US 20120130940A1

(19) **United States**

(12) **Patent Application Publication**

(10) **Pub. No.: US 2012/0130940 A1**

(43) **Pub. Date: May 24, 2012**

Gattani et al.

(54) **REAL-TIME ANALYTICS OF STREAMING DATA**

(75) Inventors: **Abhishek Gattani**, Sunnyvale, CA (US); **Anand Rajaraman**, Palo Alto, CA (US)

(73) Assignee: **WAL-MART STORES, INC.**, Bentonville, AR (US)

(21) Appl. No.: **13/300,523**

(22) Filed: **Nov. 18, 2011**

Related U.S. Application Data

(60) Provisional application No. 61/415,279, filed on Nov. 18, 2010, provisional application No. 61/415,282, filed on Nov. 18, 2010.

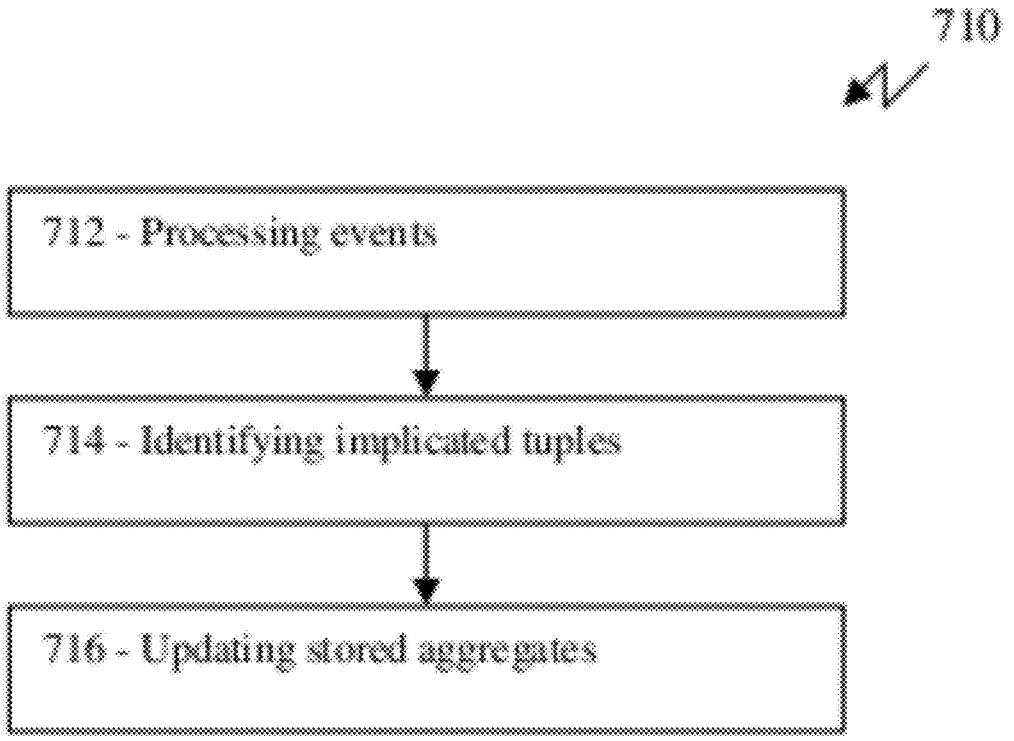
Publication Classification

(51) **Int. Cl.**
G06F 17/30 (2006.01)

(52) **U.S. Cl.** **707/600**; 707/791; 707/E17.056

(57) **ABSTRACT**

Storage media, systems and methods are disclosed herein for analyzing data streams in real time. More particularly, storage media, systems and methods are presented for processing data streams to calculate results for prospective queries. The results may be advantageously computed prior to the formulation of the specific query, for example, based on a pre-established framework of potential query parameters. More particularly, a universe of potential queries may be extrapolated from the pre-established framework of potential query parameters. Results for each of the potential queries may them be tracked in real time. For example, results for each of the potential queries may be continuously updated based on real-time processing of events in a data stream.



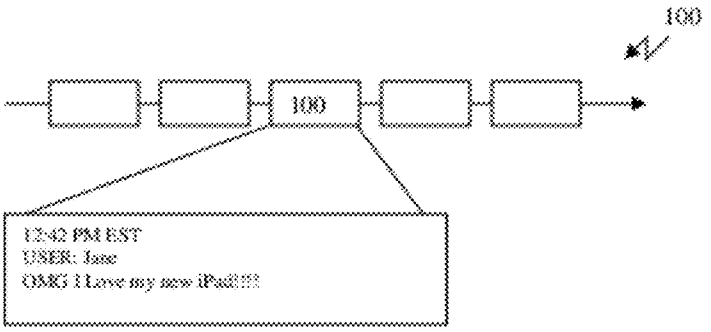


Figure 1

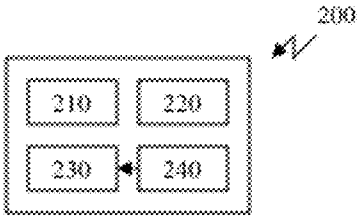


Figure 2

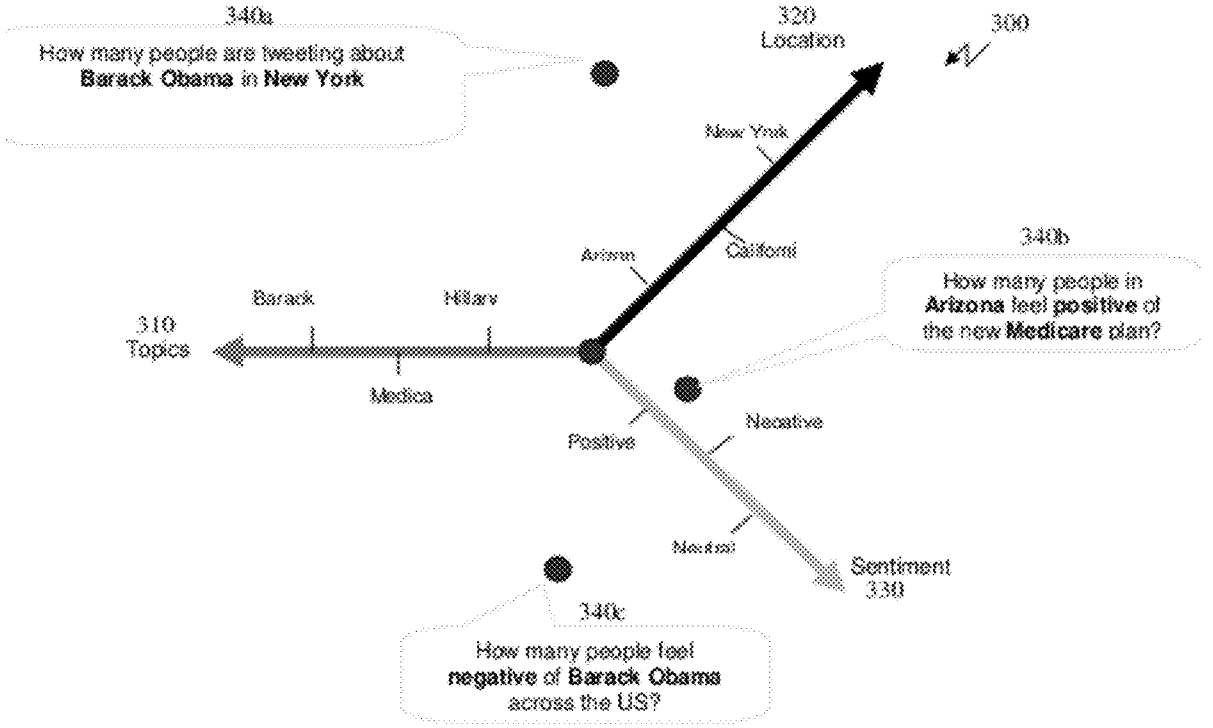


Figure 3

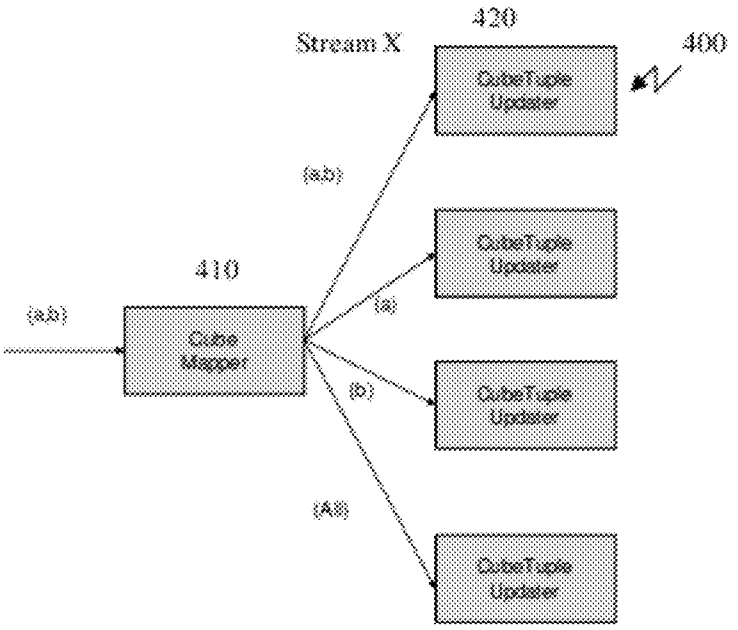


Figure 4

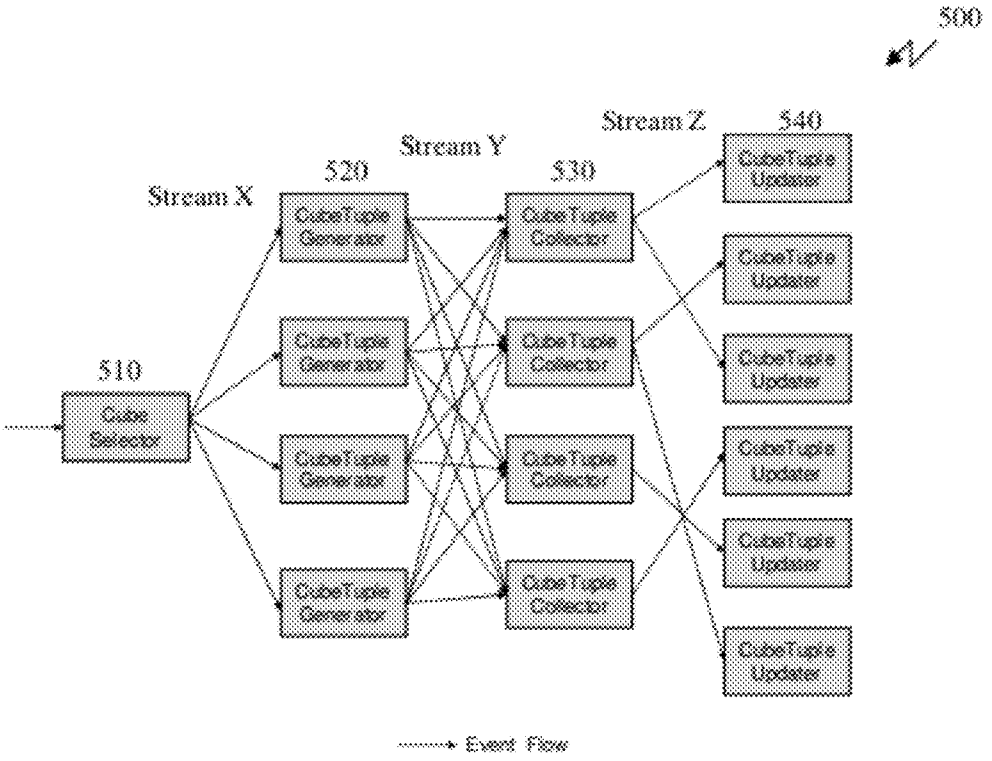


Figure 5

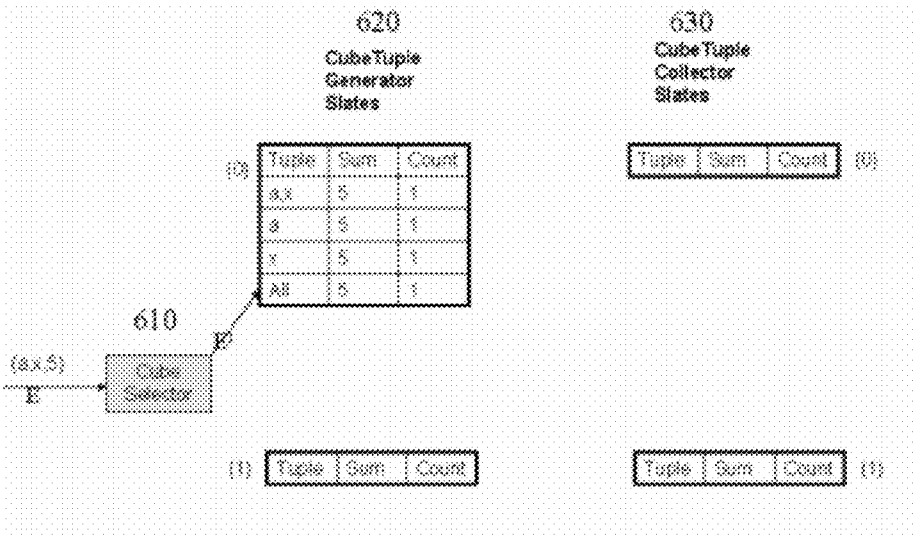


Figure 6a

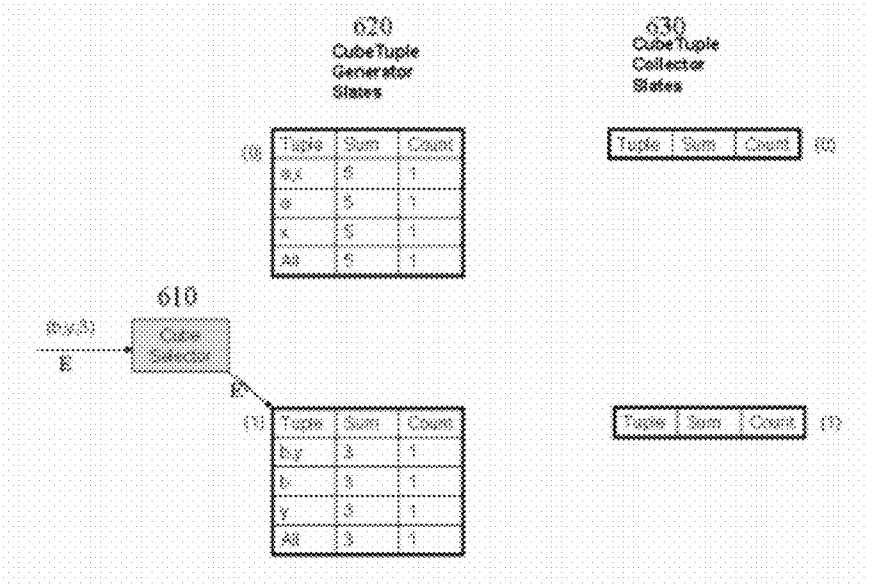


Figure 6b

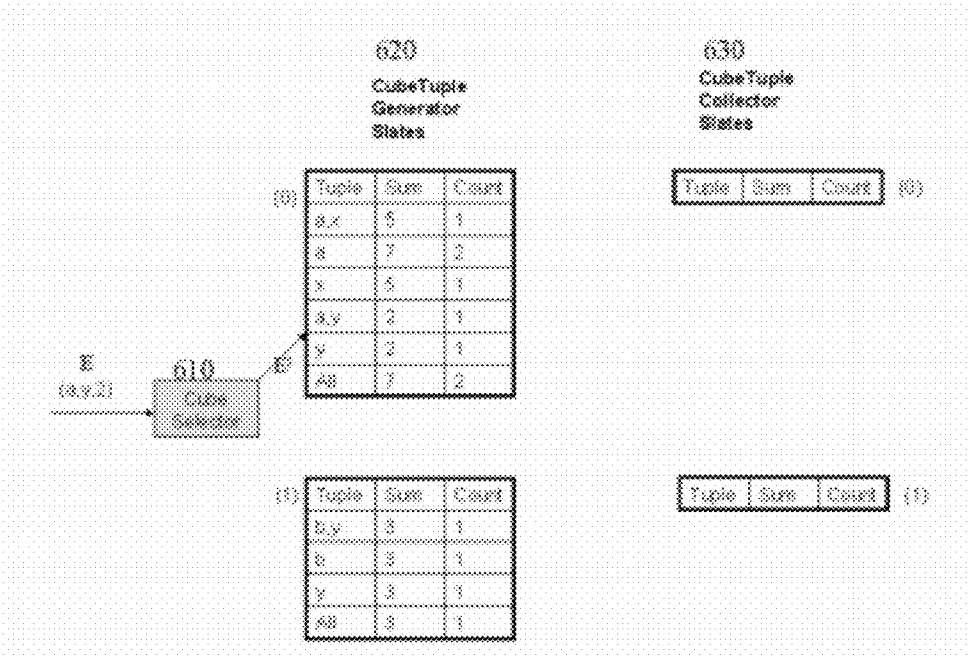


Figure 6c

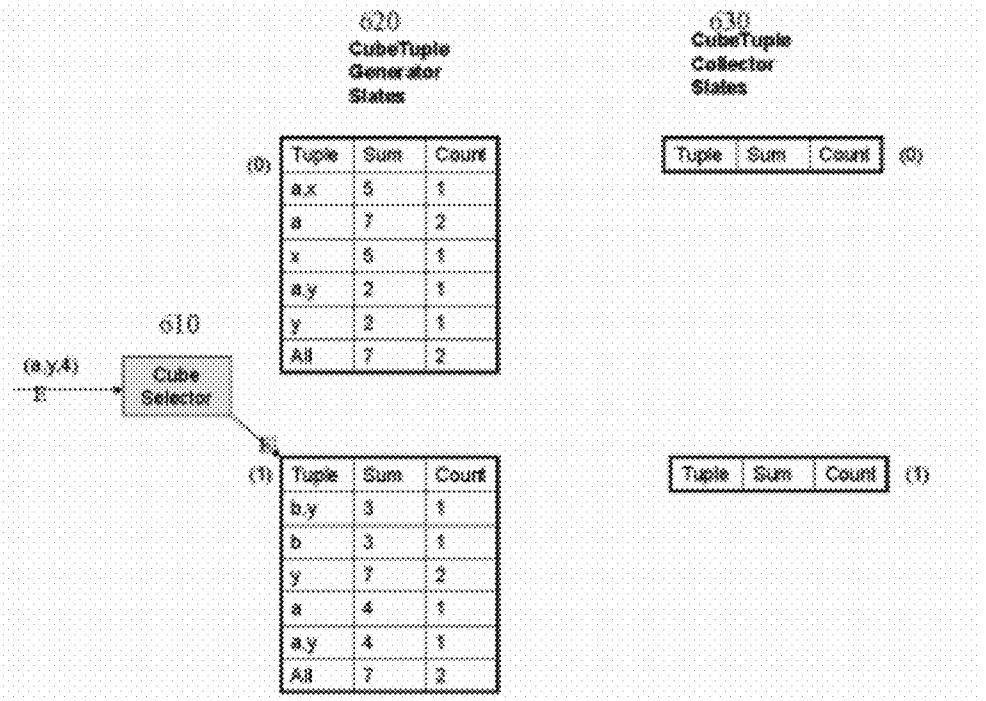


Figure 6d

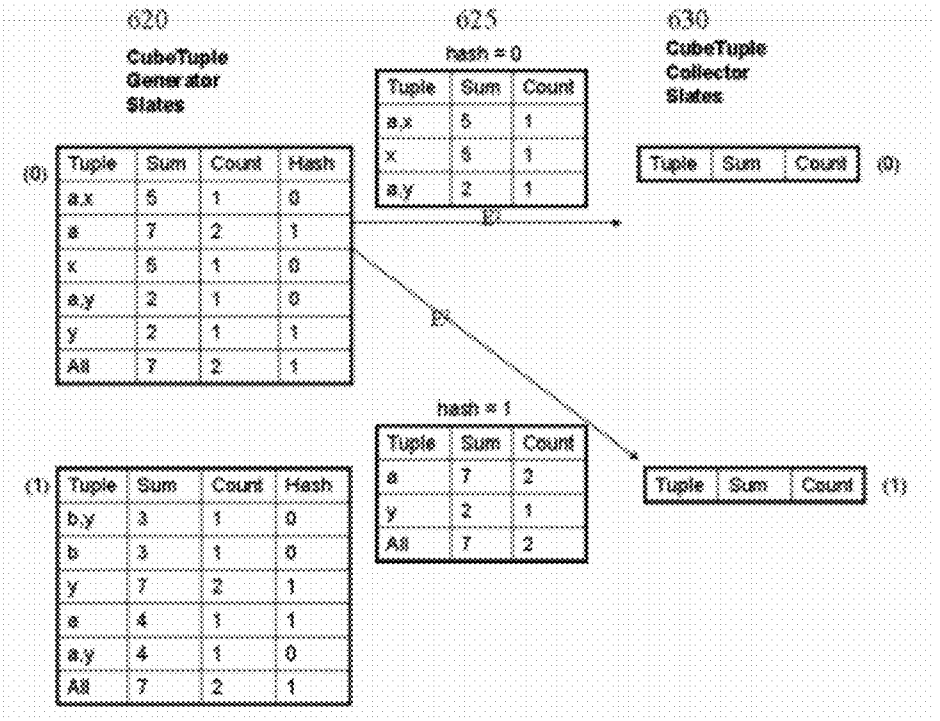


Figure 6e

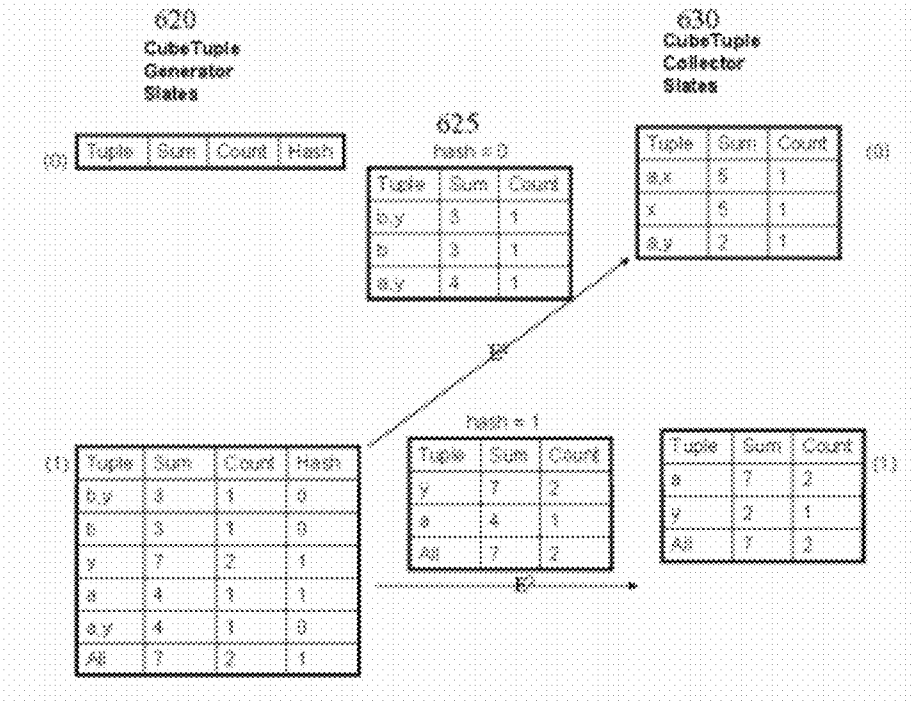


Figure 6f

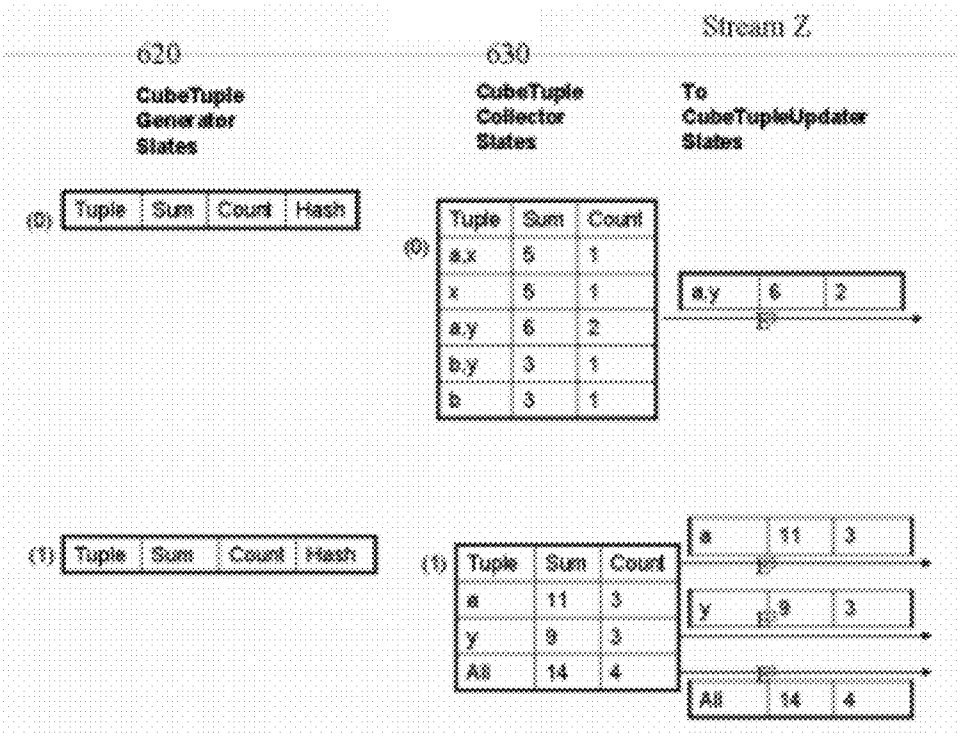

Stream Z
620
630
CubeTuple Generator States
CubeTuple Collector States
To CubeTupleUpdater States
(0)
Tuple
Sum
Count
Hash
(0)
Tuple
Sum
Count
a,x
5
1
x
5
1
a,y
6
2
b,y
3
1
b
3
1
a,y
6
2
(1)
Tuple
Sum
Count
Hash
(1)
Tuple
Sum
Count
a
11
3
y
9
3
All
14
4
a
11
3
y
9
3
All
14
4

Figure 6g

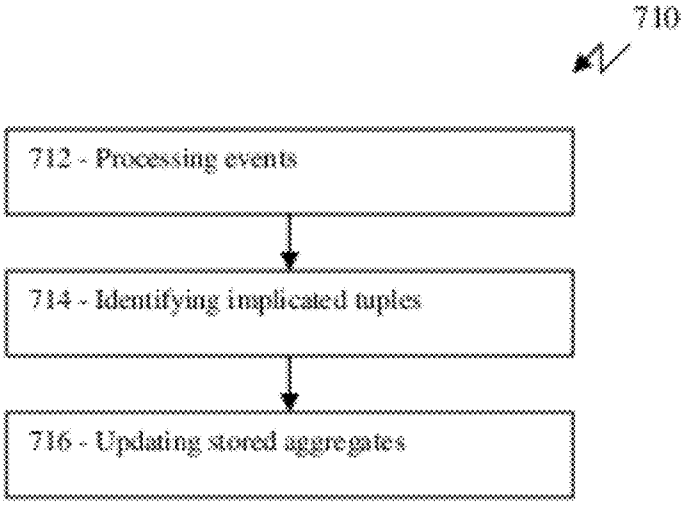


Figure 7a

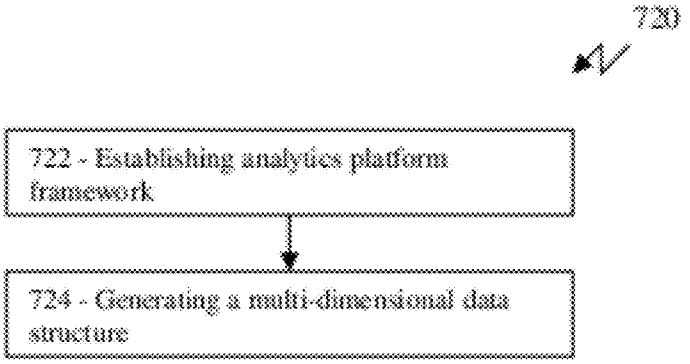


Figure 7b

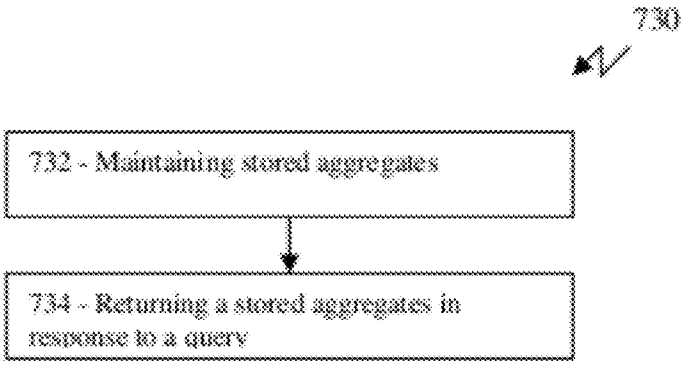


Figure 7c

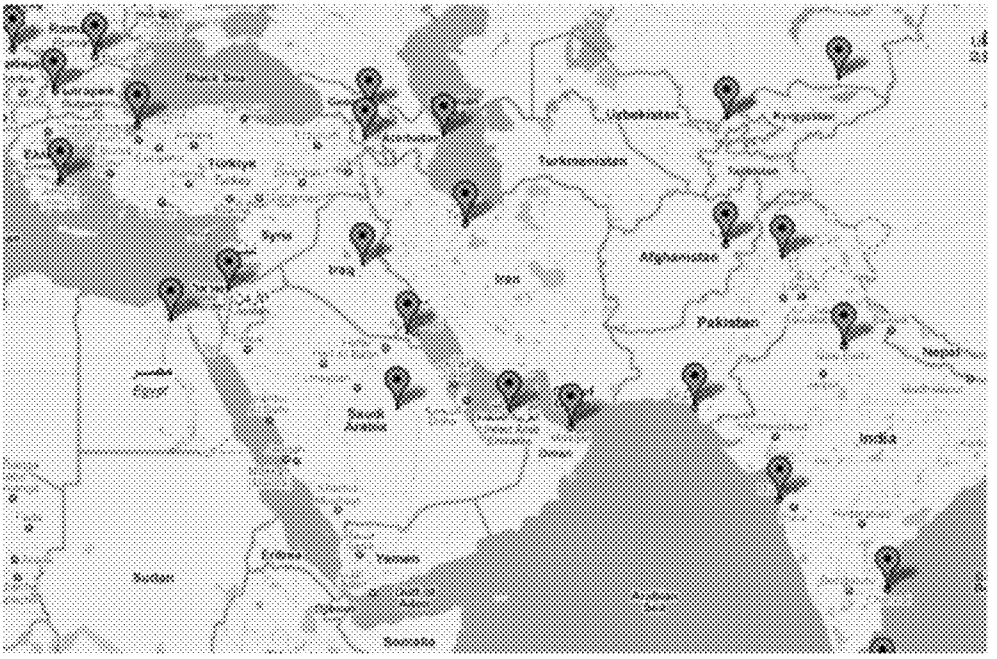


Figure 8a

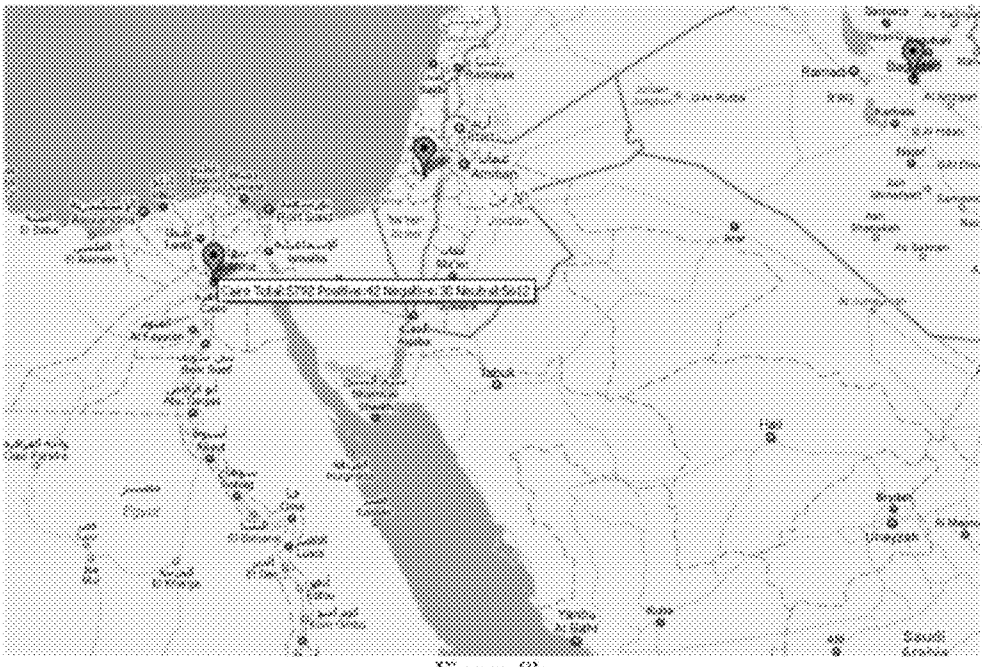


Figure 8b

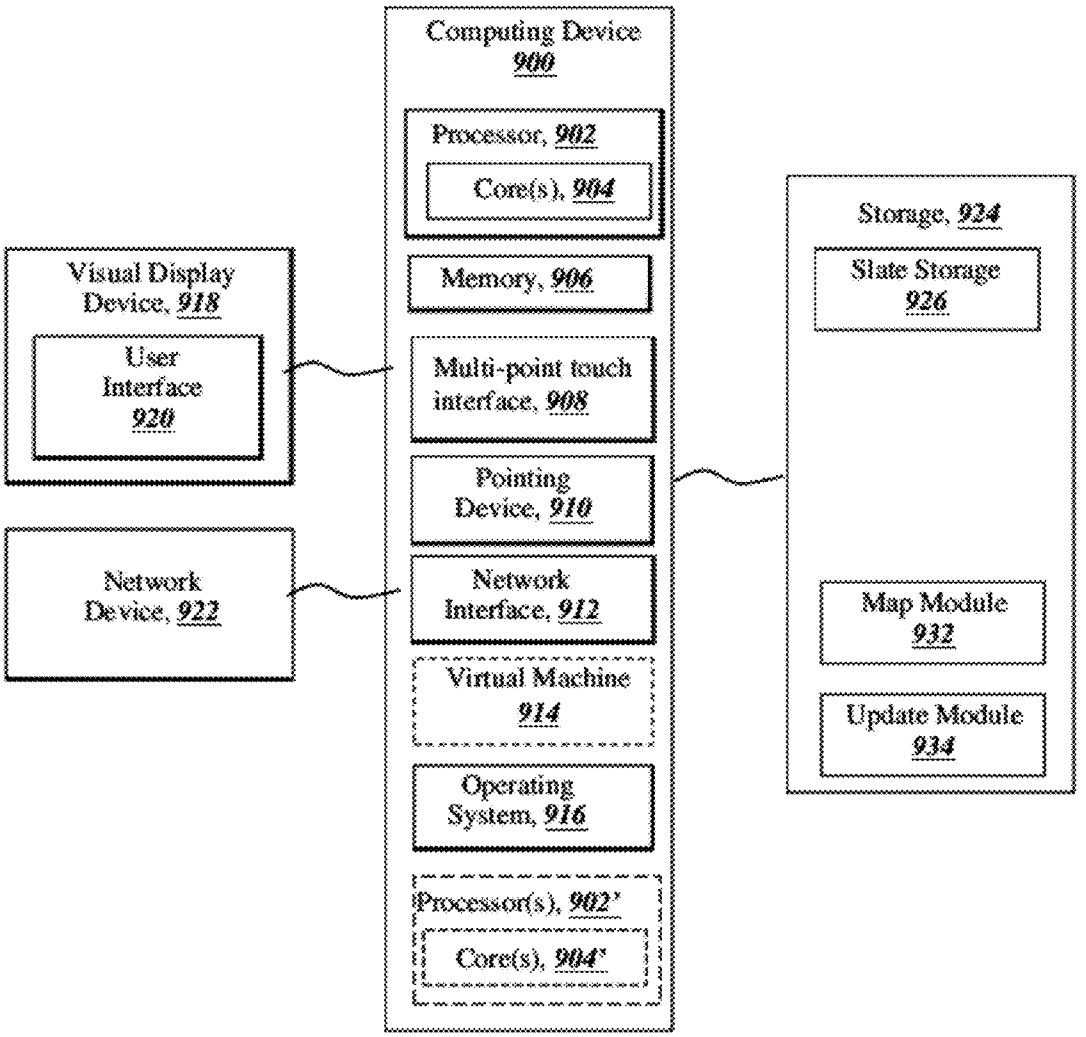

Computing Device
900
Processor, 902
Core(s), 904
Memory, 906
Multi-point touch interface, 908
Pointing Device, 910
Network Interface, 912
Virtual Machine 914
Operating System, 916
Processor(s), 902'
Core(s), 904'
Visual Display Device, 918
User Interface 920
Network Device, 922
Storage, 924
Slate Storage 926
Map Module 932
Update Module 934

Figure 9

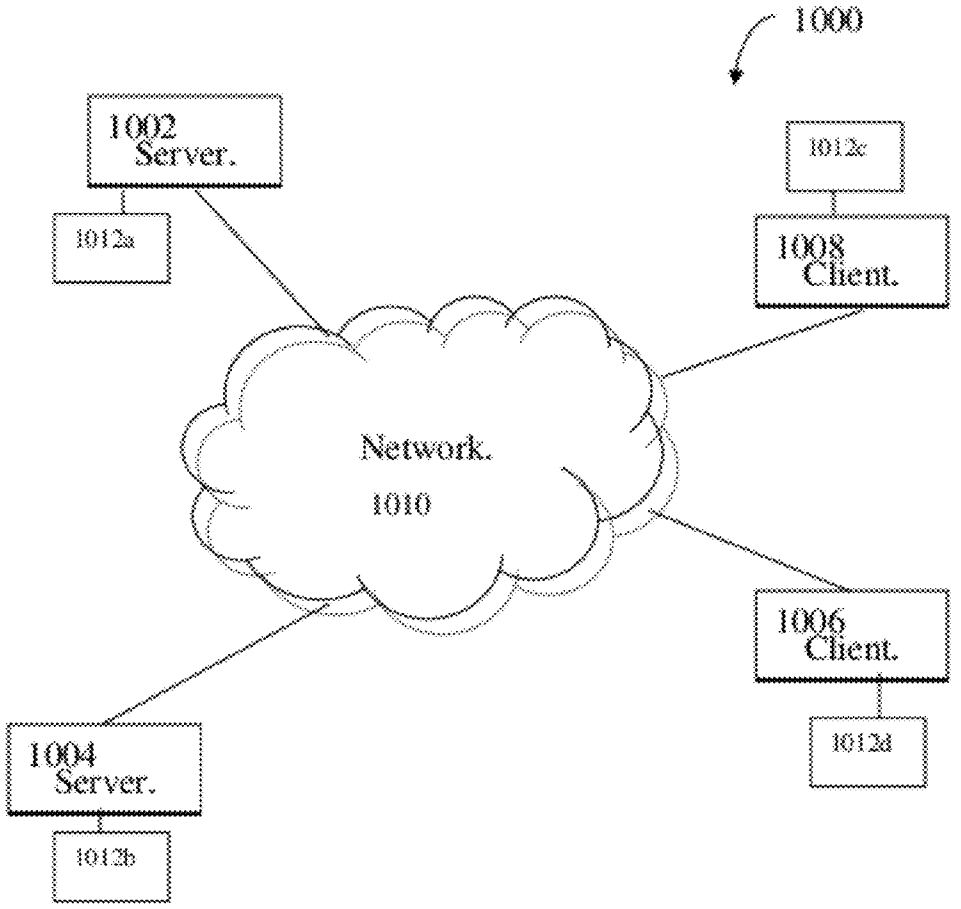

1000
1002
Server.
1012a
1012c
1008
Client.
Network.
1010
1006
Client.
1012d
1004
Server.
1012b

Figure 10

REAL-TIME ANALYTICS OF STREAMING DATA

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] The present application claims priority to U.S. Provisional Patent Application No. 61/415,279, filed Nov. 18, 2010 (entitled "Social Genome"), and U.S. Provisional Patent Application No. 61/415,282, filed Nov. 18, 2010 (entitled "Managing Real-Time Data Streams"). This application also relates to U.S. Provisional Patent Application No. 61/345,252 entitled "Content Feed," filed May 17, 2010, U.S. patent application Ser. No. 13/106,706 entitled "Processing Data Feeds," filed May 12, 2011, a U.S. non-provisional patent application titled "Processing Data Feeds," filed Nov. 18, 2011 (Attorney Docket No. 114826-50302), a U.S. non-provisional patent application entitled "Methods Systems and Devices for Recommending Products and Services" filed Nov. 18, 2011 (Attorney Docket No. 114826-50602), and a U.S. non-provisional patent application entitled "Social Genome," filed Nov. 18, 2011 (Attorney Docket No. 114826-50202). The entire contents of each of the above-referenced applications are incorporated herein in their entirety by reference.

BACKGROUND

[0002] In recent years, social media services such as Twitter™, Digg™, Myspace™ and Facebook™ have seen a meteoric rise in popularity resulting in an ever evolving universe of streaming content/data which is often user/consumer generated. Thus, social media is able to capture, better than many other sources, a raw and unfiltered pulse of society.

[0003] Potential applications for data harvested from social media are vast. For example, from a marketing intelligence standpoint, a company may gather and analyze information relevant to the company's markets to promote accurate and confident decision-making in determining market opportunity, market penetration strategy, market development metrics, etc.

TECHNICAL FIELD

[0004] The present disclosure relates to real-time analytics of data streams. More particularly, the present disclosure relates to storage media, systems and methods for processing data streams and analyzing data extracted therefrom.

SUMMARY

[0005] Storage media, systems and methods for performing real time analytics on streaming data are disclosed herein.

[0006] In exemplary embodiments a method for performing real time analytics on streaming data may include: processing events in a data stream to extract from each event a set of attribute-value pairs for one or more dimension attributes and one or more value attributes; identifying one or more tuples in a multidimensional data structure implicated by the extracted attribute-value pairs for the one or more dimension attributes; and updating, for each implicated tuple, one or more stored aggregates associated therewith, based on the extracted attribute-value pairs for the one or more value attributes. In some embodiments, a tuple frequency may be tracked over an interval for each of the implicated tuples, wherein the updating the one or more stored aggregates includes discarding each implicated tuple with a low tuple frequency over the interval. In other embodiments, for each value attribute, aggregates over an interval for a first plurality of implicated tuples having same attribute-value pairs for zero or more ordinary dimension attributes and different attribute-value pairs for one or more leaderboard dimension attributes may be tracked, and a top-N values determined for the one or more leadership dimension attributes over the interval, for example, wherein the Top-N values are characterized as resulting in the highest aggregates, the lowest aggregates or the aggregates closest to a selected value, over the interval. In yet other embodiments, the one or more dimension attributes may include a K-Gram for identifying topics of interest. Thus, for example a tuple frequency for each of the implicated tuples including a K-Gram may be tracked over an interval, wherein the updating the one or more stored aggregates includes discarding each implicated tuple with a low tuple frequency over the interval, whereby statistics for trending K-Gram-value pairs are tracked.

[0007] In other exemplary embodiments, a method for implementing a real time analytics platform may include establishing an analytics platform framework characterized by one or more time windows, one or more dimension attributes, and one or more value attribute; and generating a first multi-dimensional data structure for maintaining, for each tuple of the one or more dimension attributes, an aggregate of each of the one or more value attributes over each of the one or more time windows.

[0008] In other exemplary embodiments a method for performing real-time analytics on a data stream may include: processing a data stream to maintain a plurality of stored aggregates for a universe of prospective queries extrapolated from a pre-established framework of possible query parameters; and returning one of the stored aggregates in response to a query.

[0009] In exemplary embodiments, a system for performing real time analytics on streaming data, may include: a processor for processing an event in a data stream to extract a set of attribute-value pairs for one or more dimension attributes and one or more value attributes; a mapper for identifying one or more tuples in a multidimensional data structure implicated by the extracted attribute-value pairs for the one or more dimension attributes; and one or more updaters for updating, for each implicated tuple, one or more stored aggregates associated therewith, based on the extracted attribute-value pairs for the one or more value attributes.

[0010] In other exemplary embodiments, a system for performing real-time analytics on a data stream may include: a processor for processing a data stream to maintain a plurality of stored aggregates for a universe of prospective queries extrapolated from a pre-established framework of possible query parameters; and memory for storing the plurality of stored aggregates.

[0011] In exemplary embodiments, a multi-dimensional data structure, for implementing a real-time analytics platform characterized by one or more time windows, one or more dimension attributes, and one or more value attributes, may include: a plurality of tuples associated with the one or more dimension attributes; and a slate associated with each tuple for maintaining an aggregate for each of the one or more value attributes over each of the one or more time windows.

[0012] In other exemplary embodiments, a multi-dimensional data structure for implementing a real-time analytics platform may include: a plurality of stored tuples each representing a set of search query parameters for prospective que-

ries extrapolated from a pre-established framework of possible query parameters and one or more stored aggregates associated with each of the stored tuples, wherein each aggregate represents a result for a prospective query characterized by the set of search query parameters represented in the tuple associated with that aggregate.

[0013] In exemplary embodiments, a non-transitory computer readable medium may store processor executable instructions for performing methods described herein. For example, the computer readable medium may store processor executable instructions for processing events in a data stream to extract from each event a set of attribute-value pairs for one or more dimension attributes and one or more value attributes; identifying one or more tuples in a multidimensional data structure implicated by the extracted attribute-value pairs for the one or more dimension attributes; and updating, for each implicated tuple, one or more stored aggregates associated therewith, based on the extracted attribute-value pairs for the one or more value attributes. In other embodiments, the computer readable medium may store processor executable instructions for establishing an analytics platform framework characterized by one or more time windows, one or more dimension attributes, and one or more value attribute; and generating a first multi-dimensional data structure for maintaining, for each tuple of the one or more dimension attributes, an aggregate of each of the one or more value attributes over each of the one or more time windows. In yet other embodiments, the computer readable medium may store processor executable instructions for processing a data stream to maintain a plurality of stored aggregates for a universe of prospective queries extrapolated from a pre-established framework of possible query parameters; and returning one of the stored aggregates in response to a query.

[0014] The foregoing and other objects, aspects, features and advantages of exemplary embodiments will be more fully understood from the following description when read together with the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

[0015] FIG. **1** depicts an exemplary data stream, according to the present disclosure.

[0016] FIG. **2** depicts an exemplary query, according to the present disclosure.

[0017] FIG. **3** depicts an exemplary data cube, according to the present disclosure.

[0018] FIG. **4** depicts an exemplary implementation of a distributed architecture for maintaining a data cube, according to the present disclosure.

[0019] FIG. **5** depicts another exemplary implementation of a distributed architecture for maintaining a data cube, according to the present disclosure.

[0020] FIGS. **6***a-g* depict a sequence of events for a worked example using the distributed architecture of FIG. **5**, according to the present disclosure.

[0021] FIGS. **7***a-c* depict flowcharts for exemplary methods for performing real time analytics on streaming data, according to the present disclosure.

[0022] FIGS. **8***a-b* depict overlaying real-time social statistics on a geographic map.

[0023] FIG. **9** depicts an exemplary computing device for implementing embodiments of the present disclosure.

[0024] FIG. **10** depicts an exemplary network environment for implementing a distributed architecture, according to the present disclosure.

DETAILED DESCRIPTION

[0025] Storage media, systems and methods are disclosed herein for analyzing data streams in real time and/or pre-computing statistics in real time with no query time computation. More particularly, storage media, systems and methods are presented for processing data streams to calculate results for prospective queries. The results may be advantageously computed prior to the formulation of the specific query, for example, based on a pre-established framework of potential query parameters. More particularly, a universe of potential queries may be extrapolated from the pre-established framework of potential query parameters. Results for each of the potential queries may them be tracked in real time. For example results for each of the potential queries may be continuously updated based on real-time processing of events in a data stream.

[0026] Note that, as used herein the term event generally refers to an atomic unit in a data stream, for example, a single tweet™ in a Twitter™ feed or a single purchasing transaction in a transaction stream. In exemplary embodiments, a data stream may include a continuous flow of data that is not pre-divided into discrete events. Thus, in some embodiments, an event may be inferred, for example, by identifying a set of one or more related attributes in the data stream. For example, related attributes may be identified based on temporal and/or source commonalities. In some embodiments, a contextual analysis (for example, a semantic analysis) of attributes in a data steam may be used to identify a set of one or more related attributes. Exemplary embodiments of semantic analysis, for example using a doctagger to identify and/or group topics, are is described herein

[0027] It is appreciated that, although exemplary embodiments presented herein relate to social analytics, the storage media, systems and methods of the present disclosure may be used for real-time analysis of any type of streaming data, structured or unstructured. For example, the storage media, systems and methods of the present disclosure may be used for real-time analysis of purchase transactions, customer reviews/feedback, customer wish lists/shopping carts, etc.

[0028] In exemplary embodiments, prospective queries of data streams may include queries related to social statistics. For example:

[0029] How many events relate to product P between 10 a.m. and 11 a.m. today?

[0030] What percentage of events had a positive opinion of product P yesterday?

[0031] How do women in Arizona feel about a limited time offer, based on events during the offer?

[0032] What is the average age of women in Arizona who purchased product P last year?

[0033] The result for each of the above queries may be calculated as an aggregate of a value attribute (number of events, percentage of events, sentiment, and average age, respectively) over a specified time window (between 10 a.m. and 11 a.m. today, yesterday, during the limited time offer, and last year, respectively) for a specified set of dimension attributes (related to product P, positive sentiment related to product P, related to the limited time offer from women in Arizona, and related to women in Arizona who purchased product P, respectively). The storage media, systems and

methods of the present disclosure advantageously facilitate identifying and maintaining time-based aggregates, such as described above, prior to the formulation of a queries.

[0034] In exemplary embodiments, a framework of potential query parameters may be pre-established by selecting, for example, via a user input, attributes of interest including one or more time windows, one or more dimension attributes and one or more value attributes. The framework may further include, for each of the one or more value attributes, an aggregate function defining how to aggregate instances of the value attribute. As used herein, the term dimension attribute refers to an identifiable attribute in a data stream which is of interest as pertaining to a query search parameter. By comparison, the term value attribute refers to an identifiable attribute in a data stream of interest which is of interest as pertaining to a query result parameter. Notably, depending on the particular framework of potential query parameters, a same attribute may be both a dimension attribute and a value attribute. For example, the attribute "sentiment" may be used as both a search parameter (such as, in the query "how many women have a positive opinion about product P?") and a result parameter (such as in the query "what is the sentiment of women regarding product P?").

[0035] In exemplary embodiments a pre-established framework of query parameters may be used to generate a multi-dimensional data structure for maintaining, for each tuple of the one or more dimension attributes; an aggregate of each of the one or more value attributes over each of the one or more time windows. As used herein the term tuple may refer to a set of dimension attribute-value pairs. For example, for exemplary dimension attributes Person (P) Location (L) and Thing (T), a tuple may take the form P=p, L =**1**, and T=t, (also expressed as the tuple (p, l, t) for dimensions (P, L, T)).

[0036] In exemplary embodiments, the aggregates stored in the multi-dimensional data structure may be updated for each new event processed from a data stream. In particular, the event may be analyzed to extract a set of related attribute-value pairs including for one or more dimension attributes and one or more value attributes. The extracted set of related attributes-value pairs may then be used to identify or more implicated tuples in the data structure for updating. Thus, aggregates associated with each of the implicated tuples for each of the one or more value attributes, may be updated by applying an appropriate aggregation function to each identified value attribute-value pair. In this way the multi-dimensional data structure may maintain real-time analytics of the data stream.

[0037] In exemplary embodiments, a distributed architecture, such as Muppet (map, update), may be used to implement the storage media, systems and methods of the present disclosure. Exemplary implementations of Muppet are further described herein as well as in U.S. non-provisional patent application entitled "Processing Data Feeds," filed Nov. 18, 2011 (Attorney Docket No. 114826-50302). In general, a distributed architecture may be used to map an event to one or more implicated tuples in one or more multi-dimensional data structures and update, for each of the implicated tuples, one or more slates, for example based on one or more value attribute-value pairs in the event. Advantageously, slates for a plurality of implicated tuples may be updated in parallel for example, using different processing nodes.

[0038] The terms "map" and "mapper," as used herein, relate to a stream operation performed in exemplary embodiments in which events in a data stream are processed in a real-time manner to generate one or more new events which are then published to a same or different data stream. In exemplary embodiments, a mapper may be used to publish events to one or more updaters for updating an aggregate value contained in a slates.

[0039] The terms "update" and "updater" refer to a stream operation performed in exemplary embodiments in which events in one or more real-time data streams are processed in a real-time manner to create or update one or more persistent static "slate" data structures that are stored in a persistent manner, for example, in a durable disk storage (note that, as used herein the terms "store," "stored" "storage" etc., imply persistence a non-transitory storage medium). In some exemplary embodiments, an updater may generate zero, one or more new stream events. The generated stream events may be published to one or more real-time data streams. In an exemplary embodiments, an updater may publish stream events to a data stream from which it accepts stream events as input.

[0040] As used herein, the term "slate" refers to a static data structure that may be used to record aggregates as described herein. A slate may have any suitable data structure or format. In an exemplary format, a slate may include a collection of one or more attribute-value pairs. A slate may be stored corresponding to a unique slatekey and updater that updates the slate.

[0041] In exemplary embodiments, time-based aggregates of a given value attribute over a given time-window may be maintained in a multi-dimensional data structure (sometimes referred to herein as a data cube). The dimensions of the data structure generally reflect one or more dimension attributes selected in a pre-established framework of potential query parameters.

[0042] In a naïve implementation of a distributed architecture for maintaining the data structure, fan-out may exponentially relate to the number of dimensions in the data structure. The term "fan-out" for a distributed architecture may refer to the ratio of internal events generated by the mapping function relative to the number of external events (e.g., tweets™) processed.

[0043] Since, handling and storing such a volume of data may prove impractical, alternative implementations of a distributed architecture are also presented herein that take advantage of various properties of data streams to considerably reduce fan-out to a manageable number.

[0044] With initial reference to FIG. **1**, an exemplary event **100** in a data stream **10** is depicted. Event **100** may be processed to identify a plurality of attribute-value pairs **110** contained therin. Examples of attributes may include:

[0045] Event ID, for example, a unique per-event identifier.

[0046] Sentiment, for example, with potential values of +1, 0, or −1, indicating positive, neutral, or negative sentiment.

[0047] Gender, for example, with potential values of Male, Female, and Unknown.

[0048] Country, for example, with potential values drawn from an enumerated set of country codes including unknown.

[0049] Topic, for example, with potential values detected via semantic analysis.

[0050] Product, for example, with potential values drawn from a product database.

[0051] Price, for example, with potential values in different currencies.

[0052] Timestamp, for example, based on time published or time received.

[0053] One usefull attribute, accordingly to the present disclosure, is the timestamp. In exemplary embodiments, two assumptions may be made regarding the timestamp: first, that the timestamp represents actual wall-clock time in some appropriate timezone; and second, that timestamps are monontonically increasing. These assumptions are generally reasonable for streaming data (for example, in Twitter™ each tweet™ contains a timestamp that satisfies these conditions). One reason that timestamps are useful is that query results are represented as aggregates over a time window. Using timestamps, aggregates may be unambiguously interpreted to include a set of events whose timestamps fall within the specified time window.

[0054] With reference to FIG. **2**, an exemplary query **200** is depicted. Query **200** may specify, for example, a time window **210**, a set of dimension attribute-value pairs **220**, one or more value attributes of interest **230**, and an aggregate function **240** related to each value attribute **230**. Exemplary instances of query **200** are described below:

Example 1

[0055] Query: How many people posted about product P between 10 am and 11 am today?
Time Window: 10 am to 11 am today

Dimension Attribute-Value Pair: Product=P

[0056] Value attribute: Event Id

Aggregate Function: Count

[0057] Example 1 may be rewritten as the following SQL query:

SELECT COUNT(EventId)

[0058] FROM event E
WHERE t.product="P"
[0059] AND t.timestamp>=10 am AND t.timestamp<=11 am

Example 2

[0060] How many women in Arizona posted about product P between 10 am and 11 am?
Same as last example except for additional Dimension Attribute-Value Pairs:

Gender="F", State="AZ"

Example 3

[0061] What was the sentiment about product P among women in Arizona in December 2010?

Time Window: December 2010

Dimension Attribute-Value Pairs: Product="P", Gender="F", State="AZ"

Value Attribute: Sentiment

Aggregate Function: AggregateSentiment

[0062] In the third query example, AggregateSentiment represents a custom-defined aggregation function for combining sentiment values. For example, the function may maintain a 3-tuple of count, one each for positive, neutral, and negative sentiments. Alternatively, the function may be configured to calculate an average sentiment.

[0063] Time Windows:

[0064] In exemplary embodiments, aggregates may be calculated for each of a plurality of time windows. In some embodiments arbitrary time windows may be supplanted by a standard set of time windows, for example:

[0065] By the minute for the past 60 minutes

[0066] By the hour for the past 24 hours

[0067] By the week for the past 4 weeks

[0068] By the month for the past 24 months (for y-o-y same month comparisons)

[0069] By the year for the past 10 years

[0070] In general, the standard set of time windows may reflect an assumption that the further back time the coarser the time granularity of interest. Thus, the standard set of time windows may include time windows of varying time granularity. In exemplary embodiments, it may be sufficient to maintain aggregates for a finite number of progressively older and courser sets of time windows, such reflected above.

[0071] Aggregate Functions:

[0072] In general there are two kinds of aggregate functions: Algebraic and Holistic. Roughly speaking, algebraic aggregates are those, like SUM, that can be computed incrementally; in other words, by aggregating subsets of the data, and computing the final result using those aggregates without going back to the base data. In contrast, Holistic aggregates typically require the base data when recalculating the aggregate. One example of a holistic aggregate is the median. Suppose you divide a data set arbitrarily into two parts, and compute the median of the two parts; there is no way to compute the median of the entire data set from the medians of the two parts.

[0073] The systems and method of the subject disclosure typically utilize algebraic aggregation functions, which may be computed incrementally. Thus, for example, an average may be represented as a 2-tuple a sum and a count, wherein the average may be calculated by dividing the sum by the count. The use of algebraic aggregations functions, advantageously simplifies the update process, thereby facilitating the real-time data processing and analytics as described herein. In exemplary embodiments aggregates are assumed to be commutative and associative, which makes manipulations thereof simpler.

[0074] Data Cubes:

[0075] As noted above, the storage media, systems and methods of the present disclosure may advantageously utilize a multi-dimensional data structure for maintaining a universe of aggregates for a pre-established framework of potential query parameters, wherein the pre-established framework of potential query parameters is characterized, by one or more time windows, one or more dimension attributes, one or more value attributes and one or more aggregation functions.

[0076] Suppose, for example, a framework characterized by time window W, dimension attributes Topic (T), State (S), Gender (G), a single value attribute Sentiment (Se), and an aggregate function f. The following are the aggregates may be of interest for instances t, s and g of T, S and G:

(1) T=t, S=s, G=g, f(Se), W
(2) T=t, S=s, f(Se), W
(3) T=t, G=g, f(Se), W
(4) S=s, G=g, f(Se), W
(5) T=t, f(Se), W

(6) S=s, f(Se), W
(7) G=g, f(Se), W
(8) All, f(Se), W (computed across all events in time window W)

[0077] The multi-dimensional data structure for storing aggregates f(Se), W for all potential combinations and values of t, s, and g, may be referred to as the data cube for aggregate f(Se) and time window W. The term data cube refers to the fact the aggregates may be arranged as the vertices of a hypercube. In general, given K dimension attributes, there are 2^K aggregates for each set of values of the dimension attributes, which is the number of vertices of a hypercube in K dimensions.

[0078] As describe herein, a set of dimension attribute-value pairs, such as T=t, G=g, may be referred to as a tuple. Referring to the above example, the data cube for the aggregate f(Se) allows rapid lookup of the aggregate of value attribute Se for every tuple over timewindow W. In exemplary embodiments, a data cube may store aggregates for a plurality of different time windows, for example for a standard set of time windows such as described herein.

[0079] With reference to FIG. **3**, an exemplary data cube **300** is depicted for three dimension attributes: Topic (T) **310**, Location (L) **320** and Sentiment (S) **330**. Topic (T) **310** may include as instances, names for various topics, for example, as grouped via a semantic hierarchy. Location (L) **320** may include as instances, names of locations, for example names of States in the united States. Sentiment (S) **330** may include instances selected from positive negative or neutral. The dimension attributes Topic (T) **310**, Location (L) **320** and Sentiment (S) **33** may be reflected along the vertices of the data cube **300**.

[0080] The data cube **300** may maintain for each tuple (t, **1**, s) of T, L, S an aggregate for a value attribute (in this case: event count, i.e., the number of events processed for the tuple (t, l, s)). In exemplary embodiments, each tuple (t, l, s) may be associated with a slate for storing the event count. In some embodiments the slate may further be associated with an updater for updating the slate for a new event and a mapper for mapping new events to the updater.

[0081] In exemplary embodiments, data cube **300** is updated based on a new event. Thus, a new event may be processed to identify one or more dimension attributes therein. For example, a new event may state "I love living in NYC," from which dimension attributes Person (P), Location (L) and Sentiment (S) may be extracted (for example P=user L=NYC (New York City), and S=positive may be extracted. The tuples of (P, L, S) implicated are as follows:

[0082] (user, NYC, positive)
[0083] (user, NYC,)
[0084] (user, , positive)
[0085] (, NYC, positive)
[0086] (user,)
[0087] (, NYC,)
[0088] (, positive)
[0089] (, ,)

[0090] Thus, overlapping tuples of (T, L, S) implicated by the new event are:

[0091] (, NYC, positive)
[0092] (, NYC,)
[0093] (, , positive) and
[0094] (, ,)

[0095] Thus, the event count associated with each of the four implicated tuples may be updated (for example, by incrementing the count by one).

[0096] Note that for an event containing L dimension attributes, M of which overlap with K dimension attributes of a data cube, there are 2^M tuples of the data cube which are implicated (i.e. 2^M tuples which overlap between the data cube and the event). Thus, a mapper may be used generate the 2^L tuples for the event and map the 2^M subset thereof to the data cube. An update may then be used update a slate associated with each of the 2^M tuples received from the mapper. Distributed architecture implementations for maintaining a data cube are described in greater below

[0097] Data cube **300** advantageously maintains, in real time, results for any prospective query using the framework (T, L, S), of an event count over one or more pre-established time windows. FIG. **3**, illustrates three examples of queries **340***a-c*, the answers to which are maintained and therefore pre-computed in data cube **300**. For example, query **340***a* asks "how many people are posting about Barack Obama in New York?" The result to query **340***a* may be obtained by returning the event count for the tuple (Barack Obama, New York, All), for example the event count stored in the slate associated with the tuple (Barack Obama, New York, All). As another example, query **340***b* asks "How many people in Arizona feel positive of the new Medicare plan?" The result to query **340***b* be obtained by returning the event count for the tuple (Medicare, Arizona, Positive). As another example, query **340***c* asks "How many people feel negative of Barack Obama across the US?" The result to query **340***c* may obtained by returning the event count for the tuple (Barrack Obama, United States (e.g., all states), Negative).

[0098] In exemplary embodiments, it is contemplated that the number of the dimension of a data cube may be automatically determined based on the types of attributes reflected in the data stream. For example, event types with the greatest frequencies, such as above a selected threshold, may be used as the dimensions for the data cube. Thus, for example the data stream may be analyzed to determine the best candidate attributes for cube dimensions.

Naïve Distributed Architecture Implementation:

[0099] Referring to FIG. **4**, an exemplary implementation of a distributed architecture **400** for maintaining a data cube may include a mapper, for example, CubeMapper **410**, and one or more updaters, for example, CubeTupleUpdaters **420**. The mapper and/or updaters may be distributed to one or more processing nodes in the distributed architecture, e.g., via a network architecture such as described herein, for example with reference to FIG. **10**.

[0100] The CubeMapper **410** may advantageously determine, for example based on a set of attribute-value pairs extracted for an event, which data cubes to maintain. In exemplary embodiments, a data cube may be defined by a set of dimension attributes (for example, Topic, Gender, State), and an aggregation function for a value attribute. In exemplary embodiments, a configuration file may list the data cubes of interest, and give each data cube a name. The aggregation function may be specified, for example, in javascript, as a function that takes two parameters (the current value of the aggregate and a new event) and returns a single value (the new value of the aggregate). Note that as a special case, the current value may be null, in which case the function may return the aggregate corresponding to just the one event. The value of the aggregate may be in any data structure/format, for example a JSON object.

[0101] In some embodiments, a generated data cube may apply only to specific kinds of topics. For example, the query

framework may call for a data cube specific for persons of interest, such as customers, celebrities, etc., or for occasions such as holidays, the Oscars, etc. Thus, in exemplary embodiments the mapper may implement a selection function, based on selection criterion, to filter out only a subset of events from a data stream. The selection function may, for example, accept a single parameter (the event) and return either True (this event's data should be part of this data cube) or False.

[0102] An example of a configuration file listing data cube's of interest is provided below:
Cubes.congfig:

CubeName: SentimentCube

[0103] Select Function: True ##all events

Dimensions: Topic, State, Gender

[0104] AggregateFunction: lambda(sentiment, event) { . . . ; return sentiment}

CubeName: OscarsVotes

[0105] SelectFunction: lambda(event) {return event.event_id=oscars;}

Dimensions: Topic, Gender, Age

[0106] AggregateFunction: lambda(sentiment, event) { . . . ; return sentiment}

[0107] The CubeConfig file may advantageously be replicated for reference at each processing node in the distributed architecture.

[0108] The CubeMapper **410** may processes each event in a data stream and determine which data cubes it is eligible for. Suppose an event E with K dimension elements (for example, K=2 dimension elements a and b depicted in FIG. **3**) is eligible for a data cube. The CubeMapper **410** constructs the 2^K tuples from the event E (for example, the 2^2 tuples: (a), (b), (a,b) and (All) depicted in FIG. **3**) and generates an event E^1 for each tuple. The key for each event E^1 is the pair (CubeName, Tuple) and the value is the content event E, for example, including a value attribute-value pair. The generated events E^1 are sent on to the CubeTupleUpdaters **420**.

[0109] Each CubeTupleUpdater **420** maintains a slate for every (CubeName, Tuple) pair it receives. Thus, the CubeTupleUpdater receives an event E^1 for a single tuple, extracts an instance of a value attribute and applies the aggregation function to add the instance to its store. In exemplary embodiments, The updater keeps track of the aggregate value for a plurality of time windows for example a standard set of time windows such as described herein. Thus, a query for any tuple and any Time Window may be answered via a quick slate lookup.

Alternative Distributed Architecture Implementations:

[0110] There are two potential problems with the naïve implementation described above. First, for a data cube of K dimensions, the CubeMapper **410** may generate 2K events for each incoming event. This leads to very high fan-out for cubes with K>3. Second since, statistics are stored for every possible tuple that occurs in the data, The number of cube slates required is proportional to the number of possible combinations of dimension values that actually occurs in the data. For example, suppose there are 50 states, 2 genders, and 1 million topics. The number of slates needed may approach 50×2×1 million or 100 million. This may be impractical from a storage perspective. Exemplary alternate implementations of a distributed architecture present herein may help mitigate/prevent such potential problems.

[0111] A key property of data stream analytics is that events don't exist in a vacuum but rather often reflect and are influenced by a collective pulse. Thus, events often exhibit a great deal of clustering, for example, of topics, products, people, etc. Moreover, it is expected that queries to the data cube involve instances of dimension elements that are of interest to a large number of people. Taking advantage of the forgoing assumptions, tuples may be advantageously filtered based on frequency. Frequency filtering may be implemented, for example, by selecting a small time window (for example 1 minute) referred to as the delta window D. Let S be the set of all tuples corresponding to all social updates during a delta window D. A threshold δ is applied to filter out all tuples in S with frequency less than δ from being sent to updaters (for example, CubeTupleUpdaters **320** of FIG. **3**). For example, δ may be selected to be 1 or 2. A simple experiment with actual data suggests that setting δ to 2 may eliminate over 90% of tuples in a delta window of 1 minute. Moreover, a qualitative examination of such tuples indicated that the eliminated tuples generally came from events that are not really of interest (for example, spam, outliers of some sort, or just semantic analysis errors). Thus, filtering also a second effect of reducing noise, e.g., from semantic analysis errors. On the other hand, tuples that do occur frequently in a 1-minute window often correlate well with the global interests. Frequency filtering of tuples thus both improves performance and improves the quality of the data cube.

[0112] FIG. **5** depicts an exemplary distributed architecture **500** for maintaining a data cube while implementing frequency filtering. Thus, the distributed architecture **500** may include a mapper (CubeS elector **510**) and 3 types of updaters (CubeTupleGenerators **520**, the CubeTupleCollectors **530**, and the CubeTupleUpdaters **540**).

[0113] Suppose T is a current timestamp. Interval I may then be defined as follows: I=floor(T/D), where D is the delta window (e.g., 1 minute). That is, the interval I counts time in units of the Delta Window. The CubeTupleGenerators **520** buffer all tuples with the same Interval, and then dispatch them to the CubeTupleCollectors **530**.

[0114] In exemplary embodiments, an assumption may be made that the stream has a large number of events in each Delta Window—that is, the Delta Window is very large compared to the average gap between events (for example, Twitter™, processes approximately 100,000 tweets™ in a Delta Window of 1 minute resulting in an average inter-event gap of less than a millisecond). Thus, in exemplary embodiments, one may detect when an interval has ended and the next one has begun based on a processing of the first event whose Interval is higher than the current Interval. Alternatively, intervals may be tracked independent of events received.

[0115] In some embodiments, intervals may be based on e.g., a requisite number of events received rather than a time frame (for example, provided that the integration of such batches over the one or more time windows being tracked results in an acceptable margin of error). In other embodiments, the threshold δ may be variable, e.g., based on a changing frequency of events in the data stream. In some embodiments, a different threshold may be applied depending on the number of dimension attributes-value pairs in the

tuple. Thus, frequency filtering may account for variations in event traffic when filtering tuples.

[0116] With reference again to FIG. **5**, for each data cube, there may be P CubeTupleGenerator slates and P CubeTupleCollector slates, with keys 1, . . . , P. So as to ensure uniform distribution across the cluster, P may be selected to be a small multiple of the number of processing nodes in the distributed architecture **500**. For example, if the distributed architecture **500** includes 8 processing nodes, one might select P=32 or 64.

[0117] The CubeSelector **510** is similar to the CubeMapper **210** in the naïve implementation. It uses the config file to determine if the current Event E is eligible for a data cube. If it is, the CubeSelector **510** uses a hash function to map the timestamp of the event to one of P values, and emits to stream X an event E^1 whose key is (CubeName, P) and value is the content of the event E.

[0118] Each CubeTupleGenerator **520** (CTG) subscribes to stream X. A CTG generates all possible tuples from an event E^1 and maintains a table with 3 columns: Tuple, AggregateValue, and Count. Here Count is the number of events that have contributed to this tuple. The CTG also stores the Interval I of the first event it received during this Delta Window. When the CTG receives a new event E^1, it computes its Interval J. If J=I, the CTG enumerates the tuples of E^1 and updates its table accordingly. On the other hand, if J>I, the CTG starts distributing its data as follows:

[0119] For each tuple, the CTG uses a hash function to map the tuple into one of P buckets 0,1 . . . , P−1. For each bucket in 1, . . . P−1, the CTG creates and emits to stream Y an event E^2 whose key is the bucket number, and whose value is the set of rows in its table whose tuple maps to that bucket number. The value also indicates the Interval I. Once the CTG has sent out all P events, it empties its tables and stores J as the new Interval value. The CTG then proceeds to process a newly received event E^1.

[0120] Each CubeTupleCollector **530** (CTC) subscribes to stream Y. For each interval I, a CTC slate receives P events , one from each CTG slate. The CTC maintains a table with 3 columns: Tuple, AggregateValue, Count. The AggregateValue for a tuple is the aggregate of the partial AggregateValues for that tuple from each CTG, and the Count is the sum of the corresponding partial counts. The CTC also keeps track of the current interval I and a message counter M, both initialized to 0. When the CTC receives an event E^2 from the CTG with interval J, it first compares I and J:

[0121] If I=J, the CTC table is updated using the tuples in the event, and M is incremented by 1. If M=P (i.e., the CTC has received events from all the P CTCs), the CTC knows it has received all the tuples for the interval I. At this point, the CTG discards all infrequent tuples with count less than the threshold δ. For every frequent tuple, it creates a new event E^3 whose key is the pair (CubeName, Tuple) and whose value contains the Aggregate and the Count. The event E^3 is published to stream Z, to which the CubeTupleUpdaters **540** subscribe. The CTC then empties its table, sets its current interval to I+1 and resets M to 0.

[0122] If J>I, this means is that the CTC has received tuples for the next interval before it receives all the tuples for the current interval. Assuming that the aforementioned assumption about the Delta Interval being much larger than the average interval between events holds true, the only real reason for this is a node failure, which should be relatively infrequent. In this case, the interval I is declared closed, the tuples accumulated thus far are filtered. I is set to be equal to J, M to 1, and process the newly arrived event.

[0123] Finally, If J<I, a delayed event for an interval that is past was received. This event is ignored.

[0124] Each CubeTupleUpdater **540** maintains a slate for every (CubeName, Tuple) pair it receives. Thus, the CubeTupleUpdater receives an event E^3 for a single tuple, extracts an instance of a value attribute and applies the aggregation function to add the instance to its store. In exemplary embodiments, The updater keeps track of the aggregate value for a plurality of time windows for example a standard set of time windows such as described herein. Thus, a query for any tuple and any Time Window may be answered via a quick slate lookup.

[0125] With reference to FIGS. **6***a-g* an exemplary worked example of the distributed architecture **500** is depicted, demonstrating exemplary contents of CTG slates **620** and CTC slates **630** of a CTG and CTC such as described above with reference to FIG. **5**. For the worked example, depicted , a data cube C with two dimensions (A and X) is assumed. The aggregate is SUM and the filtering threshold δ=1. During the Delta Window, **4** tuples arrive. In FIG. **6***a*, an event E including attribute-value pairs A=a X=x and value attribute V=5 is received by the Cube Selecter **610**. In FIG. **6***b*, an event E including attribute-value pairs B=b Y=y and value attribute V=3 is received by the Cube Selecter **610**. In FIG. **6***c*, an event E including attribute-value pairs A=a Y=y and value attribute V=2 is received by the Cube Selecter **610**. In FIG. **6***d* an event E including attribute-value pairs A=a Y=y and value attribute V=4 is received by the Cube Selector **610**. In each case, Cube Selecter **610** uses the config file to determine if the Event E is eligible for a data cube, for example data cube C. If it is, the CubeSelector **610** uses a hash function to map the timestamp of the event to one of P Values and emits an event E^1 whose key is (CubeName, P) and value is the content of the event E. As depicted in FIGS. **6***a-d* CTG slates **620** buffer all tuples with the same interval. More particularly the CTG slates **620** maintains a table with 3 columns: Tuple, AggregateValue, and Count. With reference to FIG. **6***e*, once the interval is closed, a hash function to map each from the CTG slates **620** tuple into one of P buckets 0,1 . . . , P−1, for example buckets **625** of FIG. **6***e*. For each bucket in 1,. . . P−1, the creates and emits an event E^2 whose key is the bucket number, and whose value is the set of rows in its table whose tuple maps to that bucket number. With reference to FIGS. **6***f* and **6***g*, for each interval I, CTC slates **630** receive P events, one from each CTG slate **620**. The CTC slates **630** maintains a table with 3 columns: Tuple, AgregateValue, Count. As depicted, once the CTG slate has sent out its events E^2, it empties its tables and begins processing events E^1 for a new interval. As depicted in Figure g**6**, once the CTC slates **630** have received all the tuples for a given interval all infrequent tuples, for example with count less than the threshold δ are discarded. Then for every frequent tuple, a new event E^3 is created whose key is the pair (CubeName, Tuple) and whose value contains the Aggregate and the Count. The event E^3 is published to stream Z subscribed to by CubeTupleUpdaters. Data cube C could then be updated for eligible events E^3.

Leaderboard Queries:

[0126] Exemplary implementations presented above, primarily related to point aggregates—wherein a tuple (a point in the hypercube) is specified and the desired result of the query is the corresponding aggregate value. The storage

media, systems and methods herein, however, are not limited to such implementations. Indeed in some embodiments, the data cube is used tracks results for different type of queries for, example, leaderboard queries.

[0127] A leaderboard query specifies a tuple and a leaderboard attribute, and asks for the Top-N values of the leader board attribute among events that satisfy the tuple. For example, a leaderboard query might pose the following question:

What were the Top 10 most popular topics posted about by people in Arizona between 10 am and 11 am today?

In SQL:

[0128] SELECT topic, count(eventid) as freq
FROM events e
WHERE t.state="AZ" and t.timestamp >="10 am" AND t.timestamp<"11 am"
GROUP BY topic
ORDER BY freq

LIMIT 10

[0129] Other exemplary leaderboard queries might pose the following question:

What were the Top 10 least popular topics posted about by people in Arizona between 10 am and 11 am today?

What were the Top 10 locations closest to New York where users posted about a sales event.

[0130] The data cube may easily be extended to support leaderboard queries. In the simple case, if the leaderboard attribute has a small and known set of values (e.g., state or gender), one can simply look up the slates corresponding to all possible values and pick the Top-N. The harder case occurs when the leaderboard attribute can take on a large number of values (or example, topic, product, URL, domain). In this case we cannot possibly examine all slates for all values of the attribute.

[0131] In exemplary embodiments, leaderboard queries are indicated by adding a line to the data cube config file to indicate the leaderboard dimensions for each cube and the value for "N" (for Top-N queries; e.g., 10). For example:

CubeName: SentimentCube

[0132] Select Function: True ##all events

Dimensions: Topic, State, Gender

Leaderboards: Topic(10)

[0133] AggregateFunction: lambda(sentiment, event) { . . . ; return sentiment}

[0134] To simplify the present description, assume there is at most one leaderboard dimension in each data cube (ss would be appreciated by one of ordinary skill in the art, it is straightforward to support more given the description for one). Non-leaderboard dimensions of the data cube may be referred to as ordinary dimensions.

[0135] Recall that in the distribution step of the CTG a tuple is hashed into one of P buckets. For the purposes of a leaderboard query the hash function is applied only to the ordinary dimensions of each tuple. This ensures all tuples of the form: A=a, B=b, C=*, where C is the leaderboard attribute and A and B are ordinary attributes, end up at the same slate of the CTC. The CTC can now compute the Top-N leaderboard values for each tuple, and pass them on to the CTU in the event it constructs for the tuple. For example, the Top-N values may be characterized as resulting in the highest aggregates, the lowest aggregates, and the aggregates closest to a selected value, etc.

[0136] Thus, the CTU may maintain the leaderboard for each time window. Where a time window is used for filtering by the CTC, the CTU may get the leaderboard directly from the CTC. Leaderboards for larger time windows are approximated using those for the smaller time windows. For example, an hourly leaderboard may be computed by looking at the Top 10 for each minute of the hour, and summing the minute-by-minute values. While it is possible to make some errors using this method (for example, where the most frequent item during the hour was not in the top 10 for any minute of the hour) accurate results are generally achieved.

K-grams

[0137] One potential disadvantage in querying using dimensions requiring semantic analysis, is that the semantic analysis is limited by the available hierarchy. For example, a semantic analysis engine can only tag instances it recognizes. An interesting advantage the storage media, systems and methods described herein is that the data cube may be used to maintain statistics on topics products, locations etc. (collectively, referred to as interesting topics or topics of interest) without the help of semantic analysis. Thus in exemplary embodiments a "K-Gram" is added as one of the data cube dimensions, with a relative high threshold **6** (e.g., 50). Thus, a slate is automatically created and maintained for any k-gram that was mentioned very often in a delta time window. In exemplary embodiments, a TF.IDF threshold may be instead of a pure frequency threshold to achieve better results. Thus, the data cube may be used to identify trending k-grams and automatically maintains stats for them without help from semantic analysis. In exemplary embodiments identified topics of interest can be communicated to the semantic analysis engine and "candidate topics" and the maintained statistics relating thereto can be used by the engine to whether to incorporate each candidate topic into the taxonomy.

Exemplary Methods

[0138] FIGS. **7***a-c*, illustrate exemplary methods according to the present disclosure.

[0139] With reference to FIG. **7***a*, an exemplary method **710** is depicted for performing real time analytics on streaming data. Method **710** generally includes steps of: (**712**) processing events in a data stream to extract from each event a set of attribute-value pairs for one or more dimension attributes and one or more value attributes; (**714**) identifying one or more tuples in a multidimensional data structure implicated by the extracted attribute-value pairs for the one or more dimension attributes; and (**716**) for each implicated tuple, updating one or more stored aggregates associated therewith, based on the extracted attribute-value pairs for the one or more value attributes.

[0140] With reference to FIG. **7***b*, an exemplary method **720** is depicted for performing real time analytics on streaming data. Method **720** generally includes steps of (**722**) establishing an analytics platform framework characterized by one or more time windows, one or more dimension attributes, and one or more value attribute; and (**724**) generating a first multidimensional data structure for maintaining, for each tuple of

the one or more dimension attributes, an aggregate of each of the one or more value attributes over each of the one or more time windows.

[0141] With reference to FIG. **7***c* an exemplary method **730** is depicted for performing real time analytics on streaming data. Method **730** generally includes steps of: (**732**) processing a data stream to maintain a plurality of stored aggregates for a universe of prospective queries extrapolated from a pre-established framework of possible query parameters; and (**734**) returning one of the stored aggregates in response to a query.

[0142] It is explicitly contemplated that the storage media, systems and methods presented herein may include one or more programmable processing units having associated therewith executable instructions held on one or more computer readable medium, RAM, ROM, hard drive, and/or hardware. In exemplary embodiments, the hardware, firmware and/or executable code may be provided, for example, as upgrade module(s) for use in conjunction with existing infrastructure (for example, existing devices/processing units). Hardware may, for example, include components and/or logic circuitry for executing the embodiments taught herein as a computing process.

[0143] Displays and/or other feedback means may also be included to convey detected/processed data, for example adjusted output representative of a particle characteristic. The display and/or other feedback means may be stand-alone or may be included as one or more components/modules of the processing unit(s). In exemplary embodiments, the display and/or other feedback means may be used to facilitate querying a data cube. In other embodiments, the display may be used to visualize, in real-time, various social statistics maintained by the data cube. For example, as depicted in FIGS. **8***a* and **8***b*, real-time social statistics may be overlaid on a geographic map.

[0144] The actual software code or control hardware which may be used to implement some of the present embodiments is not intended to limit the scope of such embodiments. For example, certain aspects of the embodiments described herein may be implemented in code using any suitable programming language type such as, for example, assembly code, C, C# or C++ using, for example, conventional or object-oriented programming techniques. Such code is stored or held on any type of suitable non-transitory computer-readable medium or media such as, for example, a magnetic or optical storage medium.

[0145] As used herein, a "processor," "processing unit," "computer" or "computer system" may be, for example, a wireless or wire line variety of a microcomputer, minicomputer, server, mainframe, laptop, personal data assistant (PDA), wireless e-mail device (for example, "BlackBerry," "Android" or "Apple," trade-designated devices), cellular phone, pager, processor, fax machine, scanner, or any other programmable device configured to transmit and receive data over a network. Computer systems disclosed herein may include memory for storing certain software applications used in obtaining, processing and communicating data. It can be appreciated that such memory may be internal or external to the disclosed embodiments. The memory may also include non-transitory storage medium for storing software, including a hard disk, an optical disk, floppy disk, ROM (read only memory), RAM (random access memory), PROM (programmable ROM), EEPROM (electrically erasable PROM), flash memory storage devices, or the like.

[0146] FIG. **9** depicts a block diagram representing an exemplary computing device **900** that may be used as a processing node (also referred to as a worker node) for aggregating and/or storing data as described herein, for example a processing node in a distributed architecture as described herein. The computing device **900** may be any computer system, such as a workstation, desktop computer, server, laptop, handheld computer, tablet computer (e.g., the iPad™ tablet computer), mobile computing or communication device (e.g., the iPhone™ mobile communication device, the Android™ mobile communication device, and the like), or other form of computing or telecommunications device that is capable of communication and that has sufficient processor power and memory capacity to perform the operations described herein. A distributed computational system may be provided comprising a plurality of such computing devices.

[0147] The computing device **900** includes one or more non-transitory computer-readable media having encoded thereon one or more computer-executable instructions or software for implementing exemplary methods described herein. The non-transitory computer-readable media may include, but are not limited to, one or more types of hardware memory, non-transitory tangible media (for example, one or more magnetic storage disks, one or more optical disks, one or more USB flash drives), and the like. For example, memory **906** included in the computing device **900** may store computer-readable and computer-executable instructions or software for implementing exemplary embodiments. The computing device **900** also includes processor **902** and associated core **904**, and in some embodiments, one or more additional processor(s) **902'** and associated core(s) **904'** (for example, in the case of computer systems having multiple processors/cores), for executing computer-readable and computer-executable instructions or software stored in the memory **906** and other programs for controlling system hardware. Processor **902** and processor(s) **902'** may each be a single core processor or multiple core (**904** and **904'**) processor.

[0148] Virtualization may be employed in the computing device **900** so that infrastructure and resources in the computing device may be shared dynamically. A virtual machine **914** may be provided to handle a process running on multiple processors so that the process appears to be using only one computing resource rather than multiple computing resources. Multiple virtual machines may also be used with one processor.

[0149] Memory **906** may include a computer system memory or random access memory, such as DRAM, SRAM, EDO RAM, and the like. Memory **906** may include other types of memory as well, or combinations thereof. Memory **906** may be used to store one or more slates on a temporary basis, for example, in cache.

[0150] A user may interact with the computing device **900** through a visual display device **918**, such as a screen or monitor, that may display one or more interfaces **920** that may be provided in accordance with exemplary embodiments. The visual display device **918** may also display other aspects, elements and/or information or data associated with exemplary embodiments. The computing device **900** may include other I/O devices for receiving input from a user, for example, a keyboard or any suitable multi-point touch interface **908**, a pointing device **910** (e.g., a mouse, a user's finger interfacing directly with a display device, etc.). The keyboard **908** and the pointing device **910** may be coupled to the visual display device **918**. The computing device **900** may include other

suitable conventional I/O peripherals. In exemplary embodiments, the one or more of the interfaces **920** includes an application program interface (API).

[0151] The computing device **900** may include one or more audio input devices **924**, such as one or more microphones, that may be used by a user to provide one or more audio input streams.

[0152] The computing device **900** may include one or more non-transitory storage devices **924**, such as a durable disk storage (which may include any suitable optical or magnetic durable storage device, e.g., RAM, ROM, Flash, USB drive, or other semiconductor-based storage medium), a hard-drive, CD-ROM, or other non-transitory computer readable media, for storing data and computer-readable instructions and/or software that implement exemplary embodiments as taught herein. For example, the storage device **924** may provide a slate storage **926** for storing computer-executable instructions for implementing the social genome data structure as described herein, for example for storing an updating (via one or more updaters) one or more slates, as described herein. The storage device **924** may store one or more map modules **932** and one or more update modules **934**, as described herein. The storage device **924** may be provided on the computing device **900** or provided separately or remotely from the computing device **900**. The storage device **924** may be used to store one or more slates in a durable manner.

[0153] Exemplary mappers and updaters may be programmatically implemented by a computer process in any suitable programming language, for example, a scripting programming language, an object-oriented programming language (e.g., Java), and the like. In an exemplary object-oriented implementation, a general Mapper class or interface and Updater class or interface may be defined by the system to generally specify attributes and functionality of a generic update operation. For each desired update operation, a sub-class may be created based on the Updater class. One or more object instances may be created from each sub-class at a processor node, for example, a CubeTupleGenerator object may be instantiated from a CubeTupleGenerator sub-class.

[0154] The computing device **900** may include a network interface **912** configured to interface via one or more network devices **922** with one or more networks, for example, Local Area Network (LAN), Wide Area Network (WAN) or the Internet through a variety of connections including, but not limited to, standard telephone lines, LAN or WAN links (for example, 802.11, T1, T3, 56 kb, X.25), broadband connections (for example, ISDN, Frame Relay, ATM), wireless connections, controller area network (CAN), or some combination of any or all of the above. The network interface **912** may include a built-in network adapter, network interface card, PCMCIA network card, card bus network adapter, wireless network adapter, USB network adapter, modem or any other device suitable for interfacing the computing device **900** to any type of network capable of communication and performing the operations described herein. The network device **922** may include one or more suitable devices for receiving and transmitting communications over the network including, but not limited to, one or more receivers, one or more transmitters, one or more transceivers, one or more antennae, and the like.

[0155] The computing device **900** may run any operating system **916**, such as any of the versions of the Microsoft® Windows® operating systems, the different releases of the Unix and Linux operating systems, any version of the MacOS® for Macintosh computers, any embedded operating system, any real-time operating system, any open source operating system, any proprietary operating system, any operating systems for mobile computing devices, or any other operating system capable of running on the computing device and performing the operations described herein. In exemplary embodiments, the operating system **916** may be run in native mode or emulated mode. In an exemplary embodiment, the operating system **916** may be run on one or more cloud machine instances.

[0156] FIG. **10** depicts an exemplary network environment **1000** suitable for a distributed implementation of exemplary embodiments. The network environment **1000** may include one or more servers **1002** and **1004** coupled to one or more clients **1006** and **1008** via a communication network **1010**. The network interface **912** and the network device **922** of the computing device **900** enable the servers **1002** and **1004** to communicate with the clients **1006** and **1008** via the communication network **1010**. The communication network **1010** may include, but is not limited to, the Internet, an intranet, a LAN (Local Area Network), a WAN (Wide Area Network), a MAN (Metropolitan Area Network), a wireless network, an optical network, and the like. The communication facilities provided by the communication network **1010** are capable of supporting distributed implementations of exemplary embodiments.

[0157] In an exemplary embodiment, the servers **1002** and **1004** may provide the clients **1006** and **1008** with computer-readable and/or computer-executable components or products under a particular condition, such as a license agreement. In some exemplary embodiments, the computer-readable and/or computer-executable components or products provided by the servers may include those for providing one or more real-time data streams to worker processes at worker nodes. The clients **1006** and **1008** may process the data streams using the computer-readable and/or computer-executable components and products provided by the servers **1002** and **1004**. In some exemplary embodiments, the computer-readable and/or computer-executable components or products provided by the servers may include those for providing and executing one or more map and/or update operations, for example using one or more mappers or updaters. The clients **1006** and **1008** may execute the map and update operations using the computer-readable and/or computer-executable components and products provided by the servers **1002** and **1004**. In some exemplary embodiments, the clients **1006** and **1008** may transmit events generated by update operations to the servers **1002** and **1004** for publication in one or more data streams. In some exemplary embodiments, the clients **1006** and **1008** may transmit one or more slates created or updated by update operations to the servers **1002** and **1004** for persistent storage on a disk storage or for storage in memory, e.g., in cache.

[0158] Alternatively, in another exemplary embodiment, the clients **1006** and **1008** may provide the servers **1002** and **1004** with computer-readable and computer-executable components or products under a particular condition, such as a license agreement. In some exemplary embodiments, the computer-readable and/or computer-executable components or products provided by the clients may include those for providing one or more real-time data streams to worker processes. The servers **1002** and **1006** may process the data streams using the computer-readable and/or computer-executable components and products provided by the clients

1006 and **1008**. In some exemplary embodiments, the computer-readable and/or computer-executable components or products provided by the clients may include those for providing and executing one or more map and/or update operations. The servers **1002** and **1004** may execute the map and update operations using the computer-readable and/or computer-executable components and products provided by the clients **1006** and **1008**. In some exemplary embodiments, the servers **1002** and **1004** may transmit events generated by update operations to the clients **1006** and **1008** for publication in one or more data streams. In some exemplary embodiments, the servers **1002** and **1004** may transmit one or more slates created or updated by update operations to the clients **1006** and **1008** for persistent storage on a disk storage or for storage in memory, e.g., in cache.

[0159] In exemplary embodiments one or more mappers and one or more updaters for example map module **932** and update module **934** of FIG. **9**, may be distributed to throughout various processing nodes of the network environment **1000**, for example nodes **1012***a-d*.

[0160] Although the teachings herein have been described with reference to exemplary embodiments and implementations thereof, the disclosed systems, methods and non-transitory storage medium are not limited to such exemplary embodiments/implementations. Rather, as will be readily apparent to persons skilled in the art from the description taught herein, the disclosed storage media, systems and methods are susceptible to modifications, alterations and enhancements without departing from the spirit or scope hereof. Accordingly, all such modifications, alterations and enhancements within the scope hereof are encompassed herein.

What is claimed:

1. A method for performing real time analytics on streaming data, the method comprising:

processing events in a data stream to extract from each event a set of attribute-value pairs for one or more dimension attributes and one or more value attributes;

identifying one or more tuples in a multidimensional data structure implicated by the extracted attribute-value pairs for the one or more dimension attributes;

for each implicated tuple, updating one or more stored aggregates associated therewith, based on the extracted attribute-value pairs for the one or more value attributes.

2. The method of claim **1** further comprising tracking over an interval a tuple frequency for each of the implicated tuples, wherein the updating the one or more stored aggregates includes discarding each implicated tuple with a low tuple frequency over the interval.

3. The method of claim **1** further comprising tracking over an interval, for each value attribute, aggregates for a first plurality of implicated tuples having same attribute-value pairs for zero or more ordinary dimension attributes and different attribute-value pairs for one or more leaderboard dimension attributes, and determining a top-N values for the one or more leadership dimension attributes over the interval.

4. The method of claim **3**, wherein the Top-N values are characterized as resulting in one of (i) the highest aggregates (ii) the lowest aggregates, and (iii) the aggregates closest to a selected value, over the interval.

5. The method of claim **3** further comprising determining a set of top-N values for each of a plurality of intervals in a time window and determining a top-N values for the one or more leadership dimensions attributes over the time window based on the plurality of sets of top-N values.

6. The method of claim **1**, wherein the one or more dimension attributes include a K-Gram for identifying topics of interest.

7. The method of claim **6**, further comprising tracking over an interval a tuple frequency for each of the implicated tuples including a K-Gram, wherein the updating the one or more stored aggregates includes discarding each implicated tuple with a low tuple frequency over the interval, whereby statistics for trending K-Gram-value pairs are tracked.

8. A method for implementing a real time analytics platform, the method comprising:

establishing an analytics platform framework characterized by one or more time windows, one or more dimension attributes, and one or more value attribute;

generating a first multi-dimensional data structure for maintaining, for each tuple of the one or more dimension attributes, an aggregate of each of the one or more value attributes over each of the one or more time windows.

9. A system for performing real time analytics on streaming data, the system comprising:

a processor for processing an event in a data stream to extract a set of attribute-value pairs for one or more dimension attributes and one or more value attributes;

a mapper for identifying one or more tuples in a multidimensional data structure implicated by the extracted attribute-value pairs for the one or more dimension attributes; and

one or more updaters for updating, for each implicated tuple, one or more stored aggregates associated therewith, based on the extracted attribute-value pairs for the one or more value attributes.

10. The system of claim **9**, wherein the system is configured to track over an interval a tuple frequency for each of the implicated tuples, wherein the updating the one or more stored aggregates includes discarding each implicated tuple with a low tuple frequency over the interval.

11. The system of claim **9** wherein the system is configured to: (i) update, over an interval, for each value attribute, aggregates for a first plurality of implicated tuples having same attribute-value pairs for zero or more ordinary dimension attributes and different attribute-value pairs for one or more leaderboard dimension attributes, and (ii) determine a top-N values for the one or more leadership dimension attributes over the interval.

12. The system of claim **9**, wherein the one or more dimension attributes include a K-Gram for identifying topics of interest, wherein the system is configured to track over an interval a tuple frequency for each of the implicated tuples including a K-Gram, wherein the updating the one or more stored aggregates includes discarding each implicated tuple with a low tuple frequency over the interval, whereby statistics for trending K-Gram-value pairs are tracked.

13. A multi-dimensional data structure for implementing a real-time analytics platform characterized by one or more time windows, one or more dimension attributes, and one or more value attributes, the data structure comprising:

a plurality of tuples associated with the one or more dimension attributes; and

a slate associated with each tuple for maintaining an aggregate for each of the one or more value attributes over each of the one or more time windows.

14. A method for performing real-time analytics on a data stream, the methods comprising:

processing a data stream to maintain a plurality of stored aggregates for a universe of prospective queries extrapolated from a pre-established framework of possible query parameters;

returning one of the stored aggregates in response to a query.

15. A system for performing real-time analytics on a data stream the system comprising:

a processor for processing a data stream to maintain a plurality of stored aggregates for a universe of prospective queries extrapolated from a pre-established framework of possible query parameters; and

memory for storing the plurality of stored aggregates.

16. The system of claim **15**, further comprising an interface for providing a query, wherein the processor is configured to return one of the stored aggregates in response to the query.

17. A multi-dimensional data structure for implementing a real-time analytics platform, the data structure comprising:

a plurality of stored tuples each representing a set of search query parameters for prospective queries extrapolated from a pre-established framework of possible query parameters; and

one or more stored aggregates associated with each of the stored tuples, wherein each aggregate represents a result for a prospective query characterized by the set of search query parameters represented in the tuple associated with that aggregate.

18. A non-transitory computer readable medium storing processor executable instructions for performing real time analytics on streaming data, including instructions for:

processing events in a data stream to extract from each event a set of attribute-value pairs for one or more dimension attributes and one or more value attributes;

identifying one or more tuples in a multidimensional data structure implicated by the extracted attribute-value pairs for the one or more dimension attributes;

for each implicated tuple, updating one or more stored aggregates associated therewith, based on the extracted attribute-value pairs for the one or more value attributes.

19. A non-transitory computer readable medium storing processor executable instructions for performing real time analytics on streaming data, including instructions for:

establishing an analytics platform framework characterized by one or more time windows, one or more dimension attributes, and one or more value attribute;

generating a first multi-dimensional data structure for maintaining, for each tuple of the one or more dimension attributes, an aggregate of each of the one or more value attributes over each of the one or more time windows.

20. A non-transitory computer readable medium storing processor executable instructions for performing real time analytics on streaming data, including instructions for:

processing a data stream to maintain a plurality of stored aggregates for a universe of prospective queries extrapolated from a pre-established framework of possible query parameters; and

returning one of the stored aggregates in response to a query.

* * * * *