset of all tuples in \( T \) where \( A \neq \text{null} \). Let \( X_d^+ \) be the set of all dimension attributes \( B \in S_D \) such that \( X_d \rightarrow B \). We call \( X_d \) a determinant of \( A \) and \( X_d^+ \) the closure of \( X_d \) in \( S_D \).

Then the aggregable property \( \text{agg}_A(F, X) \) holds in \( T \) for \( F \), where

1. Function \( F \) is applicable to \( A \) and its application is meaningful along dimension attributes that are not in \( X_f \).
2. If \( A \) is a measure attribute, then \( X = (X_d \cup X_f)^+ - X_f \).
3. If \( A \) is a dimension attribute, then \( X = S_D - \{A\} - X_f \).

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 to determine the aggregation functions that are applicable to some measure attribute, such as a statistic classification of measurements [34, 49, 51], the attribute’s aggregation behavior [38, 40], or the compatibility between the dimension types and the measure types [29]. These categorizations can be used in our context to define both the “applicability” of a function \( F \) on some attribute \( A \) and the “incompatibility” of its application with respect to set of (forbidden) dimension attributes \( X_f \).

Item 2 covers the “logical semantics” defined by the literal functional dependencies between dimension and measure attributes. Note that any measure value of attribute \( A \) is identified by \( X_d \) and therefore contained at most once in any partition defined by the partitioning attributes \( Y = (S_D - X_d) \cup X_f \). Then, we can show that this property is also true for any partition identified by the subset \( Y' \subseteq Y \) of partitioning attributes where \( Y' = (S_D - (X_d \cup X_f)^+) \cup X_f \). To show that a value determined by \( X_d \) can only appear in one sub-partition \( Y' \) of a partition \( Y \), suppose that there exists a pair of tuples \( t \) and \( t' \) where \( t.Y = t'.Y, t.X_d = t'.X_d \) and \( t.Y' \neq t'.Y' \). Then \( t.V = t'.V \) for \( V = X_d \cup X_f \) and \( t.U \neq t'.U \) for \( U = (X_d \cup X_f)^+ - X_f \). This is in contradiction with \( (X_d \cup X_f) \rightarrow (X_d \cup X_f)^+ \). Then, by taking \( X = S_D - Y' \), we obtain \( X = (X_d \cup X_f)^+ - X_f \).

In conclusion, Item 2 considers that any aggregation of a measure attribute \( A \) along any subset of \( X \) is logically correct, since any partition only contains at most one occurrence for each measure determined by \( X_d \). It is by definition meaningful, since it also respects the applicability constraint (Item 1) by excluding the attributes in \( X_f \).

Finally, Item 3 mainly states that any dimension attribute can be aggregated along all 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 applicable aggregation functions are COUNT and COUNT_DISTINCT (see also Table 9 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 8).

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 \( X_1 \) along which \( A \) can be aggregated using \( F \). It is also easy to show that if measure attribute \( A \) can be aggregated along any subset of \( X_1 \) and any subset of \( X_2 \) using \( F \), then it also can be aggregated along any subset of the union \( X_1 \cup X_2 \).

### 3.2 Default Rules for Aggregable Properties

In the cases of dimension tables built from non-analytic tables, or fact tables built from dimension tables and non-analytic tables, we provide default rules to compute the aggregable properties of
Table 9. Categories of Attributes and Their Properties

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

each attribute. These rules are based on the categorization of attributes and the properties of aggregation functions. The effort required from the designer of an analytic table 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 (e.g., using a scale-based categorization of measure attributes as in Reference [51]).

Default applicable functions: We provide a simple default categorization of attributes: NUM (numerical), DESC (descriptive/categorical), and STAT (statistical). These three 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 9 describes the six common SQL aggregation functions applicable to each category. We therefore use the attribute category of $A$ to define which aggregation function $F$ is applicable to $A$.

Default values of $X_d$ and $X_f$ for a measure attribute: If $A$ is a measure attribute of $T$, then we use the default rule that $A$ depends on all dimensions over which $T$ is defined. This actually means that, by default, $X_d$ is the fact identifier containing the identifiers of all dimensions (automatically determined using the attribute graphs of the dimensions). However, a necessary condition in the definition of an aggregable property is that $X_d$ is minimal. If the designer of the fact table considers that some dimension attribute in the default value of $X_d$ is unnecessary to determine $A$, then it is removed from $X_d$ and added to all sets of forbidden attributes $X_f$ associated with $A$.

We assume by default that the set of forbidden attributes $X_f$ is empty. If there exists a meaningless aggregation using a function $F$ along some dimensions, then this should be indicated by the designer of the fact table by adding the corresponding dimension attributes to the set $X_f$ associated with $A$ and $F$.

Default value of $X_f$ for a dimension attribute: As already mentioned, if $A$ is a dimension attribute, then 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 $agg_A(F, X)$, 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 10.

Example 8. Consider the fact table PRODUCT_LIST (PROD_SKU, COUNTRY, BRAND, YEAR, QTY) displayed in Figure 3.
Table 10. Default Values of $X_d$ and $X_f$ and Possible User Actions

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

- Attribute QTY in table PRODUCT_LIST is of category NUM, so we get from Table 9 the list of applicable aggregation functions. By default, the set of attributes $X_d$ that determines QTY is the fact identifier of PRODUCT_LIST, $X_d = \{\text{PROD_SKU}, \text{BRAND}, \text{YEAR}\}$. This set is, however, not minimal (QTY only depends on $\{\text{PROD_SKU}, \text{YEAR}\}$), so the designer of the table should remove attribute BRAND from $X_d$ and add it to $X_f$, which is by default empty. Since $(X_d \cup X_f) = \{\text{PROD_SKU}, \text{BRAND}, \text{COUNTRY}, \text{YEAR}\}$, we get the aggregable properties $agg_{\text{QTY}}(F, \{\text{PROD_SKU}, \text{COUNTRY}, \text{YEAR}\})$ for $F \in \{\text{SUM, COUNT, AVG, \ldots}\}$. In other words, QTY can only be aggregated along attributes PROD_SKU COUNTRY and YEAR.

- The dimension attribute PROD_SKU is of category DESC, which determines its applicable aggregation functions. By default, $X_f$ is empty, and we get the aggregable properties $agg_{\text{PROD_SKU}}(F, \{\text{BRAND, COUNTRY, YEAR}\})$ for $F \in \{\text{COUNT, COUNT_DISTINCT}\}$. Thus, it is correct for example, to count the number of (distinct) products per country by aggregating along BRAND and YEAR.

Next, consider the fact table STORE_SALES (Table 3(b)).

- The measure attribute AMOUNT is of category NUM, which determines its applicable aggregation functions. The default value of $X_d = \{\text{STORE_ID, YEAR}\}$ is minimal and $X_f = \emptyset$. Since STORE_ID literally determines all other dimension attributes of $\text{PROD, X = (X_d \cup X_f)^+ = \{\text{STORE_ID, CITY, STATE, COUNTRY, YEAR}\}}$ and we get the aggregable properties $agg_{\text{AMOUNT}}(F, X)$ for $F \in \{\text{SUM, COUNT, AVG, \ldots}\}$. That is, AMOUNT is aggregable along any subset of the dimension attributes of STORE_SALES.

Finally, consider the fact table DEM (Demographics) displayed in Table 1(a).

- Attribute POP is of category NUM. However, summing up POP along dimension attribute YEAR would clearly be incorrect, while it is correct along any attribute of dimension REGION. Thus, the designer of the fact table should add YEAR to $X_f$ for SUM and POP. By default, $X_d = \{\text{CITY, STATE, COUNTRY, YEAR}\}$ and it is minimal. So $X = (X_d \cup X_f)^+ - X_f = \{\text{CITY, STATE, COUNTRY}\}$ and we have the property $agg_{\text{POP}}(\text{SUM}, X)$.

- Attribute UNEMP is of category STAT, which determines its applicable functions. By default, $X_d$ is the same as for POP and $X_f$ is empty, and we have $X = (X_d \cup X_f)^+ - X_f = \{\text{CITY, STATE, COUNTRY, YEAR}\}$. Finally, suppose that the designer of the table considers that functions COUNT_DISTINCT and COUNT are meaningless for UNEMP; then the only aggregable properties are $agg_{\text{UNEMP}}(F, X)$ for $F \in \{\text{MAX, MIN}\}$.

### 3.3 Propagating Aggregable Properties

For analytic tables that result from (analytic) queries over analytic tables, we present propagation rules to obtain the aggregable properties of their attributes. In several cases, these rules do not require any user input. This is a new contribution, because in previous works on summarizability, the user must explicitly specify for every fact table which attribute is aggregable and how.

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 that 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 $A' = F(A)$ 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 11.
(4) If $A'$ is a new attribute resulting from the evaluation of an expression $f(Z) \rightarrow A'$ in $Q$, then the aggregation functions that are applicable to $A'$ are determined by the category of $A'$ (default or user-defined) using Table 9.

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 that 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 $F = \text{SUM}(A)$, the resulting attribute becomes of category STAT when $F = \text{AVG}(A)$. This change is detected using the classification in Table 11.

Next, we must determine for each attribute $A'$ in the result table the maximal subset of dimension attributes $X'$ of $T_r$ along which an aggregation using $F$ is meaningful and logically correct according to Definition 9. The propagation rules in Tables 12 and 13 define the values of $X'_d$ and $X'_f$ for the result of a unary or binary query, respectively. The value of $X'$ is then automatically determined by the formulas of Definition 9, and in the case of a measure attribute, the attribute graphs of each dimension are used to compute the closure $(X'_d \cup X'_f)^+$. When a rule specifies $X'_d$ is a determinant of $A'$, it means that a default value of $X'_d$ is computed (as per Table 10) and if it is not minimal to determine $A'$, then a user action is needed. The last column shows the required user actions: None means no action required, Minimize $X'_d$ means remove dimension attributes from the default value of $X'_d$ to obtain a determinant for measure attribute $A'$, and Complete $X'_f$ means adding attributes removed from $X'_d$ and possibly new forbidden attribute to $X'_f$.

**Example 9.** Consider fact table PRODUCT_LIST and its aggregable properties from Example 8.

- Let $T_r = \text{Filter}_{\text{PRODUCT_LIST}}((\text{YEAR } = '2017'))$. Property agg_{QTY}(\text{SUM} | X)$ holds in PRODUCT_LIST for $X = \{\text{PROD_SKU, YEAR}\}$, $X'_d = \{\text{PROD_SKU, YEAR}\}$, and $X'_f = \emptyset$. By default, $X'_d = X_d$, which is no longer minimal. So, the user action should be to remove YEAR from $X'_d$ and add it to $X'_f$, which is by default equal to $X_f$. Thus, by Table 12, agg_{QTY}(\text{SUM} | X') holds in $T_r$, with $X'_d = \{\text{PROD_SKU}\}$, $X'_f = \{\text{YEAR}\}$, and $X' = (X'_d \cup X'_f)^+ - X'_f = \{\text{PROD_SKU}\}$.
- Let $T_r = \text{Pivot}_{\text{PRODUCT_LIST}}(\text{QTY} | \text{BRAND})$. It produces two new attributes QTY_COCOCOLA and QTY_ZORA. By Table 12, since $X_d \not\subseteq Y$, the default value of $X'_d$ is $X_d - Y = \{\text{PROD_SKU, YEAR}\}$, which is minimal to determine each new measure attribute. Then $X'_f = X_f - Y = \emptyset$. Thus, both aggregable properties agg_{QTY_COCOCOLA}(\text{SUM} | X') and agg_{QTY_ZORA}(\text{SUM} | X') hold in $T_r$, where $X' = (X'_d \cup X'_f)^+ - X'_f = \{\text{PROD_SKU, YEAR}\}$.
Table 12. Propagation Rules for Unary Operations on \(T(S)\) Returning Table \(T_r(S')\)

| Unary query on \(T(S)\) | Propagation rule for inferring the aggregable properties of attribute \(A' \in S_r\) in the result \(T_r(S_r)\) | User action |
|--------------------------|-------------------------------------------------|-------------|
| \(Filter_T(P \mid Y)\) | dimension attribute \(A' \in S_D\) and \(agg_{A'}(F, X)\) holds in \(T\): \(agg_{A'}(F, X')\) holds in \(T_r\) with \(X'_f = X_f\) | None |
| Project\(_T\)(\(X, f(Z) \rightarrow m\)) | dimension attribute \(A' \in Y\) and \(agg_{A'}(F, X)\) holds in \(T\): \(agg_{A'}(F, X')\) holds in \(T_r\) with \(X'_f = X_f\). | None |
| Pivot\(_T\)(A \mid Y) | dimension attribute \(A' \in S_D - Y\) and \(agg_{A'}(F, X)\) holds in \(T\): \(agg_{A'}(F, X')\) holds in \(T_r\) with \(X'_f = X_f - Y\) | None |
| Agg\(_T\)(F(A) \mid Y) | dimension attribute \(A' \in Y\) and \(agg_{A'}(F, X)\) holds in \(T\): \(agg_{A'}(F, X')\) holds in \(T_r\) with \(X'_f = X_f\). | None |

- Let \(T_r = \text{Agg}_{\text{PRODUCT\_LIST}}(\text{SUM}(\text{QTY}) \mid Y)\) with \(Y = \{\text{BRAND}, \text{COUNTRY}, \text{YEAR}\}\). Aggregable property \(agg_{\text{SUM}(\text{QTY})}(\text{SUM}, X)\) holds in \text{PRODUCT\_LIST} with \(X = \{\text{PROD\_SKU}, \text{COUNTRY}, \text{YEAR}\}\) and \(X_f = \{\text{BRAND}\}\). By default, we have \(X_d' = \{\text{BRAND}, \text{YEAR}\}\), which is minimal and does no require any minimize user action. Thus, by Table 12, \(X'_f = X_f \cap Y = \{\text{BRAND}\}\) and \(X' = (X_d' \cup X'_f)^+ - X'_f = \{\text{COUNTRY}, \text{YEAR}\}\). Since function \text{SUM} returns a value of category \text{NUM}, we finally have \(agg_{\text{SUM}(\text{QTY})}(G \mid X')\) holds in table \(T_r\) with \(G \in \{\text{SUM}, \text{AVG}, \text{COUNT}, \text{COUNT\_DISTINCT}, \text{MIN}, \text{MAX}\}\).

- Let \(T_r = \text{Agg}_{\text{PRODUCT\_LIST}}(\text{F}(\text{PROD\_SKU}) \mid \{\text{BRAND}, \text{YEAR}\})\), with \(F \in \{\text{COUNT}, \text{COUNT\_DISTINCT}\}\). For each value of \(F\), property \(agg_{\text{F}(\text{PROD\_SKU})}(F \mid X)\) holds in \text{PRODUCT\_LIST} for \(X = \{\text{BRAND}, \text{COUNTRY}, \text{YEAR}\}\). Since \(F\) returns values of category \text{NUM}, \(G \in \{\text{SUM}, \text{AVG}, \text{COUNT}, \text{COUNT\_DISTINCT}, \text{MIN}, \text{MAX}\}\) is applicable to \text{PROD\_SKU}. Then, by the propagation rule, \(X'_f = X_f \cap Y = \emptyset\). The default value of \(X_d\) is \{\text{BRAND}, \text{YEAR}\}, which is minimal. Thus, \(agg_{\text{F}(\text{PROD\_SKU})}(G \mid X')\) holds in table \(T_r\) with \(X' = (X_d' \cup X'_f)^+ - X'_f = \{\text{BRAND}, \text{YEAR}\}\).

In each propagation rule for merge, union, and set difference summarized in Table 13, 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.

The union operator might extend \(T\) with new tuples from \(T'\) violating the condition \(X_d' \rightarrow^+ A'\) for measure attribute \(A'\). So, the propagation rule must check if \(X_d' \rightarrow^+ A'\) still holds on the result of the union (for all other operations we proved that there exists \(X_d' \subseteq X_d\) such that \(X_d' \rightarrow^+ A'\)). If not, then the user can either resolve the uniqueness violation by removing the duplicate tuples on \(X_d\) (action \text{Resolve conflicts}) or add new attributes to \(X_d\) to make it a determinant for \(A'\) (action
Table 13. Propagation Rules for Binary Operations

| Binary query on $T(S)$ and $T'(S')$ | Propagation rule for inferring the aggregable properties of attribute $A' \in S$ in the result $T_r(S_r)$ | User action |
|-------------------------------------|-------------------------------------------------|-------------|
| $T_r = T \bowtie T'$               | dimension attribute $A' \in SD$ and $\text{agg}_{A'}(F, X)$ holds in $T$: $\text{agg}_{A'}(F, X')$ holds in $T_r$ with $X'_d = X_f$. | Complete $X'_d$ |
| $T_r = T \bowtie T'$               | measure attribute $A' \in S - SD$ and $\text{agg}_{A'}(F, X)$ holds in $T$: $\text{agg}_{A'}(F, X')$ holds in $T_r$ with $X'_d = X_d$ and $X'_f = X_f$. | Complete $X'_f$ |
| $T_r = T \bowcirc T'$             | dimension attribute $A' \in SD$ and $\text{agg}_{A'}(F, X)$ holds in $T$: $\text{agg}_{A'}(F, X')$ holds in $T_r$ with $X'_d$ is determinant of $A'$ and $X'_f = X_f$. | Complete $X'_f$ |
| $T_r = T \bowcirc T'$             | measure attribute $A' \in S - SD$ and $\text{agg}_{A'}(F, X)$ holds in $T$: $\text{agg}_{A'}(F, X')$ holds in $T_r$ with $X'_d = X_d$ and $X'_f = X_f$. | Complete $X'_d$ or Resolve conflicts |
| $T_r = T \cup T'$                 | dimension attribute $A' \in SD$ and $\text{agg}_{A'}(F, X)$ holds in $T$: $\text{agg}_{A'}(F, X')$ holds in $T_r$ with $X'_d = X_d$ and $X'_f = X_f$. | None |
| $T_r = T - T'$                    | measure attrib. $A' \in S - SD$ and $\text{agg}_{A'}(F, X)$ holds in $T$: if $X_d = SD$ or $X_d$ is a determinant of $A'$ in $T_r$ then $\text{agg}_{A'}(F, X')$ holds in $T_r$ with $X'_d = X_d$ and $X'_f = X_f$. | None |
| $T_r = T - T'$                    | else $\text{agg}_{A'}(F, X')$ holds in $T_r$ with $X'_f = X_f$ and $X'_d$ is a determinant of $A'$. | Complete $X'_d$ or Resolve conflicts |

If the added attributes are in $X_f$, then they must be removed from $X'_f$, which is by default equal to $X_f$. With the former action, the user wishes to keep the original semantics of $A'$ defined in $T$ and $T'$.

For union, propagation rules involve a uniqueness test over hierarchical dimension attributes, which can be performed efficiently using specific data structures that represent fact tables in main-memory (see References [6, 7]). In the other cases, the conditions of the propagation rules are immediate to compute, because they only involve the manipulation of metadata properties.

The next proposition states that the rules of Tables 12 and 13 are correct with respect to Definition 9. Following this definition, each rule producing a new propagated aggregable property $\text{agg}_{A'}(F', X')$ is correct, if (1) $F'$ can be applied on $A'$, (2) $X'_d$ is a determinant of $A'$ (if $A'$ is a measure attribute), and (3) $X'_f$ propagates the forbidden attributes defined for $A'$ (if $A'$ exists in $T$).

**Proposition 1 (Correctness of Summarizability Propagation Rules).** The propagation rules of Tables 12 and 13 are correct.

## 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 $T$.

### 4.1 Summarizable Attributes and Distribution Functions

Figure 4 illustrates the definition of summarizable attributes when some attribute $A$ of table $T$ is aggregated with some function $F$ for each partition of attributes $Z_2$. If 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 $F$, complete $X'_d)$. If the added attributes are in $X_f$, then they must be removed from $X'_f$, which is by default equal to $X_f$. With the former action, the user wishes to keep the original semantics of $A'$ defined in $T$ and $T'$.

For union, propagation rules involve a uniqueness test over hierarchical dimension attributes, which can be performed efficiently using specific data structures that represent fact tables in main-memory (see References [6, 7]). In the other cases, the conditions of the propagation rules are immediate to compute, because they only involve the manipulation of metadata properties.

The next proposition states that the rules of Tables 12 and 13 are correct with respect to Definition 9. Following this definition, each rule producing a new propagated aggregable property $\text{agg}_{A'}(F', X')$ is correct, if (1) $F'$ can be applied on $A'$, (2) $X'_d$ is a determinant of $A'$ (if $A'$ is a measure attribute), and (3) $X'_f$ propagates the forbidden attributes defined for $A'$ (if $A'$ exists in $T$).

**Proposition 1 (Correctness of Summarizability Propagation Rules).** The propagation rules of Tables 12 and 13 are correct.

## 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 $T$.

### 4.1 Summarizable Attributes and Distribution Functions

Figure 4 illustrates the definition of summarizable attributes when some attribute $A$ of table $T$ is aggregated with some function $F$ for each partition of attributes $Z_2$. If 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 $G$, then we shall say that $A$ is summarizable with respect to $Z_1$ and $F$ using $G$.

The Definition 10 below formalizes Figure 4 and subsumes the definition of summarizability addressed in previous work [22, 27–29, 40]. For simplicity, we use the expression valid aggregate query for a query $\text{Agg}_T(F(A) | Z)$ such that the aggregable property $\text{agg}_A(F, X)$ 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 $T_1 = \text{Agg}_{T_1}(F(A) | Z_1)$ 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 query $\text{Agg}_{T}(F(A) | Z_2)$ is valid and the equation

$$\text{Agg}_{T_1}(G(F(A)) | Z_2) = \text{Agg}_{T}(F(A) | Z_2)$$

(1)

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

**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 $F(V_1 \cup \cdots \cup V_n) = G(F(V_1), \ldots, F(V_n))$, then $F$ is said to be distributive using function $G$ on partitioning $P$.

Distributive aggregation functions are also called decomposable aggregation functions in References [24, 57]: In the above definition, $F$ would be said to be decomposable with union and $G$. 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 any subset of $Z$.

The following proposition relates the previous definition of distributive functions to the notion of summarizable attributes. In the following, we assume to know the distributive properties of all usable aggregation functions as part of our system metadata.

**Proposition 2 (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 $G$.

**Example 10.** Consider attribute $\text{PROD_SKU}$ in table $\text{PRODUCT_LIST}$ with property $\text{agg}_{\text{PROD_SKU}}(\text{COUNT}, [\text{BRAND}, \text{COUNTRY}, \text{YEAR}])$. Since function $\text{COUNT}$ is distributive using function $\text{SUM}$ on table $\text{PROD_SKU}$ with any partitioning of the table, $\text{PROD_SKU}$ is summarizable with respect to grouping set $Z = [\text{BRAND}, \text{COUNTRY}, \text{YEAR}]$ and $\text{COUNT}$ using function $\text{SUM}$. Thus, if $Z_2 = [\text{COUNTRY}, \text{YEAR}]$ and $T_1 = \text{Agg}_{\text{PRODUCT_LIST}}(\text{COUNT}(\text{PROD_SKU}) | Z)$, then the following equation holds:

$$\text{Agg}_{\text{PRODUCT_LIST}}(\text{COUNT}(\text{PROD_SKU}) | Z_2) = \text{Agg}_{T_1}(\text{SUM}(\text{COUNT}(\text{PROD_SKU})) | Z_2).$$

However, function distributivity is a sufficient but not a necessary condition for summarizability, as shown by the following proposition, which defines a weaker sufficient condition for summarizability with $\text{COUNT\_DISTINCT}$, which is not distributive using $\text{SUM}$.
Table 14. Propagation Rule for Aggregate Operation Preserving Summarizability

| Query on T(S) | New propagation rule for inferring the aggregable properties of new measure attribute \( A' \) in the result \( T_r(S_r) \) | User action |
|---------------|-------------------------------------------------|-------------|
| \( \text{Agg}_T(F(A) \mid Y) \) | new measure attribute \( A' = F(A) \) and \( \text{agg}_{\text{new}}(F, X) \) holds in \( T \): if \( G \) can be applied on \( A' \) (Table 9) and \( F \) is distributive using \( G \) then \( \text{agg}_{\text{new}}(G, X') \) holds in \( T_r \) with \( X'_d \) determinant for \( A' \) and \( X'_d = X_f \cap Y \). if \( F = \text{COUNT}\_\text{DISTINCT} \) and \( K \) is a minimal subset of \( Y \) such that \( K \cup \{A\} \mapsto Y \) in \( T \) then \( \text{agg}_{\text{new}}(\text{SUM}, X') \) holds in \( T_r \) with \( X'_d \) determinant for \( A' \) and \( X'_d = (X_f \cap Y) \cup K \). | Minimize \( X'_d \) Complete \( X'_d \) |

Proposition 3 (Summarizability of COUNT\_DISTINCT using SUM). Let \( T(S) \) be an analytic table with a set of dimension attributes \( S_D \) and an aggregable attribute \( A \). Let \( T_1 = \text{Agg}_T(\text{COUNT}\_\text{DISTINCT}(A) \mid Z_1) \) be a valid aggregate query over \( T \). If \( Z_2 \subseteq Z_1 \) and the LFD \( Z_2 \cup \{A\} \mapsto Z_r \) holds in \( T \), then the following equation is true:

\[
\text{Agg}_{T_1}(\text{SUM}(\text{COUNT}\_\text{DISTINCT}(A)) \mid Z_2) = \text{Agg}_T(\text{COUNT}\_\text{DISTINCT}(A) \mid Z_2).
\]

(2)

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 \( Z_2 \).

Observe that if Equation (2) holds for any subset \( Z_2 \subset Z_1 \), then \( A \) is summarizable with respect to grouping set \( Z_1 \) and COUNT\_DISTINCT using function SUM (Definition 10). Proposition 3 can be generalized to other duplicate-sensitive functions like SUM\_DISTINCT using SUM.

Example 11. Let \( T_1 = \text{Agg}_{\text{PRODUCT}\_\text{LIST}}(\text{COUNT}\_\text{DISTINCT}(\text{PROD}\_\text{SKU}) \mid Z_1) \) where \( Z_1 = \{\text{BRAND}, \text{COUNTRY}, \text{YEAR}\} \). Function COUNT\_DISTINCT is only distributive using SUM if no pair of partitions share the same value. This is not the case with partitioning \( Z_1 \) (e.g., there exist two partitions of \( Z_1 \) with the same product "cz-shirt-s"), so by Proposition 2, attribute PROD\_SKU is not summarizable with respect to grouping set \( Z_1 \) and function COUNT\_DISTINCT using SUM. However, for \( Z_2 = \{\text{BRAND}, \text{YEAR}\} \), we have \( Z_2 \cup \text{PROD}\_\text{SKU} \mapsto Z_1 \) in table PRODUCT\_LIST. So, by Proposition 3, Equation (2) is valid for that particular subset \( Z_2 \) of \( Z_1 \). Intuitively, as long as we keep the attributes of \( Z_2 \) in the grouping set, a PROD\_SKU will not be double counted in the result of each partition.

4.2 Controlling Attribute Summarizability Using Aggregable Properties

Given the result of an aggregate query \( T_1 = \text{Agg}_T(F(A) \mid Z_1) \) over some attribute \( A \), we want to automatically compute the subset of dimension attributes \( X' \subseteq Z_1 \) of \( T_1 \) such that \( \text{agg}_{\text{new}}(G, X') \) holds in \( T_1 \), and for any \( Z_2 \) such that \( Z_1 - X' \subseteq Z_2, \text{Agg}_{T_1}(G(F(A)) \mid Z_2) = \text{Agg}_T(F(A) \mid Z_2) \).

Definition 12 (Summarizability Preserving Aggregable Property). Let \( T(S) \) be an analytic table and \( T_1 = \text{Agg}_T(F(A) \mid Z_1) \) be the result of a valid aggregate query. We say that \( \text{agg}_{\text{new}}(G, X') \) preserves the summarizability of \( A \) with respect to grouping set \( Z_1 \) if, for any subset \( Z_2 \) such that \( Z_1 - X' \subseteq Z_2 \), attribute \( A \) is summarizable in \( T \) with respect to grouping set \( Z_2 \) and function \( F \) using \( G \).

In Table 14, we present a propagation rule for aggregation that preserves summarizability. Its correctness shown in the next proposition is based on Propositions 2 and 3.

Proposition 4 (Propagation of Aggregable Properties with Summarizability Preservation). Let \( T(S) \) be an analytic table with dimension attributes \( S_D \subseteq S \) and let \( T_r = \text{Agg}_{T_r}(F(A) \mid Y) \) be the result of a valid aggregate query. Then the aggregable properties inferred by the rule of Table 14 for attribute \( A' = F(A) \) preserve the summarizability of \( A \) with respect to grouping set \( Y \) and are correct.
For the second condition in Table 14, observe that there might exist several minimal subsets of attributes \( K_i \) of \( Y \) such that \( K_i \cup \{ A \} \) determines all attributes of \( Y \) in \( T \). These sets \( K_i \) can easily be computed using the attribute graphs of the corresponding dimensions.

**Example 12.** Let \( T_1 = \text{Agg}_{\text{PRODUCT\_LIST}}(\text{COUNT\_DISTINCT(\text{PROD\_SKU}) | Y}) \) where \( Y = \{ \text{BRAND, COUNTRY, YEAR} \} \) and \( X_f = \emptyset \). By the second condition in Table 14, the only aggregation that holds among the attributes of \( Y \) is \( \text{BRAND} \mapsto \text{COUNTRY} \). Thus, there is a single minimal set \( K = \{ \text{BRAND, YEAR} \} \) such that \( K \cup \{ \text{PROD\_SKU} \} \) determines all attributes in \( Y \). Thus, \( X_f' = (X_f \cap Y) \cup K = K \). By default, \( X_f' = \{ \text{BRAND, YEAR} \} \), which is already minimal, so no user action is needed. Finally, \( X' = Y \cap (X_d \cup X_f')^+ - X_f' = \{ \text{COUNTRY} \} \) and \( \text{Agg}_{\text{COUNT\_DISTINCT(\text{PROD\_SKU})}(\text{SUM | COUNTRY})} \) preserves the summarizability of \( \text{PROD\_SKU} \) with respect to grouping set \( Y \).

**Comparisons to previous work on summarizability.** Consider the following sequence of operations: \( T_1 = \text{Agg}_{\text{STORE\_SALES}}(\text{SUM(AMOUNT) | \{CITY, STATE, COUNTRY, YEAR\}}) \) and \( T_2 = \text{Agg}_{T_1}(\text{SUM(AMOUNT) | \{COUNTRY, YEAR\}}) \). In Reference [29], the second aggregation is considered to be incorrect, because it violates some completeness condition requiring that (a) no store is missing in \( \text{STORE\_SALES} \) with respect to \( \text{SALES\_ORG} \) and (b) attribute \( \text{STATE} \) cannot be optional. In Reference [40], the same above condition (b) is imposed. In addition, a more restrictive definition of distributive function is used (no function \( G \) is allowed) that prevents the use of \( \text{COUNT\_function} \). In Reference [27], the second aggregation would not be allowed because the \( \text{STORE\_SALES} \) does not satisfy the Generalized Multidimensional Normal Form (i.e., no context dependency can be expressed to characterize all null values in \( \text{STATE} \) and above condition (a) is not satisfied). In Reference [22], the second aggregation would also be rejected, because the \( \text{SALES\_ORG} \) hierarchy is not strict with respect to functional dependencies with nulls (a non-null city value can have multiple state values). By contrast, our treatment of null values in aggregation operations and our propagated aggregable properties on \( T_1 \) properly detect that the second aggregation is correct. A more detailed comparison can be found in the long version of this article [46].

### 4.3 Generalized Attribute Summarizability

In the previous sections, we defined the summarizability of an attribute through a sequence of two aggregation queries. In Definition 13, we extend the summarizability of an attribute to a sequence of an arbitrary analytic query followed by an aggregation query. This case was not addressed by previous work on summarizability.

**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 \( \text{agg}_A(F, X) \) be an aggregable property of attribute \( A \) that holds in \( T \) and \( T_1 \) and \( X \subseteq Z \subseteq (S \cap S_1) - X \). If for any two aggregate queries \( Q \left( T \right) = \text{Agg}_F(A | Z') \) and \( Q \left( T_1 \right) = \text{Agg}_F(A | Z') \) such that \( Z \subseteq Z' \), both queries are valid and for any two tuples \( t_1 \in Q \left( T \right) \) and \( t_2 \in Q \left( T_1 \right) \), then we have \( t_1.Z' = t_2.Z' \Rightarrow t_1.F(A) = t_2.F(A) \), then \( A \) is said to be \( G\text{-summarizable} \) in \( T \) with respect to query \( Q \), grouping set \( Z \) and function \( F \).

It is easy to show that \( Q \left( T \right) \) and \( Q \left( T_1 \right) \) in the previous definition are always valid. Since the aggregable property \( \text{agg}_A(F, X) \) holds in \( T \) and \( T_1 \) and \( X \subseteq Z \), this property also holds for all sets \( Z' \) where \( Z \subseteq Z' \) and both queries are valid.

The Section 4.3 is illustrated in Figure 5. We make a few observations. First, unlike Definition 10, \( T_2 \) 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 \( F \). They are used in the definition to determine which aggregate queries
Consider a fact table with respect to \( x \) and \( b \), let \( f \) be an attribute of \( F = \{A, B, C\} \). Consider now the fact tables \( 1 \) and \( 2 \) are \( G \)-summarizable in \( A \) with respect to \( Q \). Any attribute \( A \) is \( G \)-summarizable with respect to \( Q \) and \( F \).

**Example 13.** Consider a fact table \( T \) (in Table 15) defined over two dimension \( D_1 \) and \( D_2 \) with dimension attributes \( A_1, A_2, A_3 \) from dimension \( D_1 \) (where \( A_1 \not< A_2 \not< A_3 \) and \( A_2 \not\rightarrow A_3 \)) and \( B_1, B_2 \) from dimension \( D_2 \) (where \( B_1 \not< B_2 \)). We shall say that within table \( T \), \( A_3 \) is the highest attribute of \( D_1 \) while \( B_2 \) is the highest attribute of \( D_2 \).

Take query \( Q_1(T) = \text{filter}_T ([A_3 = c_1] \mid [A_3]) \) whose result table \( T_1 \) is displayed in Table 15. If we take a valid grouping set \( Z = \{A_3\} \), then for each \( p \in \text{dom}(A_3) \) and partition \( T^p = \sigma_{A_3 = p}(T) \) of \( T \) we have either \( Q_1(T^p) = T^p \) 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 have either \( \Pi_{Z'}(Q_1(T^p)) = \Pi_{Z'}(T^p) \) or \( \Pi_{Z'}(Q_1(T^p)) = \emptyset \) is a projection without duplicate elimination. Therefore, any valid aggregation query with grouping attributes \( Z' \) returns, for each partition of \( T_1 \) defined by \( Z' \), the same result when it is applied to \( T \) or \( T_1 \). 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 \( Q_2(T) = \text{filter}_T ([M \in \{x_3, x_4\}] \mid [M]) \). 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 14.** Consider now the fact tables \( T \) and \( T' \) (displayed in Figure 6) defined over the same dimensions \( D_1 \) and \( D_2 \) as before. Take the left-merge query \( Q(T, T') = T \bowtie_Y T' = T_1 \) where \( Y = \{A_1, A_2, B_1, B_2\} \). Any attribute \( A \) of \( T \) is \( G \)-summarizable in \( T \) with respect to \( Q \) and any function \( F \) applicable to \( A \), because the duplicate preserving projection of \( T_1 \) on the attributes of \( T \) is equal to table \( T \) (hence, the condition of Definition 13 is always satisfied).

ACM Journal of Data and Information Quality, Vol. 15, No. 2, Article 12. Publication date: June 2023.
We can obtain the same table by applying a right-merge $T_1 = Q'(T', T) = T' \bowtie_Y T$. Let us then 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 $T'^{\text{high}, e_1}$ is different from the duplicate-preserving projection of $T'^{\text{high}, e_1}$ 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 $Y$.

Now take the same right merge query $T_1 = T' \bowtie_Y T$ (equal to $T \bowtie_Y T'$) as before applied to the tables displayed in Figure 7. 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 $(b_3, e_1)$ 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., $\text{COUNT\_DISTINCT}$, $\text{MIN}$, $\text{MAX}$).

**Proposition 5 (Queries Satisfying G-summarizability).** Let $Q$ be a unary or binary analytic query with some input table $T(S)$ returning a table $T_1(S_1)$ and $S_D$ be the dimension attributes in $S \cap S_1$. Let $Z$ be a subset of $S_D$, and $A'$ be an attribute in $S \cap S_1$ such that $\text{agg}_{A'}(F, X)$ and $\text{agg}_{A'}(F, X_1)$ hold in $T$ and $T_1$, respectively. Then, $A'$ is G-summarizable in $T'$ with respect to query $Q$, grouping set $Z$, and function $F$ in the following cases:

- **Unary queries:**
  1. $Q = \text{Filter}_T(P \mid Y)$, $Y \subseteq S_D$, $A' \in S$ and $Z = (S_D - X_1) \cup Y$.
  2. $Q = \text{Project}_T(Y, f(Z') \rightarrow m_0)$, $A' \in Y$ and $Z = S_D - X_1$.
  3. $Q = \text{Agg}_T(G(A) \mid Y)$, $A' \in Y$, $Z = Y - X_1$ and $F \in \{\text{MIN, MAX, COUNT\_DISTINCT}\}$.
  4. $Q = \text{Pivot}_T(A \mid Y)$, $A' \in S - Y - \{A\}$, $Z = S_D - X_1 - Y - \{A\}$ and $F \in \{\text{MIN, MAX, COUNT\_DISTINCT}\}$.

- **Merge queries:** First, for all merge queries, we apply the rule restricting the aggregation function to duplicate insensitive function when $Y$ does not determine the attributes of the second argument tuple $T': Y \not\rightarrow S'$ then $F \in \{\text{MIN, MAX, COUNT\_DISTINCT}\}$. 
Then, let $Y^{\text{top}} \subseteq Y$ denote the subset of highest attributes in the set of join attributes $Y$. We can define the following rules:

(1) $Q_1 = T \bowtie_Y T'(S')$, $A' \in S$, $Z = S_D - X_1$
(2) $Q_2 = T \bowtie_Y T'(S')$ or $Q_2 = T \bowtie_Y T'(S')$:
   (a) If $A' \in Y$ and for all non-empty partitions $T'' = \sigma_{Y^{\text{top}}=y}(T')$ of $T'$, then the corresponding partition $T'' = \sigma_{Y^{\text{top}}=y}(T)$ of $T$ is empty or $\pi_Y(T'') = \pi_Y(T'')$, and then $Z = (S_D - X_1) \cup Y^{\text{top}}$
   (b) If $A' \in S-Y$ and for all non-empty partitions $T'' = \sigma_{Y-y}(T')$ of $T'$ and corresponding partitions $T'' = \sigma_{Y-y}(T)$ of $T$, $\pi_Y(T'') \subseteq \pi_Y(T'')$, then $Z = (S_D - X_1) \cup Y^{\text{top}}$.
(3) $Q_3 = T \bowtie_Y T'(S')$, $A' \in S$: if for all non-empty partitions $T'' = \sigma_{Y^{\text{top}}=y}(T')$ and corresponding partitions $T'' = \sigma_{Y^{\text{top}}=y}(T)$, $\pi_Y(T'') \subseteq \pi_Y(T'')$ then $Z = (S_D - X_1) \cup Y^{\text{top}}$ else $Z = (S_D - X_1) \cup Y$.

Set queries:

(1) $Q_1 = T \cup T'$, $A' \in S$ and if $\pi_{Y^{\text{top}}}(T) \cap \pi_{Y^{\text{top}}}(T') = \emptyset$, then $Z = S_D - X_1 \cup Y^{\text{top}}$, otherwise $Z = Y$.
(2) $Q_2 = T - T'$, $A' \in S$ and if all partitions $\sigma_{Y^{\text{top}}=y}(T)$ are equal to or disjoint with $\sigma_{Y^{\text{top}}=y}(T')$, then $Z = (S_D - X_1) \cup Y^{\text{top}}$, otherwise $Z = Y$.

Observe that for merge and set queries, we choose the highest dimension $Y^{\text{top}}$ as candidates for checking the conditions on the data partitions $T'$ and $T''$. In fact, we might check these partition containment conditions for any subset $Y'$ of attributes from $Y$ or $S_D$ instead of $Y^{\text{top}}$ and identify the minimal candidates for which these conditions hold. There are two main reasons for choosing $Y^{\text{top}}$. First, the choice of the highest attributes $Y^{\text{top}}$ is based on the realistic hypothesis that the majority of analytic queries aggregate values along these attributes plus possibly other lower attributes. Second, 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. The systematic exploration of all attribute subsets $Y'$, even with efficient pruning techniques, may take too much time in an interactive data exploration session.

Related work on query optimization and query rewriting using views. The G-summarizability property is related to query rewriting techniques developed for optimizing SQL query plans with aggregations [10, 15, 56, 57]. For example, the invariant grouping property defined in Reference [10] can be viewed as a particular case of G-summarizability. Let $T_1 = T \bowtie T'$ be the result of an equi-join between two tables $T$ and $T'$ (the property can be generalized to more than two tables) and $Q'(T_1) = \text{Agg}_{T_1}(F(\Lambda) \mid Z)$ be the result of an aggregate query $Q'$ on $T_1$. Then, table $T$ has the invariant grouping property with respect to $Q'(T_1)$ if (1) $T$ contains the aggregated attribute $\Lambda$, (2) the equi-join $T \bowtie T'$ is defined on foreign key attributes of $T$ referencing $T'$, and (3) all join attributes of $T$ are also grouping attributes in $Z$. This property makes it possible to push the aggregation query $Q'(T_1)$ down to $T$ and evaluate the new query $Q'(T) \bowtie T'$ without changing the final query result. The G-summarizability property of attributes generalizes this condition to a larger set of queries. As illustrated by Figure 5, if $T_1 = Q(T)$ and attribute $\Lambda$ is G-summarizable in $T$ with respect to some query $Q$ (which is not restricted to equi-joins), grouping set $Z$ and function $F$, then any aggregation query $Q'(T_1) = \text{Agg}_{T_1}(F(\Lambda) \mid Z')$, where $Z \subseteq Z'$ can be pushed down to $T$ such that the aggregated values of all tuples in $Q'(T_1)$ and $Q'(T)$ that match on the grouping attributes $Z'$ are equal. Observe that depending on $Q$, the result tables of $Q'(T)$ and $Q'(T_1)$ might contain tuples that are not shared by both tables (e.g., $Q$ is a union that adds new tuples to $T$ or $Q$ filters out a subset of $T$).

A second related research area concerns the problem of Answering Queries Using Views [1]. Formally, this problem consists in studying, for a given query $Q$ and a set of views $V$, the generation
of maximally contained and equivalent rewritings (queries) of \( Q \) using only the views in \( V \). This problem has also been studied in the setting where queries, views and rewritings might contain aggregate functions [2, 11, 17, 48, 52]. For example, Reference [2] proposes a complete algorithm that constructs central rewritings (only one view contributes to the value of the aggregated attribute) for a given query \( Q \). When considering the schema in Figure 5, G-summarizability is a way to check, for some aggregation query \( Q_1(T) = \text{Agg}_T(F(A)|Z') \) and view \( T_1 = Q(T) \) on \( T \), if the query \( Q_1(T_1) = \text{Agg}_{T_1}(F(A)|Z') \) using view \( T_1 \) is a central and coherent rewriting of \( Q_1(T) \) where \( t.F(A) = t'.F(A) \) for all tuples \( t \) in \( Q_1(T) \) and \( t' \) in \( Q_1(T_1) \) such that \( t.Z' = t'.Z' \) (\( Q_1(T_1) \) may be a subset of \( Q_1(T) \)). Thus, our propagation rules can be used to decide if an aggregate query \( Q_1 \) is its "own" rewriting using any given analytic view \( Q(T) \). G-summarizability offers a more restricted scope than the answering of queries using views, which finds all maximally contained or equivalent rewritings for any conjunctive aggregate query using one or more views. However, the reverse is also true, since G-summarizability accepts a scope of views, defined by any analytic query (including pivot and outer joins), that is broader than aggregate conjunctive queries. To conclude, the goal of our propagation rules is not to produce equivalent rewritings, but to characterize for a given query (view) \( T_1 = Q(T) \) the set of aggregation queries that are coherent with the results obtained by the same query on \( T \).

### 4.4 Controlling G-summarizability Using Aggregable Properties

Table 16 extends the rules of Table 12 for unary queries. As defined in Proposition 5, this extension consists in (1) adding new forbidden attributes (inferred from \( Z \)) to \( X_f' \) and (2) restricting the applicable functions \( F \). The rule for \textit{Project} is unchanged with respect to Table 12. For some aggregate query \( Q = \text{Agg}_T(G(A) \mid Y) \), the G-summarizability of attribute \( G(A) \) is already covered by the summarizability rule in Table 14. For grouping attributes \( Y \), the only refinement to the rule in Table 12 is to restrict the scope of \( F \). The same observation applies for a pivot query.

| Query on \( T(S) \) | Propagation rule for inferring the aggregable properties for attributes \( X' \in S \cap S_r \) of the result \( T_f(S_r) \) | User action |
|---------------------|-------------------------------------------------|-------------|
| \textit{Filter}_T(P \mid Y) | \begin{enumerate} \item dimension attribute \( X' \in SD \), \( Y \subseteq SD \) and \( \text{agg}_{A'}(F, X) \) holds in \( T \); \item measure attribute \( X' \in S - SD \), \( Y \subseteq SD \) and \( \text{agg}_{A'}(F, X) \) holds in \( T \); \end{enumerate} | \begin{enumerate} \item \( \text{Minimize } \text{agg} \); \item \( \text{Complete } \text{agg} \); \end{enumerate} |
| \textit{Project}_T(Y, f(Z) \rightarrow m) | \begin{enumerate} \item dimension attribute \( X' \in Y \) and \( \text{agg}_{A'}(F, X) \) holds in \( T \); \item new pivot attribute \( X' \in Y \) and \( \text{agg}_{A'}(F, X) \) holds in \( T \); \end{enumerate} | \begin{enumerate} \item \( \text{Minimize } \text{agg} \); \item \( \text{Minimize } X_f' \); \item \( \text{None} \); \end{enumerate} |
| \textit{Agg}_T(G(A) \mid Y) | \begin{enumerate} \item dimension attribute \( X' \in Y \) and \( \text{agg}_{A'}(F, X) \) holds in \( T \); \item new pivot attribute \( X' \in Y \) and \( \text{agg}_{A'}(F, X) \) holds in \( T \); \end{enumerate} | \begin{enumerate} \item \( \text{Minimize } X_f' \); \item \( \text{None} \); \end{enumerate} |
Tables 17 and 18 extend the propagation rules of Table 13 for binary queries. As for unary queries, the extension mainly consists in adding new forbidden attributes to $X'_f$ (inferred from $Z$ as defined in Proposition 5) and applying new data constraints as defined in Proposition 5.

The aggregable properties computed by the rules of Table 17 indicate that the correctness of aggregation for attributes of $T_1$ in $T−Y$ (respectively, $T′−Y$) is defined with respect to $T$ (respectively, $T'$). For attributes in $Y = \{A_1, A_2, B_1, B_2\}$, we postulated that the correctness of aggregation is defined with respect to the “first” table occurring in the query. Thus, the equivalent queries $T \bowtie_Y T'$ and $T' \bowtie_Y T$ induce different aggregable properties for $Y$ attributes in the result table. We illustrate this

| Merge query on $T(S)$ and $T'(S')$ | Propagation rule for inferring the aggregable properties of attributes $\alpha' \in S$ in the result $T_r(S_r)$ | User action |
|-----------------------------------|---------------------------------------------------------------|-------------|
| $T_r = T \bowtie_Y T'$ if $Y \neq S'$ then $F \in \{\text{COUNT}_D, \text{MIN}, \text{MAX}\}$ | if $\pi_Y(S_{top} = u)(T) = \pi_Y(S_{top} = u)(T') = \emptyset$ for all non-empty partitions $\pi_Y(S_{top} = u)(T')$: then $agg_A'(F, X')$ holds in $T_r$ with $X'_f = X_f \cup Y^{top}$ else $agg_A'(F, X')$ holds in $T_r$ with $X'_f = X_f \cup Y$. | Complete $X'_f$ |
| $T_r = T \bowtie_Y T'$ if $Y \neq S'$ then $F \in \{\text{COUNT}_D, \text{MIN}, \text{MAX}\}$ | if $\pi_Y(S_{top} = u)(T) = \pi_Y(S_{top} = u)(T') = \emptyset$ for all non-empty partitions $\pi_Y(S_{top} = u)(T')$: then $agg_A'(F, X')$ holds in $T_r$ with $X'_f = X_f \cup Y^{top}$ else $agg_A'(F, X')$ holds in $T_r$ with $X'_f = X_f \cup Y$. | Complete $X'_f$ |
| $T_r = T \bowtie_Y T'$ if $Y \neq S'$ then $F \in \{\text{COUNT}_D, \text{MIN}, \text{MAX}\}$ | if $\pi_Y(S_{top} = u)(T) = \pi_Y(S_{top} = u)(T') = \emptyset$ for all non-empty partitions $\pi_Y(S_{top} = u)(T')$: then $agg_A'(F, X')$ holds in $T_r$ with $X'_f = X_f \cup Y^{top}$ else $agg_A'(F, X')$ holds in $T_r$ with $X'_f = X_f \cup Y$. | Complete $X'_f$ |

Table 17. Propagation Rules for Merge Operations with G-summarizability

ACM Journal of Data and Information Quality, Vol. 15, No. 2, Article 12. Publication date: June 2023.
on the query and data of Figure 6 that produce Table 1: The left-merge rule applies to attributes of \{A_1, A_2, B_1, B_2\} = Y while the right-merge rule applies to attributes A_3 and M'.

- For A_1: Suppose that \(agg_A(F, X)\) holds in T where \(X = \{A_2, B_1, B_2\}\) and \(X_f = \emptyset\). Then by the propagation rule, since \(X'_f = X_f\), \(agg_A(F, X')\) holds in T_1 with \(X' = \{A_2, A_3B_1, B_2\}\). Thus, A_1 can be aggregated along any dimension attribute in T_1. Note that aggregation along A_3 is also allowed as a side effect although A_3 \(\not\in T\).

- For A_3: Suppose that \(agg_A(F, X)\) holds in T where \(X = \{A_1, A_2, B_1, B_2\}\) and \(X_f = \emptyset\). We have \(Y \mapsto S'\), since by assumption in Example 13, A_2 \(\mapsto A_3\). So, F is also applicable to A_3 in T_1. However, the partition containment condition is violated because of tuples t_3 and t_4. So \(X'_f = X_f \cup Y\) and \(agg_A(F, X')\) holds in T_1 with \(X' = \emptyset\), which means that no aggregation is possible on A_3.

- For M': Suppose that \(agg_M(F, X)\) holds in T' where \(X = \{A_1, A_2, B_1, B_2\}\) and \(X_f = \emptyset\). The partition containment condition is violated again. The default value of \(X_d\) is \(\{A_1, A_2, B_1, B_2\}\), which is minimal. Then by the propagation rule, \(X'_f = X_f \cup Y = \{A_1, A_2, B_1, B_2\}\) and \(agg_M(F, X')\) holds in T_1 with \(X' = (X_d \cup X_f')\) \(\neq X_f' = \{A_3\}\), which means that aggregation on M' is only possible along A_3, which gives the dimension hierarchy means that no aggregation is possible.

Most rules of Tables 17 and 18 use partition containment conditions over the operand data to compute aggregable properties. Thus, at the time when T_r is computed using a binary query, aggregable properties express the aggregations that are possible. For instance, in the example of Figure 6, since the partition containment condition over T and T' used in the merge rule is not satisfied, some aggregations on A_3 and M' are forbidden. However, if tuples t_3 and t_4 are later on deleted from table T', then the partition containment condition becomes satisfied and more aggregations would be allowed. Thus, our rules compute aggregable properties that are valid for a given snapshot of the operand data used to compute T_r. An interesting question that is left for future work is, given a table T_r, how to maintain its aggregable properties with respect to future changes in T or T'.

The following proposition refines the propagation rules for aggregable properties of Section 3.3 using the results of Proposition 5 to guarantee the G-summarizability of attributes.
Proposition 6 (Aggregable Properties with G-summarizability). Let \( Q \) be an analytic query with some input table \( T(S) \) returning a table \( T_r(S') \), and \( S_D \) be all the dimension attributes of \( S \cap S' \). Then, for all queries \( Q \) satisfying the conditions of Tables 16, 17, and 18, the aggregable property \( \text{agg}_{A}(F, X') \) holds in \( T_r \) and is such that for all \( Z \) where \( S_D - X' \subseteq Z \), \( A' \) is G-summarizable in \( T \) with respect to query \( Q_r \), grouping set \( Z \), and function \( F \).

5 CASE STUDY

We illustrate the impact of our complementary results on summarizability and G-summarizability using concrete examples of interactive sessions.

5.1 Interactive Session 1

We first show how to control the correctness of the two aggregation queries of the Interactive session 1 presented in the introduction by computing the aggregable properties on the attributes of tables \( T_1 \) and \( T_1' \) that result from these aggregate queries.

- \( T_1 = \text{Agg}_{\text{DEM}}(\text{COUNT\_DISTINCT}(\text{CITY}) \mid Y) \) with \( Y = \{\text{STATE}, \text{COUNTRY}\} \). By default, \( \text{agg}_{\text{CITY}}(\text{COUNT\_DISTINCT}, X) \) holds in \( \text{DEM} \) with \( X = \{\text{STATE}, \text{COUNTRY}, \text{YEAR}\} \) and \( X_f = \emptyset \). Attribute \( \text{NB\_CITIES} \) is of category \( \text{NUM} \), so by Table 9, function \( \text{SUM} \) is applicable. Following the aggregate rule of Table 14, \( K = \{\text{STATE}, \text{COUNTRY}\} \) is the minimal subset of \( Y \) such that \( K \cup \text{CITY} \) determines all attributes of \( Y \) in \( \text{DEM} \). Thus, \( X_f' = (X_f \cap Y) \cup K = K \). By default, \( X_f' = \{\text{STATE}, \text{COUNTRY}\} \), which is minimal, so no user action is required to minimize \( X_f' \). Thus, \( X' = (X_f' \cup K)^+ - K = \emptyset \), \( \text{agg}_{\text{NB\_CITIES}}(\text{SUM}, \emptyset) \) holds in \( T_1 \) and no further aggregation is possible over \( \text{NB\_CITIES} \), which explains why table \( T_1 \) is not correct.

- \( T_1' = \text{Agg}_{\text{DEM}}(\text{COUNT}(\text{CITY}) \mid Y) \) with \( Y = \{\text{STATE}, \text{COUNTRY}\} \). By Table 9, function \( \text{SUM} \) is applicable to \( \text{NB\_CITIES} \). By default, property \( \text{agg}_{\text{CITY}}(\text{COUNT}, X) \) holds in \( \text{DEM} \) with \( X = \{\text{STATE}, \text{COUNTRY}, \text{YEAR}\} \) and \( X_f = \emptyset \). By default, \( X_f' = \{\text{STATE}, \text{COUNTRY}\} \), which is minimal, so no user action is required to minimize \( X_f' \). Following the aggregate propagation rule of Table 14, since \( \text{COUNT} \) is distributive using \( \text{SUM}, \text{agg}_{\text{NB\_CITIES}}(\text{SUM}, \{\text{STATE}, \text{COUNTRY}\}) \) holds in \( T_1' \) with \( X_f' = X_f \cap Y = \emptyset \). Thus, \( X' = (X_f' \cup K)^+ = \{\text{STATE}, \text{COUNTRY}\} \), that is, \( \text{NB\_CITIES} \) is aggregable along \( \text{STATE} \), which explains why table \( T_1' \) is correct.

5.2 Interactive Session 2

We next show how to control the correctness of the aggregation queries of the Interactive session 2 of Figure 1 by propagating aggregable properties on the result of each query (the names of the tables are those used in Figure 1).

- \( T_3 = \text{Filter}_{\text{STORE\_SALES}}(P) \mid Y \) with \( Y = \{\text{COUNTRY}, \text{YEAR}\} \). By default, focusing on function \( \text{SUM} \), property \( \text{agg}_{\text{AMOUNT}}(\text{SUM}, X) \) holds in \( \text{STORE\_SALES} \) with \( X = \{\text{STORE\_ID}, \text{CITY}, \text{STATE}, \text{COUNTRY}, \text{YEAR}\} \), \( X_d = \{\text{STORE\_ID}, \text{YEAR}\} \) and \( X_f = \emptyset \). By default, \( X_f' = X_d \), which is not anymore minimal, because all tuples have \( \text{YEAR} = '2018' \). So, the user should remove attribute \( \text{YEAR} \) from \( X_f' \), and add it to \( X_f' \), which is already set to \( X_f \cup Y \) by the propagation rule Table 16 for the filter. Thus, property \( \text{agg}_{\text{AMOUNT}}(\text{SUM}, X') \) holds in \( T_3 \), with \( X_f' = \{\text{COUNTRY}, \text{YEAR}\} \), \( X_f' = \{\text{STORE\_ID}, \text{CITY}, \text{STATE}\} \), and \( X' = (X_f' \cup X_f')^+ - X_f' = \{\text{STORE\_ID}, \text{CITY}, \text{STATE}\} \). Thus, any further aggregation on \( \text{AMOUNT} \) must have both \( \text{COUNTRY} \) and \( \text{YEAR} \) in its grouping set.

- \( T_4 = \text{Agg}_{\text{DEM}}(\text{SUM}(\text{AMOUNT}) \mid Y) \) with \( Y = \{\text{CITY}, \text{STATE}, \text{COUNTRY}, \text{YEAR}\} \). By default, \( X_f' = \{\text{CITY}, \text{STATE}, \text{COUNTRY}, \text{YEAR}\} \), which is minimal. Using the aggregate rule of Table 14, since \( \text{SUM} \) is distributive, we get the property \( \text{agg}_{\text{SUM}(\text{AMOUNT})}(\text{SUM}, X') \) that holds in \( T_4 \), with \( X_f' = X_f \cap Y = \{\text{COUNTRY}, \text{YEAR}\} \) and \( X' = (X_f' \cup X_f')^+ - X_f' = \{\text{CITY}, \text{STATE}\} \).
• $T_5 = T_4 \supseteq_Y \text{DEM with } Y = \{\text{CITY, STATE, COUNTRY, YEAR}\}$. We start with attribute $\text{SUM(AMOUNT)}$. By the rule of Table 17, we get the property $\text{agg}_{\text{SUM(AMOUNT)}}(\text{SUM}, X')$ that holds in $T_5$, with $X'_d, X'_f$, and $X'$ that are the same as in table $T_4$. We now consider attribute $\text{POP}$. As we have seen in Example 8, property $\text{agg}_{\text{POP}}(\text{SUM}, X)$ holds in $T_5$, with $X_d = \{\text{CITY, STATE, COUNTRY, YEAR}\}$ and $X_f = \{\text{YEAR}\}$. By default, $X'_d = \{\text{CITY, STATE, COUNTRY, YEAR}\}$, which is minimal. Using $Y^{\text{top}} = \{\text{COUNTRY, YEAR}\}$, by the rule of Table 17, we must evaluate expressions $E_1 = \pi_Y(\sigma_{\text{COUNTRY} = \text{USA}}(\text{DEM}))$ and $E_2 = \pi_Y(\sigma_{\text{COUNTRY} = \text{USA}}(T_4))$ for all non-empty partitions $\sigma_{\text{COUNTRY} = \text{USA}}(T_4)$ and check if either $E_1 \subseteq E_2$ or $E_1 = \emptyset$. However, for the non-empty partition $\sigma_{\text{COUNTRY} = \text{USA} \land \text{YEAR} = 2018}(T_4)$, we have $\pi_Y(\sigma_{\text{COUNTRY} = \text{USA} \land \text{YEAR} = 2018}(\text{DEM})) \not\subseteq \pi_Y(\sigma_{\text{COUNTRY} = \text{USA} \land \text{YEAR} = 2018}(T_4))$ (because the city of "Palo Alto" is missing in $T_4$). Thus, property $\text{agg}_{\text{POP}}(\text{SUM}, X')$ holds in $T_5$ with $X'_d = X_f \cup Y$ and $X' = (X'_d \cup X'_f)^+ - X'_f = \{\text{COUNTRY, YEAR}\}$. This explains why the aggregation along attribute CITY leads to an incorrect result for attribute $\text{SUM(POP)}$.

5.3 Variant of Interactive Session 2

We next show how aggregable properties are propagated to the result of the queries of Variant of interactive session 2. There, the user first backtracked to DEM and ran the following queries:

• $\text{DEM'} = \text{Agg}_{\text{DEM}}(\text{SUM(POP)} ~ | ~ Y)$ with $Y = \{\text{STATE, COUNTRY, YEAR}\}$. The property $\text{agg}_{\text{POP}}(\text{SUM}, X)$ that holds in DEM is already given in the previous item. By default, $X'_d = \{\text{STATE, COUNTRY, YEAR}\}$, which is minimal. Following the rule of Table 14, since function $\text{SUM}$ is distributive, property $\text{agg}_{\text{POP}}(\text{SUM}, X')$ holds in DEM', with $X'_f = X_f = \{\text{YEAR}\}$ and $X' = (X'_d \cup X'_f)^+ - X'_f = \{\text{STATE, COUNTRY}\}$.

• $T'_5 = T_4 \supseteq_Y \text{DEM'}$ with $Y = \{\text{STATE, COUNTRY, YEAR}\}$. Let us compute the aggregable property for attribute $\text{SUM(POP)}$. By default, $X'_d = \{\text{CITY, STATE, COUNTRY, YEAR}\}$, which is, however, not minimal. So, the user should remove attribute CITY from $X'_d$ and add it to $X'_f$. Next, as for $T_5$ before, using $Y^{\text{top}} = \{\text{COUNTRY, YEAR}\}$, by the rule of Table 17, we must evaluate expressions $E_1 = \pi_Y(\sigma_{\text{CITY} = \text{MISSING}}(\text{DEM}'))$ and $E_2 = \pi_Y(\sigma_{\text{CITY} = \text{MISSING}}(T_4))$ for all non-empty partitions $\sigma_{\text{CITY} = \text{MISSING}}(T_4)$ and check if either $E_1 \subseteq E_2$ or $E_1 = \emptyset$. This condition is now satisfied, so no new attribute is added to $X'_f$. However, since $Y$ does not literally determine all the dimension attributes of $T_4$, $\text{SUM}$ is not applicable to $\text{SUM(POP)}$. Therefore, property $\text{agg}_{\text{SUM(POP)}}(F, X')$ holds in $T'_5$, with $X'_d = \{\text{CITY}\}$, $X'_d = \{\text{STATE, COUNTRY, YEAR}\}$, $X' = (X'_d \cup X'_f)^+ - X'_f = \{\text{STATE, COUNTRY, YEAR}\}$, and $F \in \{\text{COUNT\_DISTINCT, MIN, MAX}\}$, which explains why a summation of $\text{SUM(POP)}$ along CITY is incorrect.

Next, the user backtracked to table $T_3$ and ran the following queries (the resulting flow of interactive queries is in Figure 2):

• $T'_4 = \text{Agg}_{\text{AMOUNT}}(\text{SUM(AMOUNT)} ~ | ~ Y)$ with $Y = \{\text{STATE, COUNTRY, YEAR}\}$. Property $\text{agg}_{\text{AMOUNT}}(\text{SUM}, X)$ holds in $T_5$, with $X_f = \{\text{COUNTRY, YEAR}\}$. Thus, $T'_4$ is correct. We now focus on $\text{SUM(AMOUNT)}$. By default, $X'_d = \{\text{STATE, COUNTRY, YEAR}\}$, which is minimal. Using the aggregate rule of Table 14, since $\text{SUM}$ is distributive, property $\text{agg}_{\text{SUM(AMOUNT)}}(\text{SUM}, X')$ holds in $T'_4$, with $X'_f = X_f \cap Y = \{\text{COUNTRY, YEAR}\}$, and $X' = (X'_d \cup X'_f)^+ - X'_f = \{\text{STATE}\}$.

• $\text{SALES\_DEM\_USA} = T'_4 \supseteq_Y \text{DEM'}$ with $Y = \{\text{STATE, COUNTRY, YEAR}\}$. We focus on attribute $\text{SUM(POP)}$. By default, $X'_d = \{\text{STATE, COUNTRY, YEAR}\}$ is minimal. In addition, $Y$ literally determines all dimension attributes of $T'_4$, so $\text{SUM}$ is applicable to $\text{SUM(POP)}$. Next, with $Y^{\text{top}} = \{\text{COUNTRY, YEAR}\}$, the partition containment condition of the propagation rule is satisfied for all non-empty partitions $\sigma_{\text{CITY} = \text{MISSING}}(T'_4)$. Thus, $X'_f = X_f \cup Y^{\text{top}} = \{\text{COUNTRY, YEAR}\}$ and property $\text{agg}_{\text{SUM(POP)}}(\text{SUM}, X')$ holds in $\text{SALES\_DEM\_USA}$ with $X' = (X'_d \cup X'_f)^+ - X'_f = \{\text{STATE}\}$.
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 among 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 [21, 26, 29, 34, 40, 49]. 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 [22, 27–29, 40]. Our second definition introduces the new notion of G-summarizability that 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.

As one direction for future work, aggregable properties could be extended to handle other correctness issues of aggregate queries. Currently, we rely on 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 [37] (representing features) in addition to LFD.

APPENDIX

A PROOFS

Proposition 1 (Correctness of Summarizability Propagation Rules). The propagation rules of Tables 12 and 13 are correct.

Proof of Proposition 1. By Definition 9, we have to show that for each rule and new aggregable property $\text{agg}_{A'}(F', X')$: (1) $F'$ can be applied on $A'$, (2) $X'_d$ is a determinant of $A'$ (if $A'$ is a measure attribute), and (3) $X'_f$ “propagates” the forbidden attributes defined for $A'$ (if $A'$ exists in $T$). Suppose that whenever $A' \in T$, then $\text{agg}_{A'}(F, X)$ holds in $T$, for $X \subseteq S_D$.

Unary queries. We first show that the rules of Table 12 are correct.

(1) Filter queries: Let $T_r = \text{Filter}_T(P \mid Y)$. Since $T_r$ is a subset of $T$, $F$ is still applicable on $A'$ and $X'_f = X_f$ preserves all attributes in $X_f$. By default, $X'_d = X_d$. However, since filters remove tuples from the input table, it may happen that $X_d$ is not minimal for all instances of $T$ satisfying predicate $P$ (for example, if, for $P : A = 'USA'$, attribute $A$ can be removed from all LFDs $U \rightarrow B$ where $U$ contains $A$). Then, $X'_d$ can be minimized by the user to obtain a determinant of $A'$, which will also affect $X_f$.

(2) Projection queries: Let $T_r = \text{Project}_T(Y, f(Z) \rightarrow M)$.

- $A' \in Y$: 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 $F = F$, $X' = X_d$ and $X'_f = X_f$.

- For new measure attribute $A' = M$, since $A'$ is new, the forbidden set $X'_f$ is empty by default. Next, we show that $ZA$: Since $A' = f(Z)$, we know that $ZA$ and contains the determinant of $A$. However, it may not be minimal, and the user can minimize $Z$ to obtain a determinant $X'_f$ and add attributes to $X'_f$. (3) Pivot queries: Let $T_r = \text{Pivot}_T(A \mid Y)$.

- $A' \in S_D - Y$ and $\text{agg}_A(F, X)$ holds in $T$. By definition of pivot, $A'$ is a dimension attribute, and we obtain $X'_f = X_f - Y$ and $X' = (S_D - Y) - \{A'\} - X'_f$.

- $A'$ is a new attribute that holds pivoted values of $A$ and $\text{agg}_A(F, X)$ holds in $T$: If $X_d$ is a determinant of $A$ in $T$ and $X_d \not\subseteq Y$, then we can conclude that the remaining attributes $X'_d = X_d - Y$ contains a determinant of $A'$ that can be minimized by the user. Suppose that there are two tuples $t_1$ and $t_2$ in $T_r$ that 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 $t_1, A' = t'_1, A'$, $t_2, A' = t'_2, A'$, $t'_1, Y = t'_1, Y$, and $t'_2, (X_d - Y) = t'_2, (X_d - Y)$. We get $t'_1, X_d = t'_2, X_d$ and $t'_1, A' \neq t'_2, A'$, which is in contradiction with $X_d$ is a determinant of $A'$. Finally, all attributes that were forbidden for $A$ are also forbidden for $A'$, so $X'_f = X_f - Y$.

If $X_d \subseteq Y$, then a default identifier of $S_D - Y$ is computed and minimized by the designer of the table to obtain a determinant for $A'$. As before, all attributes that were forbidden for $A$ are also forbidden for $A'$, so $X'_f = X_f - Y$.

(4) Aggregate queries: Let $T_r = \text{Agg}_A(T(A) \mid Y)$:

- Every attribute $A' \in Y$ is a dimension attribute in $T$ and $T_r$. By definition, $A'$ is aggregate along any dimension attribute of $T_r$ except those in $X'_f$. All forbidden attributes in $T'$ that exist in $T_r$ are still forbidden, and $X'_f = X_f \cap Y$ preserves all forbidden attributes in $X_f$. We conclude that $\text{agg}_A(F, X)$ is correct and holds in $T_r$ where $X' = Y - \{A'\} - X'_f = X \cap Y$.

- $A' = F(A)$ is a new measure attribute. First, by assumption in the rule, $G$ is applicable to $F(A)$, so Item 1 of Definition 9 is satisfied. Second, $X'_f$ must retain the forbidden attributes of $X_f$ so $X'_f = X_f \cap Y$. Last, by the default rule, $X'_d$ is the fact identifier of $T_r$. If $X'_d$ is not minimal, then attributes that are unnecessary to determine $A'$ are removed from $X'_d$ and added to $X'_f$. Thus, a user action can be required.

**Binary queries.** We show that the rules of Table 13 are correct as follows:

(1) Merge queries:

- If $A'$ is a dimension attribute, by Definition 9, $X = S_D - \{A'\} - X_f$, and since $\{A'\} \cup X_f \subseteq S_D$ (all forbidden attributes are in $S_D$), then we can add all attributes of $S_D$ that are not in $S_D$ to $X': X' = X \cup (S_D - Y)$. By default, $X'_f = X_f$ and the user can add any new meaningless attribute from $S'_D$ to $X'_f$.

- If $A'$ is a measure attribute, then $A'$ is an attribute of table $T$ but not of table $T'$ (attribute names are pairwise distinct in $T_r$). We show that $X_d \mapsto A'$ holds in $T_r$ for each merge operation.

- If $T_r = T \Rightarrow_Y T'$ (left merge): By definition of $\Rightarrow_Y$ the projection on $S$ of $T_r$ is equal to $T$ and $X_d \mapsto A'$ still holds.

- If $T_r = T \Rightarrow_C Y T'$ (right merge): Suppose that $X_d \mapsto A'$ holds in $T$. After right-merge, $T_r$ might contain two tuples $t$ and $t'$ such that (1) $t.S \in T$, $t'.S \notin T$, (2) $t.X_d \equiv t'.X_d$, and (3) $t.A' \neq t'.A'$. Then we know that $t'.A' = \text{null}$, and since the functional dependency $X_d \mapsto A'$ only applies to the subset of $T_r$ where $A' \neq \text{null}$, $X_d \mapsto A'$ still holds in $T$.
• If $T_r = T \bowtie_{c_1} T'$ (full merge): The proof as a combination of the proofs for left merge and right merge.
• If $T_r = T \bowtie_{c_2} T'$ (strict merge): Any projection on $S$ of a tuple $T$ in $T_r$ is also a tuple in $T$. The projection on $S$ of table $T_r$ resulting from a left-merge or full-merge preserves all tuples of $T$ and $X_d$ is still a minimum set of dimension attributes in $T_r$ that determines $A'$. The same projection of the result of a right-merge or a strict-merge might eliminate some tuples of $T$ and $X_d$ might not be minimal anymore. In this case, the user should minimize $X_d$ to make it a determinant of $A'$. Then, all meaningless attributes in $X_f$ remain meaningless in $T_r$, so $X'_f = X_f$. Thus, $X' = (X_d \cup X_f) = X_f = X$. Finally, the user can always add new meaningless attributes from $S' - Y$ to $X'_f$.

(2) Set queries:
(a) If $A'$ is a dimension attribute: Let $X' = S_D - \{A\} - X_f$, where $S_D$ is the set of dimension attributes in $S$. Since $S_r = S$ for difference and union, and $X_f$ is defined for a given set of attributes (independently of a specific table), $X'_f = X_f$ and $X' = X$.
(b) If $A'$ is a measure attribute: Forbidden attributes remain the same, so $X'_f = X_f$. Then, by assumption, $X_dA'$ in table $T$. We analyze each case of query:

• Difference: By definition of analytic difference queries, $T_r \subseteq T$, thus $X_dA'$ also holds in $T_r = T - T'$. Thus, $X' = (X_d \cup X_f) = X_f = X$.
• Union: First, if $X_d = S_D$, by the definition of union, then we know that $X_dA$ holds in both tables $T$ and $T'$ and their union. Otherwise, $X_d \subset S_D$ holds, and the union may contain two tuples that have the same values on their $X_d$ attributes, and we must check if $X_dA$ still holds in $T_r = T \cup T'$. If this is not the case, then the user has two options: Resolve the conflicting tuples (e.g., eliminate one of the two tuples) to preserve the semantics of the operand tables, or add new attributes to $X_d$ to make it a determinant for $A'$. If this attribute was in $X_f$ before, then it should be removed from $X_f$.

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

Proposition 2 (Function Distributivity and Attribute Summarizability). Let $T(S)$ be an analytic table with dimension attributes $S_D \subseteq S$ and an aggregate attribute $\Lambda$ such that $\text{agg}_A(F_X)$ holds in $T$. If $F$ is distributive using function $G$ on attribute $\Lambda$ in table $T$ with partitioning attributes $Z \supseteq S_D - X$, then $\Lambda$ is summarizable with respect to grouping set $Z$ and function $F$ using function $G$.

Proof of Proposition 2. Suppose that $\text{agg}_A(F_X)$ holds in $T$ and $T_1 = \text{Agg}_{T_1}(F_{\Lambda} \mid Z_1)$ and $F$ is distributive using function $G$ on attribute $\Lambda$ of table $T$ with partitioning attributes $Z_1$. To prove that $\Lambda$ 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:

$$\text{Agg}_{T_1}(F_{\Lambda} \mid Z_2) = \text{Agg}_{T_1}(G(F_{\Lambda}) \mid Z_2).$$

(3)

First, it is obvious that both tables $T$ and $T_1$ contain the same $Z_2$ values, and therefore the result tables in Equation (3) contain the same tuples with distinct $Z_2$ values. We now show that for each pair of tuples $t \in \text{Agg}_{T_1}(F_{\Lambda} \mid Z_2)$ and $t' \in \text{Agg}_{T_1}(G(F_{\Lambda}) \mid Z_2)$ where $t.Z_2 = t'.Z_2$, we have $t.F(\Lambda) = t'.G(F(\Lambda)).$ Let $x = t.Z_2 = t'.Z_2$ and $T^x = \sigma_{Z_2=t.Z_2}(T)$ and $T^x_1 = \sigma_{Z_2=t.Z_1}(T_1)$ be the partitions of $T$ and $T_1$ on attributes $Z_2$ corresponding to the tuples used to compute $t.F(\Lambda)$. For each tuple $t_i' \in T^x_1$, there also exists a partition $T_i^{y_i} = \sigma_{Z_2=y_i}(T)$ of $T$ where $y_i = t_i'.Z_1$ and $t_i'.F(\Lambda) = F(\pi_{A}(T_i^{y_i}))$. All tuples $t_i'$ have the same $Z_2$ value $x = t.Z_2$ and are aggregated to tuple $t'$ whose value for attribute $t'.G(F(\Lambda)) = G(F(\pi_{A}(T_i^{y_i})), \ldots, F(\pi_{A}(T_i^{y_i}))).$ Since $F$ is distributive using function $G$ and $T^x = T_i^{y_i} \cup \cdots \cup T_i^{y_n}$, we obtain $G(F(\pi_{A}(T_i^{y_1})), \ldots, F(\pi_{A}(T_i^{y_n}))) = F(\pi_{A}(T_i^{y_1}) \cup \cdots \cup \pi_{A}(T_i^{y_n})) = F(\pi_{A}(T^x)).$ We conclude that $t.Z_2 = t'.Z_2$ and $t.F(\Lambda) = t'.G(F(\Lambda)).$ □
Proposition 3 (Summarizability of COUNT_DISTINCT using SUM). Let \( T(S) \) be an analytic table with a set of dimension attributes \( S_D \) and an aggregable attribute \( A \). Let \( T_1 = \text{Agg}_T(\text{COUNT\_DISTINCT}(A) \mid Z_1) \) be a valid aggregate query over \( T \). If \( Z_2 \subseteq Z_1 \) and the LFD \( Z_2 \cup \{A\} \mapsto Z_1 \) holds in \( T \), then the following equation is true:
\[
\text{Agg}_T(\text{SUM}(\text{COUNT\_DISTINCT}(A)) \mid Z_2) = \text{Agg}_T(\text{COUNT\_DISTINCT}(A) \mid Z_2).
\]
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 of Proposition 3. 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 \( Z_2 \subseteq Z_1 \) that have the same value for attribute \( t.A \) 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 \( Z_2 \cup \{A\} \mapsto Z_1 - Z_2 \) holds in \( T \), all tuples \( T \) in some partition \( T^x \subseteq T \) generated by attributes \( Z_2 \) with the same value for attribute \( t.A \) are assigned to the same partition \( T^y \subseteq T \) generated by attributes \( Z_1 \).

Let \( T^x \) be a partition of \( T \) that contains all tuples \( T \) such that \( t.Z_2 = x \). Since \( Z_2 \subseteq Z_1 \), \( T^x \) is the union of a set of partitions \( T^y_{11}, \ldots, T^y_{1n} \), \( n \geq 0 \) of \( T \) defined by attributes \( Z_1 \). Suppose that there exist two tuples \( t \in T^y_{1i} \) and \( t' \in T^y_{1j} \) where \( i \neq j \) and \( t.A = t'.A \). Then, we have \( t.Z_2 = t'.Z_2 = x \), \( t.A = t'.A \), and, since \( i \neq j \), \( t.Z_1 \neq t'.Z_1 \) (two different partitions generated by \( Z_1 \) contain the same values for \( Z_2 \) and \( A \)). This is in contradiction with \( Z_2 \cup \{A\} \mapsto Z_1 - Z_2 \). Then, if \( d_i \) is the number of distinct \( A \) values in some partition \( T^x_{i} \subseteq T \), then we can easily show that \( \sum_{i=0}^{n} d_i \) is the number of distinct \( A \) values in partition \( V \).

Proposition 4 (Propagating of Aggregable Properties with Summarizability Preservation). Let \( T(S) \) be an analytic table with dimension attributes \( S_D \subseteq S \) and let \( T_r = \text{Agg}_r(F(A) \mid Y) \) be the result of a valid aggregate query. Then the aggregable properties inferred by the rule of Table 1 for attribute \( A' = F(A) \) preserve the summarizability of \( A \) with respect to grouping set \( Y \) and are correct.

Proof of Proposition 4. By Definition 12, we have to show that for any subset \( Z_2 \subseteq Y \) such that \( Y \cap X' \subseteq Z_2 \) and \( A \) is summarizable in \( T \) with respect to \( Z_2 \) and function \( F \) using \( G \). We examine both cases of Proposition 4:

- By assumption, \( G \) can be applied on \( A' \) and \( F \) is distributive using \( G \). Thus, \( F \) is also distributive using function \( G \) on attribute \( A \) with partitioning attributes \( Y \). Then, by Proposition 2, \( A \) is summarizable with respect to \( Y \) and \( F \) using function \( G \), and Equation (1) in Definition 10 holds for any subset \( Z_2 \subseteq Y \). The rule is correct, since \( F \) is determined by Table 9, \( X'_f \) is determinant of \( A' \), and \( X'_f \) propagates all dimension attributes of \( X_f \cap Y \).
- \( F = \text{COUNT\_DISTINCT} \), \( K \) is a minimal subset of attributes of \( Y \) such that \( K \cup \{A\} \) determines all attributes of \( Y \) in \( T \) and \( X'_f = (X_f \cap Y) \cup K \). Thus, by the definition of \( X' \), we have \( X' \cap X_f = \emptyset \) and \( K \subseteq Y - X' \). By Proposition 3, it is sufficient to show that \( Z_2 \cup \{A\} \mapsto Y \) for all \( Z_2 \) where \( Y - X' \subseteq Z_2 \). Since \( K \cup \{A\} \mapsto Y \) and \( K \subseteq Y - X' \subseteq Z_2 \) we also have \( Z_2 \cup \{A\} \mapsto Y \). The rule is correct, since \( F \) produces a numerical value that can be summed, \( X'_f \) is determinant of \( A' \) and \( X'_f \) propagates all dimension attributes of \( X_f \cap Y \).

Proposition 5 (Queries Satisfying G-Summarizability). Let \( Q \) be a unary or binary analytic query with some input table \( T(S) \) returning a table \( T_1(S_1) \) and \( S_D \) be the dimension attributes in \( S \cap S_1 \). Let \( Z \) be a subset of \( S_D \), and \( A' \) be an attribute in \( S \cap S_1 \) such that \( \text{agg}_{A}(F, X) \) and \( \text{agg}_{A'}(F, X) \) hold in...
$T$ and $T_1$ respectively. Then, $A'$ is $G$-summarizable in $T$ with respect to query $Q$, grouping set $Z$, and function $F$ in the following cases:

Unary queries:

1. $Q = \text{Filter}_T(P \mid Y), Y \subseteq S_D, A' \in S$ and $Z = (S_D - X_1) \cup Y$.
2. $Q = \text{Project}_T(Y, f(Z') \rightarrow M), A' \in Y$ and $Z = S_D - X_1$.
3. $Q = \text{Agg}_T(G(A) \mid Y), A' \in Y, Z = Y - X_1$ and $F \in \{\text{MIN, MAX, COUNT\_DISTINCT}\}$.
4. $Q = \text{Pivot}_T(A \mid Y), A' \in S - Y - \{A\}, Z = S_D - X_1 - Y - \{A\}$ and $F \in \{\text{MIN, MAX, COUNT\_DISTINCT}\}$.

Merge queries: First, for all merge queries, we apply the rule restricting the aggregation function to duplicate insensitive function when $Y$ does not determine the attributes of the second argument table $T' : Y \not\leftrightarrow S'$ then $F \in \{\text{MIN, MAX, COUNT\_DISTINCT}\}$. Then, let $Y^{top} \subseteq Y$ denote the subset of highest attributes in the set of join attributes $Y$. We can define the following rules:

1. $Q = T \bowtie_Y T'(S')$, $A' \in S, Z = S_D - X_1$
2. $Q = T \bowtie_Y T'(S')$ or $Q = T \bowtie_Y T'(S')$:
   
   (a) If $A' \in Y$ and for all non-empty partitions $T''_y = \sigma_{Y^{top}=y}(T')$ of $T'$, then the corresponding partition $T^y = \sigma_{Y^{top}=y}(T)$ of $T$ is empty or $\pi_Y(T'^y) = \pi_Y(T^y)$, and then $Z = (S_D - X_1) \cup Y^{top}$.
   
   (b) If $A' \in S - Y$ and for all non-empty partitions $T''_y = \sigma_{Y=yt}(T')$ of $T'$ and corresponding partitions $T^y = \sigma_{Y=yt}(T)$ of $T$, $\pi_Y(T'^y) \subseteq \pi_Y(T^y)$, then $Z = (S_D - X_1) \cup Y^{top}$.
   
   (c) Otherwise, $Z = (S_D - X_1) \cup Y$.

Set queries:

1. $Q = T \cup T', A' \in S$ and if $\pi_{Y^{top}}(T) \cap \pi_{Y^{top}}(T') = \emptyset$, then $Z = S_D - X_1 \cup Y^{top}$, otherwise $Z = Y$.

2. $Q = T - T', A' \in S$ and if all partitions $\sigma_{Y^{top}=y}(T)$ are equal to or disjoint with $\sigma_{Y_{top}=y}(T')$, then $Z = (S_D - X_1) \cup Y^{top}$ else $Z = (S_D - X_1) \cup Y$.

Proof of Proposition 5. We already know that both aggregation queries $Q'(T)$ and $Q'(T_1)$ are valid, and we only have to show that for any two tuples $t_1 \in Q'(T)$ and $t_2 \in Q'(T_1)$, we have $t_1, t_2 \equiv_{t_2, Z} t_2, F(A) \equiv_{t_2, F(A)}$.

$G$-summarizability condition: We shall use symbol $\Pi$ to denote the duplicate preserving projection and $\pi$ to denote duplicate eliminating projection. Let $T$ and $T_1$ be two tables with a common set of dimension attributes $Z$ and some attribute $A'$. Then, the $G$-summarizability condition holds for $T, T_1, Z, A'$ if for any partition $T^x = \sigma_{Z=xt}(T)$ of $T$ and $T_1^x = \sigma_{Z=xt}(T_1)$ of $T_1$, we have (1) $\pi_{Z,A'}(T_1^x) = \pi_{Z,A'}(T^x)$ (both partitions are equal modulo duplicates), (2) $T^x$ is empty, or (3) $T_1^x$ is empty. If condition (1) also holds using the duplicate preserving projection, i.e., $\Pi_{Z,A'}(T_1^x) = \Pi_{Z,A'}(T^x)$, then we say that it strictly holds for $T, T_1, Z, A'$.

We can show that the $G$-summarizability condition is sufficient to prove that $A'$ is $G$-

summarizable in $T$ with respect to some query $Q$, grouping set $Z$ and $F$ as follows:

- For case (1), if $Z \not\leftrightarrow A'$ in $T$ and $T_1$, then $\Pi_{Z,A'}(T_1^x) = \Pi_{Z,A'}(T^x)$ (strict $G$-summarizability condition where 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 $T_1$ for all aggregation functions $F$. Otherwise, if $Z \not\leftrightarrow A'$, then we can only guarantee $\pi_{Z,A'}(T_1^x) = \pi_{Z,A'}(T^x)$, and $F$ must be restricted to aggregation functions that are not sensitive to duplicates ($F \in \{\text{COUNT\_DISTINCT, MIN, MAX}\}$).

- 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 $G$-summarizability.

ACM Journal of Data and Information Quality, Vol. 15, No. 2, Article 12. Publication date: June 2023.
For certain merge operations and union, the G-summarizability condition might not be verified by only considering the aggregable properties and the query structure. For example, union might add new tuples to Table T, such that for some partitions we obtain θ ∈ π_{Z,A′}(T^x) ⊆ π_{Z,A′}(T^x_1) (the condition does not hold). However, if we know for example that θ ∈ π_{U,A′}(T^y) = π_{U,A′}(T^y_1) for some U ⊆ Z, then we can conclude that the condition holds for U.

**Unary queries:**

1. Analytic filter $T_1 = \text{Filter}_T(P \mid Y)$: By the condition $Y \subseteq S_D$, all attributes in $Y$ are dimension attributes. By $Z = (S_D - X_1) \cup Y$, we have $Y \subseteq Z$. Then for any non-empty partition $T^x = \sigma_{Z=x}(T)$ of $T$ we can show that if $P$ is true for some tuple in $T^x$, then it is true for all tuples in $T^x$: $Z$ contains all filtering predicate $P$, and we can show that for all $Z = x$ either $Z = x \Rightarrow P(Y)$ or $Z = x \Rightarrow \neg P(Y)$. Therefore, the corresponding partition $T^x_1 = \sigma_{Z=x}(T)$.

2. Analytic projection $T_1 = \text{Project}_T(Y, f(Z) \rightarrow M)$: By definition of analytic projection, for any subset $X \subseteq Y$, we have $\Pi_X(T_1) = \Pi_X(T)$. Since $Z = S_D - X_1 \subseteq S_D \subseteq Y$ and $A' \subseteq X$, we have $\Pi_{Z,A'}(T_1) = \Pi_{Z,A'}(T)$.

3. Analytic aggregate $T_1 = \text{Agg}_T(G(A) \mid Y)$: By definition of aggregate queries, for any $X \subseteq Y$ and partition $T^x$, we have $\mu_X(T^x) = \mu_{X}(T^x)$ (duplicate eliminating projection). Since $Z \cup A' \subseteq Y$, we then have $\pi_{Z,A'}(T^x_1) = \pi_{Z,A'}(T^x)$. However, $T^x_1$ generally contains duplicate tuples that are merged into a single tuple in $T_1$. Therefore, $F$ must be restricted to functions that are not duplicate sensitive ($F \in \{\text{COUNT_DISTINCT}, \text{MIN}, \text{MAX}\}$).

4. Analytic pivot $T_1 = \text{Pivot}_T(A' \mid Y)$: We apply similar arguments as for aggregate queries on the remaining attributes $Y' = S - Y - \{A\}$ in $T_1$. By definition of pivot queries, for any subset $X \subseteq Y'$ and partition $T^x$, we have $\pi_X(T^x) = \mu_X(T^x)$ (duplicate eliminating projection) and $\pi_{Z,A'}(T^x_1) = \pi_{Z,A'}(T^x)$ in particular for $Z \subseteq Y'$ and $A' \subseteq Y'$. However, in the general case, $T^x_1$ 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 ($F \in \{\text{COUNT_DISTINCT}, \text{MIN}, \text{MAX}\}$).

**Merge queries:**

1. Left-merge query $T(S) \bowtie_T T'(S')$: By definition, $\pi_S(T) = \pi_S(T_1)$ (each tuple of $T$ produces one or more tuples in $T_1$). Thus, for any $Z \cup A' \subseteq S_D$ we also have $\pi_{Z,A'}(T) = \pi_{Z,A'}(T_1)$ (the G-summarizability condition obviously holds). By definition, a tuple in $T$ can only appear twice in $\Pi_Z(T_1)$ if $Y \not\rightarrow S'$. If that is the case, then $F$ must be restricted to functions that are not sensitive to duplicates: $F \in \{\text{COUNT_DISTINCT}, \text{MIN}, \text{MAX}\}$.

2. Right-merge query $T(S) \bowtie_T T'(S')$ or full-merge query $Q = T(S) \bowtie_T T'(S')$: Let $Y^{op} \subseteq S_D$ denote the set of highest attributes in the set of dimension attributes $S_D \subseteq S$ of $T$ and $T'$.

   a) $A' \subseteq S$ and for all non-empty partitions $T'^y = \sigma_{V^{op}=y}(T')$ of $T'$, the corresponding partition $T^y = \sigma_{V^{op}=y}(T)$ is either empty or $\pi_Y(T^y)$ is equal to $\pi_Y(\sigma_{V^{op}=y}(T'))$: From the second assumption and the definition of right-outer join, it directly follows that for any non-empty partition $T^y_1 = \sigma_{V^{op}=y}(T_1)$ of $T_1$, the corresponding partition $T^y_1 = T^y_1$ is empty ($T^y \not\rightarrow S$) or $\pi_S(T^y) = \pi_S(T^y_1)$. Then, for all $X \supseteq Y^{op}$, all non-empty partitions $\pi_X(\sigma_{X=x}(T_1))$ are equal to $\pi_X(\sigma_{X=x}(T))$ and since, by definition of $Z, Z \cup A' \supseteq Y^{op}$, the G-summarizability condition holds on $T$ and $T_1$ for $Z$ and $A'$.

   b) $A' \subseteq S - Y$ and for all non-empty partitions, $T'^y = \sigma_{V^{op}=y}(T')$, and corresponding partitions $T^y = \sigma_{V^{op}=y}(T), \pi_Y(T^y)$ is a subset of $\pi_Y(\sigma_{V^{op}=y}(T'))$. From this assumption and the definition of right-outer join, it directly follows that for any non-empty partition
\[ T^y_1 = \sigma_{\text{top}=y}(T_1) \]
and corresponding partition \( T^y \), \( \pi_{S \setminus Y}(T^y_1) - \pi_{S \setminus Y}(T^y) \) only contains null values. Then, for all \( X \subseteq S_0 \) and \( A' \in S - Y \), all non-empty partitions \( \pi_x(A'(\sigma_{x=x}(T))) \) are equal to \( \pi_x(\sigma_{x=x}(A'\neq null)(T_1)) \) and since, by definition of \( Z, Z \cup \{A'\} \subseteq Y^{\text{top}} \), the G-summarizability condition holds on \( T \) and \( T_1 \) for \( Z \) and \( A' \).

(c) Otherwise: From the definition of right-merge and full-merge, it directly follows that for any non-empty partition \( T^y_1 = \sigma_{\text{top}=y}(T_1) \), the corresponding partition \( T^y \) is either empty or equal to \( \pi_S(T^y_1) \). Then, for all \( X \supseteq Y \), all non-empty partitions \( \pi_x(\sigma_{x=x}(T)) \) are equal to \( \pi_x(\sigma_{x=x}(T_1)) \) and since, by definition of \( Z, Z \cup \{A'\} \supseteq Y \), the previous condition also holds for \( X = Z \cup \{A'\} \). Therefore, the G-summarizability condition holds for \( T \) and \( T_1 \) for \( Z \) and \( A' \).

(3) \( Q = T(S) \bowtie Y T'(S') \):

(a) “if” condition: We assume that for all non-empty partitions \( T^{y'} = \sigma_{\text{top}=y}(T') \), the corresponding partition \( T^y = \sigma_{\text{top}=y}(T) \) is contained in \( \pi_Y(T^{y'}) \). From this assumption and the definition of inner join, it directly follows that any non-empty partition \( T^y_1 = \sigma_{\text{top}=y}(T_1) \) in the result is equal to the corresponding partition \( T^{y'}: \pi_S(T^y_1) = \pi_T(T^{y'}_1) \). Then, for all \( X \supseteq Y^{\text{top}} \), all non-empty partitions \( \pi_X(\sigma_{x=x}(T)) \) are equal to \( \pi_X(\sigma_{x=x}(T_1)) \) and since, by definition of \( Z, Z \cup \{A'\} \supseteq Y^{\text{top}} \), the previous condition also holds for \( X = Z \cup \{A'\} \). Therefore, the G-summarizability condition holds on \( T \) and \( T_1 \) for grouping set \( Z \) and attribute \( A' \).

(b) “else” condition: From the definition of inner join, it directly follows that for any non-empty partition \( T^{y'}_1 = \sigma_{\text{top}=y}(T_1) \), the corresponding partition \( T^y \) is equal to \( \pi_T(T^{y'}_1) \). Then, for all \( X \supseteq Y \), all non-empty partitions \( \pi_X(\sigma_{x=x}(T_1)) \) are equal to \( \pi_X(\sigma_{x=x}(T)) \) and since, by definition of \( Z, Z \cup \{A'\} \supseteq Y \), the previous condition also holds for \( X = Z \cup \{A'\} \). Therefore, the G-summarizability condition holds on \( T \) and \( T_1 \) for grouping set \( Z \) and \( A' \).

\[
\text{Set queries:}
\]

1. \( Q = T(S) \cup T'(S) \): Union might add new tuples to \( T \) such that \( \pi_X(\sigma_{x=x}(T)) \subseteq \pi_X(\sigma_{x=x}(T_1)) \) and the G-summarizability condition does not hold. By the assumption \( \pi_{Y^{\text{top}}}(T) \cap \pi_{Y^{\text{top}}}(T') = \emptyset \) and the definition of union, it follows that for any non-empty partition \( T^y_1 = \sigma_{\text{top}=y}(T_1) \) in the result, the corresponding partition \( T^y \) is either empty or \( T^y \) is equal to \( T^y_1 \). Then, since \( Y \supseteq Y^{\text{top}} \), all non-empty partitions \( \pi_X(\sigma_{x=x}(T)) \) are equal to \( \pi_X(\sigma_{x=x}(T_1)) \) and since \( Z = S_D - X_1 \cup Y^{\text{top}} \), we have \( Z \supseteq Y^{\text{top}} \) and the G-summarizability condition also holds for \( Z \). Finally, the aggregable property condition also holds: \( S_r \cap S_D - X_1 = S_D - X_1 \subseteq Z = S_D - X_1 \cup Y^{\text{top}} \).

2. \( Q = T(S) - T'(S) \): By assumption, all partitions \( \sigma_{\text{top}=y}(T) \) are equal to or disjoint with \( \sigma_{\text{top}=y}(T') \). Then, by the definition of set-difference, it follows that for any non-empty partition \( T^y_1 = \sigma_{\text{top}=y}(T_1) \) in the result, the corresponding partition \( T^y \) is either empty or \( T^y \) is equal to \( T^y_1 \). Then, for all \( X \supseteq Y^{\text{top}} \), all non-empty partitions \( \pi_X(\sigma_{x=x}(T)) \) are equal to \( \pi_X(\sigma_{x=x}(T_1)) \) and since, by definition of \( Z, Z \supseteq Y^{\text{top}} \), the previous condition also holds for \( X = Z \cup \{A'\} \). Therefore, the G-summarizability condition holds on \( T \) and \( T_1 \) for \( Z \) and \( A' \). Finally, the aggregable property condition also holds: \( S_r \cap S_D - X_1 = S_D - X_1 \subseteq Z = S_D - X_1 \cup Y^{\text{top}} \).

**Proposition 6 (Aggregable Properties with G-summarizability).** Let \( Q \) be an analytic query with some input table \( T(S) \) returning a table \( T_r(S') \), and \( S_D \) be all the dimension attributes of \( S \cap S' \). Then, for all queries \( Q \) satisfying the conditions of Tables 16, 17, and 18, the aggregable property \( \text{agg}_{\text{A}}(F, X') \) holds in \( T_r \) and is such that for all \( Z \) where \( S_D - X' \subseteq Z \), \( A' \) is G-summarizable in \( T \) with respect to query \( Q \), grouping set \( Z \), and function \( F \).

**Proof of Proposition 6.** The proof for unary queries mainly consists in showing that the rules of Table 16 can be obtained from the rules of Table 12 by adding the G-summarizability constraints of Proposition 5. This extension consists in adding for each operation the attributes that must be in
the grouping set $Z$ (as defined by Proposition 5) to the forbidden attributes $Z_f$ of the corresponding rule in Table 12.

For example, for filter queries, since $Z = (S_D - X') \cup Y$ in Proposition 5 and $X'_f = X_f$ in Table 12, we obtain a new definition for $X'_f = X_f \cup Y$ by adding $Y$ to the new set of forbidden attributes. For aggregation queries, $Z = Y - X'$ in Proposition 5 and $X'_f = X_f \cap Y$ in Table 12, so $X'_f = X_f \cap Y$ does not change in Table 16.

Likewise, for binary queries, propagation rules of Tables 17 and 18 are obtained by extending the rules of Table 13. Rules are extended by first defining the new $X'_f$ as where $X'_f$ is replaced by its definition in Table 13 and, second, adding the partition containment conditions defined for the partition $T^y$ and $T'^y$.

For example, for right-merge and full-merge queries, $X'_f$ is as in Table 13. Then, if the first condition of the rules (a) or (b) holds (for $A' \in Y$ and $A' \in S_D - Y$), then we obtain $X'_f = X'_f \cup Y^{top}$. Otherwise, $X'_f = X'_f \cup Y$. For union, the rules of Table 13 are extended by checking if $T$ and $T'$ have no partitions in common for partitioning attributes $Y^{top}$ to determine the value of a determinant $X_d$. Then they add $Y^{top}$ to $X'_f$. If the partition containment condition is not satisfied, then $X_f = Y$, i.e., it is forbidden to aggregate along any dimension attribute.

\[ \square \]

REFERENCES

[1] Foto Afrati and Rada Chirkova. 2019. Answering queries using views. Synth. Lect. Data Manage. 14, 3 (2019), 1–275.
[2] Foto Afrati and Rada Chirkova. 2011. Selecting and using views to compute aggregate queries. J. Comput. Syst. Sci. 77, 5 (2011), 1079–1107.
[3] Sina Ariyan and Leopoldo Bertossi. 2013. A multidimensional data model with subcategories for flexibly capturing summarizability. In Proceedings of the 25th International Conference on Scientific and Statistical Database Management (SSDBM’13). Association for Computing Machinery, New York, NY. https://doi.org/10.1145/2484838.2484857
[4] Azure 2020. Azure Blob storage: Massively scalable and secure object storage for cloud-native workloads, archives, data lakes, high-performance computing, and machine learning Software. (2020). https://azure.microsoft.com/enus/services/storage/blobs/.

[5] Antonio Badia and Daniel Lemire. 2014. Functional dependencies with null markers. Comput. J. 58, 5 (2014), 1160–1168.
[6] Robert Brunel, Jan Finis, Gerald Franz, Norman May, Alfons Kemper, Thomas Neumann, and Franz Faerber. 2015. Supporting hierarchical data in SAP HANA. In Proceedings of the 31st International Conference on Data Engineering (ICDE’15). IEEE, 1280–1291.
[7] Robert Brunel, Norman May, and Alfons Kemper. 2016. Index-assisted hierarchical computations in main-memory RDBMS. Proc. VLDB Endow. 9, 12 (2016), 1065–1076. https://doi.org/10.14778/2994509.2994524
[8] Douglas Burdick, Prasad M. Deshpande, T. S. Jayram, Raghu Ramakrishnan, and Shivakumar Vaithyanathan. 2007. OLAP over uncertain and imprecise data. VLDB J. 16, 1 (2007), 123–144. https://doi.org/10.1007/s00778-006-0333-y
[9] Mónica Caniupán, Loreto Bravo, and Carlos A. Hurtado. 2012. Repairing inconsistent dimensions in data warehouses. Data Knowl. Eng. 79–80 (September 2012), 17–39. https://doi.org/10.1016/j.datak.2012.04.002
[10] Surajit Chaudhuri and Kyuseok Shim. 1994. Including group-by in query optimization. In Proceedings of 20th International Conference on Very Large Data Bases (VLDB’94), Jorge B. Bocca, Matthias Jarke, and Carlo Zaniolo (Eds.). Morgan Kaufmann, Santiago de Chile, Chile, 354–366. http://www.vldb.org/conf/1994/F354.PDF.
[11] Sara Cohen, Werner Nutt, and Alexander Serebrenik. 1999. Rewriting aggregate queries using views. In Proceedings of the 18th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, Victor Vianu and Christos H. Papadimitriou (Eds.). ACM Press, New York, NY and Philadelphia, 155–166. https://doi.org/10.1145/303976.303992
[12] Marco Console, Paolo Guagliardo, and Leonid Libkin. 2022. Fragments of bag relational algebra: Expressiveness and certain answers. Inf. Syst. 105 (2022), 101604. https://doi.org/10.1016/j.is.2020.101604
[13] Datadog. 2020. Datadog: Cloud Monitoring as a Web Service Software Datadog. Retrieved from https://www.datadoghq.com/.
[14] Su Feng, Boris Glavic, Aaron Huber, and Oliver A. Kennedy. 2021. Efficient uncertainty tracking for complex queries with attribute-level bounds. In Proceedings of the 2021 International Conference on Management of Data. Association for Computing Machinery, New York, NY, 528–540. https://doi.org/10.1145/3448016.3452791
12:40 E. Simon et al.

[15] César Galindo-Legaria and Milind Joshi. 2001. Orthogonal optimization of subqueries and aggregation. ACM SIGMOD Rec. 30, 2 (2001), 571–581.

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

[17] Jonathan Goldstein and Per-Ake Larson. 2001. Optimizing queries using materialized views: A practical, scalable solution. ACM SIGMOD Rec. 30, 2 (2001), 331–342.

[18] Paolo Guagliardo and Leonid Libkin. 2017. A formal semantics of SQL queries, its validation, and applications. Proc. VLDB Endow. 11, 1 (September 2017), 27–39. https://doi.org/10.14778/3151113.3151116

[19] F. Binti Hamzah, C. Lau, Hafeez Nazri, Dominic Vincent Litgot, Guanhua Lee, Cheng Liang Tan, MKBM Shaib, Umri 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). https://www.who.int/publications/journals/bulletin/covid-19

[20] John Horner and Il-Yeol Song. 2005. A taxonomy of inaccurate summaries and their management in OLAP systems. In Proceedings of the 24th International Conference on Conceptual Modeling (ER’05), Lois M. L. Delcambre, Christian Kop, Heinrich C. Mayr, John Mylopoulos, and Oscar Pastor (Eds.). Lecture Notes in Computer Science, Vol. 3716. Springer, 433–448. https://doi.org/10.1007/11568322_28

[21] John Horner, Il-Yeol Song, and Peter P. Chen. 2004. An analysis of additivity in OLAP systems. In ACM 7th International Workshop on Data Warehousing and OLAP (DOLAP’04), Il-Yeol Song and Karen C. Davis (Eds.). ACM, New York, NY, 83–91. https://doi.org/10.1145/1031763.1031779

[22] Carlos A. Hurtado, Claudio Gutierrez, and Alberto O. Mendelzon. 2005. Capturing summarizability with integrity constraints in OLAP. ACM Trans. Datab. Syst. 30, 3 (2005), 854–886.

[23] Christian S. Jensen, Torben Bach Pedersen, and Christian Thomsen. 2010. Multidimensional databases and data warehousing. Synth. Lect. Data Manage. 2, 1 (2010), 1–111.

[24] Paulo Jesus, Carlos Baquero, and Paulo Sérgio Almeida. 2015. A survey of distributed data aggregation algorithms. IEEE Commun. Surv. Tutor. 17, 1 (2015), 381–404. https://doi.org/10.1109/COMST.2014.2354398

[25] Kibana. 2020. Kibana: Your Window into the Elastic Stack Software Elastic. Retrieved from https://www.elastic.co/kibana

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

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

[28] Wolfgang Lehner, Jens Albrecht, and Hartmut Wedekind. 1998. Normal forms for multidimensional databases. In Proceedings of the 10th International Conference on Scientific and Statistical Database Management. IEEE, Capri, 63–72.

[29] Hans-Joachim Lenz and Arie Shoshani. 1996. Summarizability in OLAP and statistical data bases. In Proceedings of the 9th International Conference on Scientific and Statistical Database Management, Yannis E. Ioannidis and David Marshall Hansen (Eds.). IEEE Computer Society, Olympia, WA, 132–143. https://doi.org/10.1109/SSDM.1997.621175

[30] Hans-Joachim Lenz and Bernhard Thalheim. 2001. OLAP databases and aggregation functions. In Proceedings of the 13th International Conference on Scientific and Statistical Database Management. IEEE Computer Society, Los Alamitos, CA, 91–100. https://doi.org/10.1109/SSDM.2001.938542

[31] Hans-Joachim Lenz and Bernhard Thalheim. 2009. A formal framework of aggregation for the OLAP-OLTP model. J. Univ. Comput. Sci. 15, 1 (2009), 273–303. https://doi.org/10.3217/jucs-015-01-0273

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

[33] Jose-Noéborio Mazón, Jens Lechtenbörger, and Juan Trujillo. 2009. A survey on summarizability issues in multidimensional modeling. Data Knowl. Eng. 68, 12 (2009), 1452–1469.

[34] Tapio Niemi, Marko Niinimäki, Peter Thanisch, and Jyrki Nummenmaa. 2014. Detecting summarizability in OLAP. Data Knowl. Eng. 89 (2014), 1–20.

[35] olap_cube. 2020. Olap Cube. https://en.wikipedia.org/wiki/OLAP_cube

[36] paxata. 2019. Paxata—Self-Service Data Preparation for Data Analytics DataRobot. Retrieved from https://www.paxata.com/

[37] Judea Pearl, Madelyn Gleamour, and Nicholas Jewell. 2016. Causal Inferences in Statistics. John Wiley & Sons, Chichester.

[38] Torben Bach Pedersen, Christian S. Jensen, and Curtis E. Dyreson. 1999. Extending practical pre-aggregation in online analytical processing. In Proceedings of the 25th International Conference on Very Large Data Bases (VLDB’99). Morgan Kaufmann, San Francisco, CA, 663–674.

[39] Torben Bach Pedersen, Christian S. Jensen, and Curtis E. Dyreson. 1999. Supporting imprecision in multidimensional databases using granularities. In Proceedings of the 11th International Conference on Scientific and Statistical Database Management.
Controlling the Correctness of Analytic Queries

12:41

Management. Proceedings, Z. Meral Özsoyoglu, Gutlekın Özsoyoglu, and Wen-Chi Hou (Eds.). IEEE Computer Society, Los Alamitos, CA, 90–101. https://doi.org/10.1109/SSDM.1999.787624

[40] Torben Bach Pedersen, Christian S. Jensen, and Curtis E. Dyreson. 2001. A foundation for capturing and querying complex multidimensional data. Inf. Syst. 26, 5 (2001), 383–423. https://doi.org/10.1016/S0306-4379(01)00023-0

[41] powerbi. 2019. Power BI | Interactive Data Visualization BI Tools. https://powerbi.microsoft.com/en-us/.

[42] Qlik. 2019. Data Analytics for Modern Business Intelligence—Qlik Qlik. Retrieved from https://www.qlik.com/us.

[43] Liu Rutian. 2020. Semantic Services for Assisting Users to Augment Data in the Context of Analytic Data Sources. Ph.D. Dissertation. Sorbonne Université.

[44] s3. 2020. Amazon S3 Object storage built to store and retrieve any amount of data from anywhere Software Amazon. Retrieved from https://aws.amazon.com/s3.

[45] sapcloud. 2020. SAP Analytics Cloud SAP. Retrieved from https://www.sapanalytics.cloud/.

[46] Eric Simon, Bernd Amann, Rutian Liu, and Stéphane Gançarski. 2021. Controlling the correctness of aggregation operations during sessions of interactive analytic queries test. https://doi.org/10.48550/ARXIV.2111.13927

[47] Splunk. 2019. Splunk: The Data-to-Everything Platform Software Splunk. Retrieved from https://www.splunk.com/.

[48] Divesh Srivastava, Shaul Dar, H. V. Jagadish, and Alon Y. Levy. 1996. Answering queries with aggregation using views. In Proceedings of 22th International Conference on Very Large Data Bases (VLDB’96), T. M. Vijayaraman, Alejandro P. Buchmann, C. Mohan, and Nandlal L. Sarda (Eds.). Morgan Kaufmann, Mumbai (Bombay), 318–329. http://www.vldb.org/conf/1996/P318.PDF.

[49] Stanley Smith Stevens. 1946. On the theory of scales of measurement. Science 103, 2684 (1946), 677–680.

[50] Tableau. 2019. Tableau: Business Intelligence and Analytics Software Tableau. Retrieved from https://www.tableau.com/.

[51] Peter Thanisch, Tapio Niemi, Jyrki Nummenmaa, and Marko Niinimäki. 2019. Detecting measurement issues in SQL arithmetic expressions and aggregations. Data Knowl. Eng. 122 (2019), 116–129. https://doi.org/10.1016/j.datak.2019.06.001

[52] Dimitri Theodoratos and Timos K. Sellis. 2000. Answering multidimensional queries on cubes using other cubes. In Proceedings of the 12th International Conference on Scientific and Statistical Database Management, Oliver Günther and Hans-Joachim Lenz (Eds.). IEEE Computer Society, Los Alamitos, CA, 109–123. https://doi.org/10.1109/SSDM.2000.869782

[53] Riccardo Torlone. 2008. Two approaches to the integration of heterogeneous data warehouses. Distrib. Parallel Datab. 23, 1 (2008), 69–97.

[54] Trifacta. 2019. Data Wrangling Tools & Software—Trifacta Trifacta. Retrieved from https://www.trifacta.com/.

[55] Clifford H. Wagner. 1982. Simpson’s paradox in real life. Am. Statist. 36, 1 (1982), 46–48.

[56] Weipeng P. Yan and Per-Åke Larsson. 1994. Performing group-by before join. In Proceedings of the 10th International Conference on Data Engineering. IEEE Computer Society, Houston, TX, 89–100. https://doi.org/10.1109/ICDE.1994.283001

[57] Weipeng P. Yan and Per-Åke Larsson. 1995. Eager aggregation and lazy aggregation. In Proceedings of 21th International Conference on Very Large Data Bases (VLDB’95), Umeshwar Dayal, Peter M. D. Gray, and Shojiro Nishio (Eds.). Morgan Kaufmann, Zurich, 345–357.

[58] Alice Zheng and Amanda Casari. 2018. Feature Engineering for Machine Learning: Principles and Techniques for Data Scientists. O’Reilly Media, Inc., Sebastopol, CA.

Received 10 December 2021; revised 20 June 2022; accepted 16 September 2022