(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
3 April 2014 (03.04.2014)

WIPO | PCT

(10) International Publication Number
WO 2014/052675 A2

(51) International Patent Classification:
G06F 17/30 (2006.01)

(21) International Application Number:
PCT/US2013/062047

(22) International Filing Date:
26 September 2013 (26.09.2013)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
61/707,641 28 September 2012 (28.09.2012) US
13/827,987 14 March 2013 (14.03.2013) US

(71) Applicant: **ORACLE INTERNATIONAL CORPORATION** [US/US]; 500 Oracle Parkway, Mail Stop 5OP7, Redwood Shores, California 94065 (US).

(72) Inventors: **HSIAO, Eric**; 604 MacArthur Avenue, San Mateo, California 94402 (US). **SHARMA, Vishal**; P231/13, Map-3, Command Hospital, Old Airport Road, Bangalore, Karnataka 560007 (IN). **SANTOS, Adriano Covello**; 3 Knot Ln, Redwood City, California 94065 (US). **GUPTA, Rahul**; Flat # C303, Samrat Apartment, East End 'D' Main Road, 9th Block, Jayanagar, Bangalore, Karnataka 560069 (IN).

(74) Agents: **BENNETT, Jesse S.** et al.; Kilpatrick Townsend & Stockton LLP, Two Embarcadero Center, Eighth Floor, San Francisco, California 94111 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Published:

— *without international search report and to be republished upon receipt of that report (Rule 48.2(g))*

(54) Title: TACTICAL QUERY TO CONTINUOUS QUERY CONVERSION

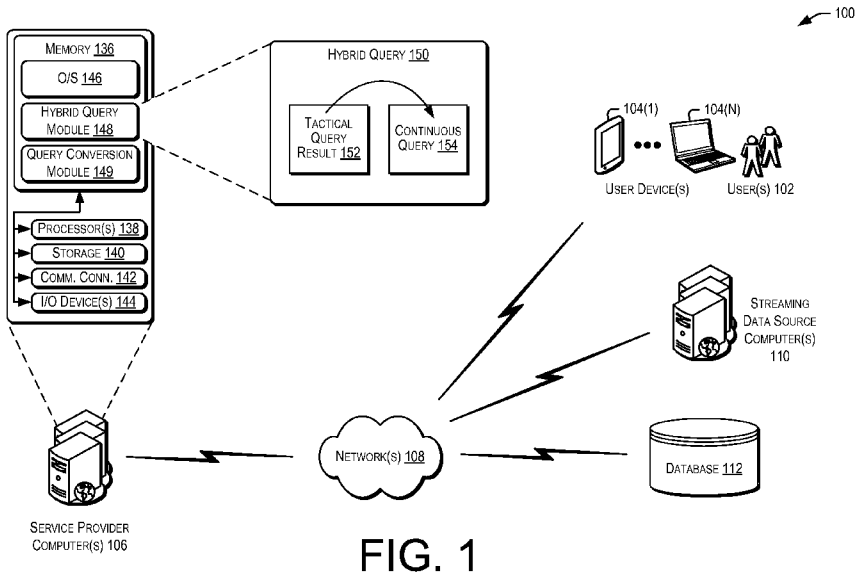

FIG. 1

(57) Abstract: Techniques for managing tactical query to continuous query conversion are provided. In some examples, a tactical query configured to enable the pulling of event data from a database to a query engine may be determined. For example, a query engine may be configured with the tactical query. The configuration may be based at least in part on an indication of the event data to be displayed. Further, in some examples, a conversion of the tactical query to a continuous query may be enabled. The conversion or enablement of the conversion may be based at least in part on a request.

WO 2014/052675 A2

TACTICAL QUERY TO CONTINUOUS QUERY CONVERSION

CROSS REFERENCES TO RELATED APPLICATIONS

[0001] The present application claims priority from and the benefit of U.S. Patent Application No. 13/827,987 (Atty. Docket No. 88325-859818 (128050US)), filed March 14, 2013, entitled "TACTICAL QUERY TO CONTINUOUS QUERY CONVERSION" and of U.S. Provisional Application No. 61/707,641 filed September 28, 2012 entitled "REAL-TIME BUSINESS EVENT ANALYSIS AND MONITORING," the entire contents of each are incorporated herein by reference for all purposes.

BACKGROUND

[0002] Many companies use business intelligence (BI) systems for strategic and tactical decision making where the decision-making cycle may span a time period of several weeks or months. Competitive pressures, however, are forcing companies to react faster to changing conditions and customer requirements. As a result, there is now a desire to use BI to help drive and optimize operations on a daily basis in some cases. Additionally, data associated with a database or streaming data may be stored, managed, and/or processed in many different ways. As noted, some BI data may focus on events. Other BI data may include historical key performance indicator (KPI) information. In some uses cases, utilizing scheduled and/or tactical queries to retrieve and/or process such event data may be beneficial. Additionally, in other examples, utilizing continuous queries to retrieve such event data may make more sense. However, managing the different types of queries as well as memory usage associated with the queries may pose technical challenges to query engines and/or the service providers that implement them.

BRIEF SUMMARY

[0003] Techniques for converting tactical queries to continuous queries are provided. In some examples, a computing system may configure a query engine with a tactical query based at least in part on an indication of event data to be displayed. Additionally, in some examples, the system may enable conversion of the tactical query to a continuous query based at least in part

on a request. The conversion may be enabled at runtime. Additionally, the conversion may include configuring a listening service to receive data pushed from the continuous query. In some examples, the continuous query may be configured to push data from a stream to the listening service associated with the query engine. Additionally, in some aspects, the tactical query may be configured to pull data on behalf of the query engine. The request may be received from a user associated with the data to be displayed. In some cases, the request may be received via a user interface configured to display the data to be displayed to the user. Further, the request may include at least one of a data window, a data range, a filter value, or a dimension change associated with the user.

[0004] Additionally, in some examples, a computer-readable memory or program may be provided. The memory or program may store a plurality of instructions that cause one or more processors to at least determine a tactical query for querying event data of a user from a database. The instructions may also cause the one or more processors to at least convert the tactical query to a continuous query configured to enable pushing of streaming event data of the user to a query engine. Additionally, the instructions may also cause the one or more processors to at least provide a user interface configured to display the active visualization based at least in part on data pushed to the query engine. The tactical query may be converted to the continuous query at runtime of the query engine. In some aspects, the tactical query may be converted to the continuous query based at least in part on a request to activate a visualization of the event data. Additionally, the visualization may be activated only for a user associated with the request to activate the visualization. In some examples, the request to activate the visualization may include at least a request to collapse the event data into one or more objects and update the one or more objects based at least in part on a user-specified time interval. Further, the request to activate the visualization may include at least an indication of a user-specified time window for at least one of displaying or updating the event data.

[0005] Furthermore, in some examples, a method may be provided. The method may be configured to determine a tactical query configured to enable pulling event data from a database to a query engine. The determination may be performed by a computing system. Additionally, in some aspects, the method may be configured to receive a request associated with generating an active visualization of the event data. The request may be received from a user associated with

the event data. Further, the method may be configured to converting, by the computing system at runtime, the tactical query to a continuous query configured to enabling streaming event data of the user to be pushed to the query engine. The method may also be configured to provide a user interface configured to display the active visualization based at least in part on the streaming event data pushed to the query engine. In some aspects, the converting may be based at least in part on the request associated with generating the active visualization of the event data. Additionally, the method may be configured to register a listening system of the query engine to receive the data pushed by the continuous query. The tactical query may be determined based at least in part on a request, from the user, to display the event data. Further, the method may be configured to receive a request to apply a filter to the continuous query or to change a dimension of the continuous query. The request associated with generating the active visualization of the event data may, in some examples, include at least one of a sliding window configuration or a time range configuration specified by the user.

[0006] According to at least one example, a system may include means for configuring, based at least in part on an indication of event data to be displayed, a query engine with a tactical query. The conversion may be enabled at runtime. Additionally, the system may include means for enabling, based at least in part on a request, conversion of the tactical query to a continuous query. Further, in some examples, the conversion may include at least configuring a listening service to receive data pushed from the continuous query. The continuous query may be configured to push data from a stream to the listening service associated with the query engine. Additionally, in some examples, the tactical query may be configured to pull data on behalf of the query engine.

[0007] The foregoing, together with other features and embodiments, will become more apparent upon referring to the following specification, claims, and accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

[0008] The detailed description is set forth with reference to the accompanying figures. In the figures, the left-most digit(s) of a reference number identifies the FIG. in which the reference number first appears. The use of the same reference numbers in different FIGS. indicates similar or identical items.

[0009] FIG. 1 is a simplified block diagram illustrating an example architecture for managing the hybrid execution of continuous and scheduled queries, according to at least one example.

[0010] FIG. 2 is a simplified block diagram illustrating at least some features of the management of the hybrid execution of continuous and scheduled queries described herein, according to at least one example.

[0011] FIG. 3 is a simplified flow diagram illustrating at least some additional features of the management of the hybrid execution of continuous and scheduled queries described herein, according to at least one example.

[0012] FIG. 4 is a simplified flow diagram illustrating at least some additional features of the tactical query to continuous query conversion described herein, according to at least one example.

[0013] FIG. 5 is a simplified process flow illustrating at least some features of the management of the hybrid execution of continuous and scheduled queries described herein, according to at least one example.

[0014] FIG. 6 is another simplified process flow illustrating at least some features of the management of the hybrid execution of continuous and scheduled queries described herein, according to at least one example.

[0015] FIG. 7 is another simplified process flow illustrating at least some features of the management of the hybrid execution of continuous and scheduled queries described herein, according to at least one example.

[0016] FIG. 8 is a simplified process flow illustrating at least some features of the tactical query to continuous query conversion described herein, according to at least one example.

[0017] FIG. 9 is another simplified process flow illustrating at least some features of the tactical query to continuous query conversion described herein, according to at least one example.

[0018] FIG. 10 is another simplified process flow illustrating at least some features of the tactical query to continuous query conversion described herein, according to at least one example.

[0019] FIG. 11 is a simplified block diagram illustrating components of a system environment that may be used in accordance with an embodiment of the management of the hybrid execution of continuous and scheduled queries described herein, according to at least one example.

[0020] FIG. 12 is a simplified block diagram illustrating a computer system that may be used in accordance with embodiments of the management of the hybrid execution of continuous and scheduled queries described herein, according to at least one example.

DETAILED DESCRIPTION

[0021] In the following description, various embodiments will be described. For purposes of explanation, specific configurations and details are set forth in order to provide a thorough understanding of the embodiments. However, it will also be apparent to one skilled in the art that the embodiments may be practiced without the specific details. Furthermore, well-known features may be omitted or simplified in order not to obscure the embodiment being described.

[0022] In some examples, mechanisms to support the combination of continuous query language (CQL) queries with other queries including, but not limited to, tactical queries (e.g., on a timer), strategic queries, relational queries, etc., may be implemented. The present disclosure may also provide the ability to embed structured query language (SQL) queries (e.g., with a timer) into CQL queries in such a way that the SQL queries can be executed on a timer and the result set used within the CQL statement. Additionally, in some examples, mechanisms to support the conversion of tactical queries to continuous queries may be implemented. For example, a query engine may be configured with one or more tactical queries configured to enable pulling data from a database or other data source. In some aspects, the tactical query may be converted to a continuous query at runtime. In this way, the query engine may then be configured to receive data pushed via a stream or other real-time data source. Further, in some examples, a tactical query may be short, highly tuned, and/or facilitate action-taking or decision making in a time-sensitive environment. Additionally, in some examples, a tactical query may

include a clear service level expectation and/or may consume a very small percentage of the overall system resources. Further, tactical queries may be repetitively executed and take advantage of techniques such as, but not limited to, request (query plan) caching and/or session-pooling. In some examples, a tactical may not do a full table scan and/or may be a short, single row query that can execute and return in a very short time.

[0023] A continuous data stream (also referred to as an event stream) may include a stream of data or events that may be continuous or unbounded in nature with no explicit end. Logically, an event or data stream may be a sequence of data elements (also referred to as events), each data element having an associated timestamp. A continuous event stream may be logically represented as a bag or set of elements (s, T), where "s" represents the data portion, and "T" is in the time domain. The "s" portion is generally referred to as a tuple or event. An event stream may thus be a sequence of time-stamped tuples or events.

[0024] In some aspects, the timestamps associated with events in a stream may equate to a clock time. In other examples, however, the time associated with events in an event stream may be defined by the application domain and may not correspond to clock time but may, for example, be represented by sequence numbers instead. Accordingly, the time information associated with an event in an event stream may be represented by a number, a timestamp, or any other information that represents a notion of time. For a system receiving an input event stream, the events arrive at the system in the order of increasing timestamps. There could be more than one event with the same timestamp.

[0025] In some examples, an event in an event stream may represent an occurrence of some worldly event (e.g., when a temperature sensor changed value to a new value, when the price of a stock symbol changed) and the time information associated with the event may indicate when the worldly event represented by the data stream event occurred.

[0026] For events received via an event stream, the time information associated with an event may be used to ensure that the events in the event stream arrive in the order of increasing timestamp values. This may enable events received in the event stream to be ordered based upon their associated time information. In order to enable this ordering, timestamps may be associated with events in an event stream in a non-decreasing manner such that a later-generated event has a

later timestamp than an earlier-generated event. As another example, if sequence numbers are being used as time information, then the sequence number associated with a later-generated event may be greater than the sequence number associated with an earlier-generated event. In some examples, multiple events may be associated with the same timestamp or sequence number, for example, when the worldly events represented by the data stream events occur at the same time. Events belonging to the same event stream may generally be processed in the order imposed on the events by the associated time information, with earlier events being processed prior to later events.

[0027] The time information (e.g., timestamps) associated with an event in an event stream may be set by the source of the stream or alternatively may be set by the system receiving the stream. For example, in certain embodiments, a heartbeat may be maintained on a system receiving an event stream, and the time associated with an event may be based upon a time of arrival of the event at the system as measured by the heartbeat. It is possible for two events in an event stream to have the same time information. It is to be noted that while timestamp ordering requirement is specific to one event stream, events of different streams could be arbitrarily interleaved.

[0028] An event stream has an associated schema "S," the schema comprising time information and a set of one or more named attributes. All events that belong to a particular event stream conform to the schema associated with that particular event stream. Accordingly, for an event stream (s, T), the event stream may have a schema 'S' as (<time_stamp>, <attribute(s)>), where <attributes> represents the data portion of the schema and can comprise one or more attributes. For example, the schema for a stock ticker event stream may comprise attributes <stock symbol>, and <stock price>. Each event received via such a stream will have a time stamp and the two attributes. For example, the stock ticker event stream may receive the following events and associated timestamps:

…

(<timestamp_N>, <NVDA,4>)

(<timestamp_N+1>, <ORCL,62>)

(<timestamp_N+2>, <PCAR,38>)

(<timestamp_N+3>, <SPOT,53>)

(<timestamp_N+4>, <PDCO,44>)

(<timestamp_N+5>, <PTEN,50>)

…

In the above stream, for stream element (<timestamp_N+1>, <ORCL,62>), the event is <ORCL,62> with attributes "stock_symbol" and "stock_value." The timestamp associated with the stream element is "timestamp_N+1". A continuous event stream is thus a flow of events, each event having the same series of attributes.

[0029] In the event monitoring and analytics platforms, there may be a desire to use historical key performance indicators (KPIs) as thresholds within a continuous query. In some examples, the events may be part of business event data (e.g., log information of actions, events, and/or transactions associated with a particular business entity). In some examples, the business event data may include, but is not limited to, records created to record a transaction; the records may, in some cases, include a timestamp. Additionally, business events may also be referred to herein as “operational events.” One example of using such historical KPIs as threshold may include requesting and/or providing a KPI trend in a trend line (e.g., by the hour) along with an average, median, and/or standard deviation (e.g., allowing comparison against a historical timeframe, such as the median value from previous day or the like). In such a case, the historical KPIs may be queried against the database, and the historical KPI may be joined with the results from the function. Alternatively, or in addition, the historical KPI may be shipping to the database with a table in the CQL engine (also referred to herein, as a “CQL engine”). Additionally, in some examples, the historical KPI may be utilized to populate the historical table as a relation in the CQL engine.

[0030] Additionally, in some examples, an abstraction of the historical KPI as a stream may be pushed automatically to the CQL engine based at least in part on a schedule from or associated with a business requirement and may provide easy-to-use language syntax and better resource utilization in terms of computation and memory.

[0031] In some aspects, a similar case may be supported by joining streams/relations with an external table and/or shipping the function to a database. The function shipping to the database,

however, may be requested to occur for every (or a subset of every) input data of the stream. However, in other examples, a custom adapter may be added to poll the table and push it as the stream. However, this approach may include additional java coding outside of the CQL engine.

[0032] Yet, by abstracting the business case of historical KPI within a single query, it may be possible to capture the business case and it may be possible to utilize the business case easily. By having an internal stream that joins to the relation, there is minimum invocation of database queries. Additionally, by encapsulating the stream generation within the single query, a custom adaptor may be avoided.

[0033] In some examples, business intelligence (BI) may help drive and optimize business operations at particular intervals (e.g., on a daily basis in some cases). This type of BI is usually called operational business intelligence, real-time business intelligence, or operational intelligence (OI). Operational Intelligence, in some examples, blurs the line between BI and business activity monitoring (BAM). For example, BI may be focused on periodic queries of historic data. As such, it may have a backward-looking focus. However, BI may also be placed into operational applications, and it may therefore expand from a mere strategic analytical tool into the front lines in business operations. As such, BI systems may also be configured to analyze event streams and compute aggregates in real time.

[0034] In some examples, a continuous query language service (CQ Service) may be configured to extend a BI analytics server to handle continuous queries and enable real-time alerts. The CQ Service, in some aspects, may provide integration with a BI analytics server and a CQL engine. By way of example only, a BI analytics server may delegate continuous queries to the CQ Service and the CQ Service may also act as a logical database (DB) gateway for a CQL engine. In this way, the CQL engine may be able to leverage the BI analytics server for its analytics capabilities and semantic modeling.

[0035] In some examples, the CQ Service may provide, among other things, the following functionalities:

- Remoting service for BI Analytics Server as CQL engine Gateway;
- Event source/sink adapter;

- Generate data definition languages (DDLs) from logical SQL plus CQL extensions;
- Provide unified model for all types of continuous queries and implementation selections;
- Maintain metadata and support restartability; and
- High availability and scalability support.

[0036] Additionally, in some examples, OI is a form of real-time dynamic, business analytics that can deliver visibility and insight into business operations. OI is often linked to or compared with BI or real-time BI, in the sense that both help make sense out of large amounts of information. But there are some basic differences: OI may be primarily activity-centric, whereas BI may be primarily data-centric. Additionally, OI may be more appropriate for detecting and responding to a developing situation (e.g., trend and pattern), unlike BI which may traditionally be used as an after-the-fact and report-based approach to identifying patterns.

[0037] In some examples, a business event analysis and monitoring (BEAM) system may include a CQL engine to process and/or receive in-flight data. For example, a CQL engine may be an in-memory database engine configured to query or otherwise process incoming real-time information (e.g., BI or OI). The CQL engine may utilize or understand temporal semantics and be configured to allow definition of a window of data to process. Utilizing a CQL engine may, in some cases, involve always running a query on incoming data.

[0038] In some aspects, the CQL engine may include a full blown query language. As such, a user may specify computations in terms of a query. Additionally, the CQL engine may be designed for optimizing memory, utilizing query language features, operator sharing, rich pattern matching, rich language constructs, etc. Additionally, in some examples, the CQL engine may process both historical data and streaming data. For example, a user can set a query to send an alert when California sales hit above a certain target. Thus, in some examples, the alert may be based at least in part on historical sales data as well as incoming live (i.e., real-time) sales data.

[0039] In some examples, the CQL engine or other features of the below described concepts may be configured to combine a historical context (i.e., warehouse data) with incoming data in a

real-time fashion. Thus, in some cases, the present disclosure may describe the boundary of database stored information and in-flight information. Both the database stored information and the inflight information may include BI data. As such, the database may, in some examples, be a BI server or it may be any type of database. Further, in some examples, the features of the present disclosure may enable the implementation of the above features without users knowing how to program or otherwise write code. In other words, the features may be provided in a feature-rich user interface (UI) or other manner that allows non-developers to implement the combination of historical data with real-time data.

[0040] Additionally, in some examples, the present disclosure may describe dashboard customization and/or personalization. A CEP engine may be configured to include advanced, continuous analysis of real-time information and historical data. Business process models (BPMs) may include performing model-driven execution of policies and processes defined as BPM notation (BPMN) models. Key result indicators (KRI) may be utilized to tell a user how they have done in a perspective or critical success factor (CSF). For example, it may provide results for many actions, it may cover a longer period of time than key performance indicators (KPIs), and/or it may be reviewed on monthly or quarterly periods. Result indicators (RIs) may be utilized to tell a user what they have done. For example, it may summarize activity, and financial performance measure and/or it may update daily, weekly, or monthly. Further, in some aspects, performance indicators (PIs) may be utilized to inform a user what actions to take or at least make recommendations. Additionally, it may include non-financial information and may, in some cases, complement the KPI.

[0041] In some aspects, PI may be reviewed 24/7, daily, weekly, or less regularly. In some cases, KPI may include a set of measures that are most critical for the current and future success of an organization. Some KPIs may be updated daily or even 24/7 while the rest of the information may be reported weekly. Examples of KPI notifications may include, but are not limited to, whether a plane or other service vehicle was delayed or whether a trailer has been sent out underweight the previous day for a distribution company (e.g., to discover better utilization of the trucks).

[0042] In some examples, embodiments for managing real-time business events may include integrating (e.g., seamlessly) business activity monitoring, complex event processing, and

business intelligence to provide a complex, and real-time set of operational information. Additionally, continuous monitoring of business events may be utilized to gain real-time visibility of business processes and/or workflows. In some examples, OI may be supplemented with traditional business intelligence. As such, operational intelligence may give more insight into business operations versus BI, which, as noted above, is more data centric. For example, OI may get inside to determine how a business is doing in a real-time fashion. Whereas BI may be more akin to data warehousing (e.g., indicating information after the fact).

[0043] Examples of KPI may include real-time call processing time. For example, a user may set real time KPI to be 15 minutes, versus weeks or days. As such, users may be enabled to take actions right away. Further, by coupling historical (data centric) information from BI warehouses with current real-time data, users may be able to view how a business is running in the current state (including continuously updated, streaming data). In some examples, advanced continuous analysis of real-time information may be included in the data processing. Additionally, incremental computations may be performed and included in displays, visualizations, user interfaces (UIs), etc.

[0044] Additionally, in some examples, the present disclosure may be directed towards enabling the conversion of tactical queries to continuous queries. As noted above, the conversion may be enabled at runtime such that a user may request the conversion while viewing static data and/or historical business event data retrieved via one or more tactical queries. In some aspects, a business event analysis and monitoring (BAM) composer may be a web application that provides Design Time at Run Time API (DT@RT) capabilities to are capable of defining dashboards, KPIs, and/or alerts. It may be built, in some examples, using a Service Oriented Architecture (SOA) common console framework or other suitable framework. It may act as a design time tool to build alerts and business dashboards and it may be configured to hide the complexity of CQL, data controls, ADF task flows, interactions with the analytics server, etc., from user. It may also help users create the data objects (semantic layer) requested by business views in dashboards. As such, methods may be implemented for converting from tactical query (e.g., SQL or other language for querying historical data) to a real-time query (e.g., CQL or other type of continuous query language). In some examples, the conversion may include changing

from a data pull model to a data push model. Further, the conversion may also provide the ability to slice and dice the data and/or enable personalization by a user.

[0045] In some examples, one or more dashboards or other user interfaces may be provided that may be configured to allow "slice and dice" functionality of the business views at runtime. Through "slice and dice," functionality, there may be at least three functionalities available to the user including, but not limited to, filters, dimension changes, and/or activation of an active data service (ADS). In the case of ADS, a user can select either a (moving) time based criteria (e.g. the last 10 minutes) or a event based criteria (e.g. the last 100 events) or both and it's supplemented with an output throttle window (e.g. to collapse output to every minute even there could be multiple events within the minute). The output throttle may give a user a choice on how fast to receive these new data. In some aspects, a user may be able to apply additional filters on the tactical and/or continuous queries that are already being implemented or have already begun collecting and/or displaying data. For example, a query that is already saved at design time may have a filter applied to it such that the only particular (e.g., relevant as determined by the user) data may be provided. Additionally, in some aspects, a user may also be able to change the dimension (e.g., when the dimension is selected at design time) at runtime (e.g., while viewing the data on the dashboard). Further, in some examples, a user may be able to view real-time changes happening to the data of the original business view's query. For example, the view may be changed from static to active in order to activate a visualization of the changes occurring on the data. In other words, by enabling the ADS, the system may be configured to convert a tactical query (data delivered once) into a continuous query (continuous, incremental computation with the output being pushed out continuously) to account for real-time data changes. In some examples, each of the above changes may remain implemented for each particular user and may be saved as part of a user personalization (e.g., associated with a user profile or other setting).

[0046] The techniques described above and below may be implemented in a number of ways and in a number of contexts. Several example implementations and contexts are provided with reference to the following figures, as described below in more detail. However, the following implementations and contexts are but a few of many.

[0047] FIG. 1 depicts a simplified example system or architecture 100 in which techniques for managing the hybrid execution of continuous and scheduled queries may be implemented. In

architecture 100, one or more users 102 (e.g., account holders) may utilize user computing devices 104(1)-(N) (collectively, "user devices 104") to access one or more service provider computers 106 via one or more networks 108. In some aspects, the service provider computers 106 may also be in communication with one or more streaming data source computers 110 and/or one or more databases 112 via the networks 108. For example, the users 102 may utilize the service provider computers 106 to access or otherwise manage data of the streaming data source computers 110 and/or the databases 112. The databases 112 may be relational databases, SQL servers, or the like and may, in some examples, manage historical data, event data, relations, archived relations, or the like on behalf of the users 102. Additionally, the databases 112 may receive or otherwise store data provided by the streaming data source computers 110. In some examples, the users 102 may utilize the user devices 104 to interact with the service provider computers 106 by providing queries (also referred to as "query statements") or other requests for data (e.g., historical event data, streaming event data, etc.). Such queries or requests may then be executed by the service provider computers 106 to process data of the databases 112 and/or incoming data from the streaming data source computers 110. Further, in some examples, the streaming data source computers 110 and/or the databases 112 may be part of an integrated, distributed environment associated with the service provider computers 106.

[0048] In some examples, the networks 108 may include any one or a combination of multiple different types of networks, such as cable networks, the Internet, wireless networks, cellular networks, intranet systems, and/or other private and/or public networks. While the illustrated example represents the users 102 accessing the service provider computers 106 over the networks 108, the described techniques may equally apply in instances where the users 102 interact with one or more service provider computers 106 via the one or more user devices 104 over a landline phone, via a kiosk, or in any other manner. It is also noted that the described techniques may apply in other client/server arrangements (e.g., set-top boxes, etc.), as well as in non-client/server arrangements (e.g., locally stored applications, etc.).

[0049] The user devices 104 may be any type of computing device such as, but not limited to, a mobile phone, a smart phone, a personal digital assistant (PDA), a laptop computer, a desktop computer, a thin-client device, a tablet PC, etc. In some examples, the user devices 104 may be in communication with the service provider computers 106 via the networks 108, or via

other network connections. Further, the user devices 104 may also be configured to provide one or more queries or query statements for requesting data of the databases 112 (or other data stores) to be processed.

[0050] In some aspects, the service provider computers 106 may also be any type of computing devices such as, but not limited to, mobile, desktop, thin-client, and/or cloud computing devices, such as servers. In some examples, the service provider computers 106 may be in communication with the user devices 104 via the networks 108, or via other network connections. The service provider computers 106 may include one or more servers, perhaps arranged in a cluster, as a server farm, or as individual servers not associated with one another. These servers may be configured to perform or otherwise host features described herein including, but not limited to, the fast path evaluation of Boolean predicates described herein. Additionally, in some aspects, the service provider computers 106 may be configured as part of an integrated, distributed computing environment that includes the streaming data source computers 110 and/or the databases 112.

[0051] In one illustrative configuration, the service provider computers 106 may include at least one memory 136 and one or more processing units (or processor(s)) 138. The processor(s) 138 may be implemented as appropriate in hardware, computer-executable instructions, firmware, or combinations thereof. Computer-executable instruction or firmware implementations of the processor(s) 138 may include computer-executable or machine-executable instructions written in any suitable programming language to perform the various functions described.

[0052] The memory 136 may store program instructions that are loadable and executable on the processor(s) 138, as well as data generated during the execution of these programs. Depending on the configuration and type of service provider computers 106, the memory 136 may be volatile (such as random access memory (RAM)) and/or non-volatile (such as read-only memory (ROM), flash memory, etc.). The service provider computers 106 or servers may also include additional storage 140, which may include removable storage and/or non-removable storage. The additional storage 140 may include, but is not limited to, magnetic storage, optical disks, and/or tape storage. The disk drives and their associated computer-readable media may provide non-volatile storage of computer-readable instructions, data structures, program

modules, and other data for the computing devices. In some implementations, the memory 136 may include multiple different types of memory, such as static random access memory (SRAM), dynamic random access memory (DRAM), or ROM.

[0053] The memory 136, the additional storage 140, both removable and non-removable, are all examples of computer-readable storage media. For example, computer-readable storage media may include volatile or non-volatile, removable or non-removable media implemented in any method or technology for storage of information such as computer-readable instructions, data structures, program modules, or other data. The memory 136 and the additional storage 140 are all examples of computer storage media.

[0054] The service provider computers 106 may also contain communications connection(s) 142 that allow the identity interface computers 120 to communicate with a stored database, another computing device or server, user terminals, and/or other devices on the networks 108. The service provider computers 106 may also include input/output (I/O) device(s) 144, such as a keyboard, a mouse, a pen, a voice input device, a touch input device, a display, one or more speakers, a printer, etc.

[0055] Turning to the contents of the memory 136 in more detail, the memory 136 may include an operating system 146 and one or more application programs or services for implementing the features disclosed herein including at least a hybrid query module 148 and/or a query conversion module 149. As used herein, modules may refer to programming modules executed by servers or clusters of servers that are part of a service. In this particular context, the modules may be executed by the servers or clusters of servers that are part of the service provider computers 106. In some examples, the hybrid query module 148 may be configured to generate or otherwise provide one or more hybrid queries 150 for combining and/or executing the combination of continuous queries and scheduled database queries. For example, a continuous query (e.g., a stream) may be instantiated, and the results of a database query (e.g., on a timer) may be pushed to or otherwise included in the continuous query. Further, in some examples, a hybrid query 150 may include one or more tactical query results 152 being included in a continuous query 154. In this way, non-continuous queries (e.g., the tactical query that provides the tactical query result 152) may effectively be performed continuously. Additionally,

a few examples of the operations of the hybrid query module 148 and/or the service provider computers 106 are described in greater detail below.

[0056] Additionally, in some examples, the query conversion module 149 may be configured to enable conversion of tactical queries into continuous queries. As noted above, the conversion may be performed at runtime and may enable the users 102 to view (via an interface of the user devices 104) real-time changes to historical data was previously displayed statically (e.g., in a non-active way). In some aspects, the user 102 may be able to make visualizations active at runtime while viewing the static data (e.g., from a tactical query) in a dashboard or other user interface displayed by the user devices 104 and/or provided by the service provider computers 106. The user 102, in some examples, may need to be first authenticated and authorized for access to the dashboard and/or the query conversion module 149 functionality. By default, some dashboard views may come with SQL as the query behind the business views. If a user 102 wants to turn a static view into active one, he or she may be able to utilize the query conversion module 149 (e.g., through selection of an icon or other interface element of the dashboard). The query conversion module 149 may then convert the tactical (i.e., the SQL query in this example) into a continuous query (e.g., a CQL query) and the view may be reloaded within the dashboard to handle active changes. Other functionality (e.g., filtering and/or dimension changes) may also be enabled via the dashboard. Additionally, in some examples, the user 102 may be able to active the ADS at runtime and see the active data in the concerned view (e.g., the dashboard). These changes may be specific to the user 102 and may not affect views of other users.

[0057] In some examples, when the user 102 activates the ADS at runtime, a pageDef file (or other definition file) may be updated to indicate that the active view has been activated. Furthermore, personalization can also be achieved by storing the ADS runtime parameters into pageDef files so that when the user comes in next time, the dashboard can be where he or she left off. Additionally, a UI or other interface may provide a popup that may allow the activation of ADS using a checkbox, drop down box, etc. Internally, the query conversion module 149 may convert the tactical query to a continuous query based at least in part on the user 102 selection and/or the pageDef file. A sliding window and/or range may also be specified via the UI. As noted above, changes in a pageDef file may only be implemented for specific users 102 making the request (e.g., checking the box, etc.). As noted, activating ADS at runtime may allow users

102 to change a tactical query into a continuous query at runtime. In some examples, when a business view is rendered in a dashboard, the user 102 may be able to choose a "Make Active" option from a setting section of the UI displaying the dashboard.

[0058] Once a user 102 requests that the view become active, some options may be enabled that the user 102 can use to setup the active data properties. For example, active data collapsing and/or time window functionality may be offered. The active data collapsing functionality may include, but is not limited to, allowing the user 102 to collapse the data into one or more chunks and update the data within each chunk at once (e.g., in a specified time interval). Additionally, the time window functionality may include, but is not limited to, allowing the user 102 to specify whether the user 102 wants a time window in the data or not. If so, this option may also allow the user 102 to specify the window length and an interval after which the window should be updated. Once the user 102 selects the properties, the user 102 may make a selection (e.g., selecting an "ok" icon or the like) which may setup the data and, after a refresh of the graph, the business view may be made active. In some cases, a managed Java bean, which may be associated with each business view, may take care of populating the data in a popup. It may also handle a popup close event, which may in turn invoke code to make the appropriate changes to the pageDef file of the business view. A data control (DC) layer may then pick the data from the pageDef file and make the view active.

[0059] Additional types of computer storage media (which may also be non-transitory) that may be present in the service provider computers 106 and/or user devices 104 may include, but are not limited to, programmable random access memory (PRAM), SRAM, DRAM, RAM, ROM, electrically erasable programmable read-only memory (EEPROM), flash memory or other memory technology, compact disc read-only memory (CD-ROM), digital versatile discs (DVD) or other optical storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other medium which can be used to store the desired information and which can be accessed by the service provider computers 106 and/or user devices 104. Combinations of any of the above should also be included within the scope of computer-readable media.

[0060] Alternatively, computer-readable communication media may include computer-readable instructions, program modules, or other data transmitted within a data signal, such as a

carrier wave, or other transmission. However, as used herein, computer-readable storage media does not include computer-readable communication media.

[0061] FIG. 2 illustrates a simplified block diagram 200 with which features of the hybrid execution of continuous queries and scheduled queries may be described. As noted above, in some examples, the hybrid query 150 may be executed by the hybrid query module 148 of FIG. 1 and may include one or more tactical query results 152 and/or one or more continuous queries 154. In some examples, the hybrid query 150 may be performed or otherwise managed by a CQL engine 202. The CQL engine 202 may be configured to perform continuous queries (e.g., based at least in part on streaming data, event data, live data, real-time data, etc.). As such, performing the hybrid execution of a tactical query and continuous query may include, but is not limited to, executing the hybrid query 150. In some cases, the hybrid query 150 may be implemented and/or refreshed based at least in part on a schedule or other time interval.

[0062] In some aspects, a tactical query may retrieve data from a database similar to or the same as the database 112 of FIG. 1. The tactical query result 152 (i.e., data from the database 112) may then be included in the continuous query 154 (e.g., as an attribute or other value) that is set up to retrieve or otherwise collect events 204 from an event stream. In some examples, the events may be received from the streaming data source computers 110 of FIG. 1 or from other sources and/or streams. However, in some cases, the database 112 and/or the streaming events 204 may be outside or otherwise not accessible by the CQL engine 202. In this way, the CQL engine 202 may depend and/or rely on the hybrid query 150 for the final results. Additionally, in some examples, a continuous query service (also referred to as a "CQ service") 206 may manage and/or control the CQL engine 202 and/or the hybrid query 150.

[0063] In some examples, as noted above, the hybrid query 150 may be implemented to allow a database query language to be utilized in conjunction with continuous queries 154. For example, an SQL query (i.e., a tactical query that may provide the tactical query result 152) may be utilized to get a result 152 from the database 112. This result 152 may then be utilized in a continuous query 154 to run on streaming data (i.e., the events 204). In other words, the hybrid query 150 may provide the ability to query a stream of event data 204 based at least in part on a database query. Further, the hybrid query 150 may provide the ability to run a continuous query 154 at a periodic basis.

[0064] The hybrid query 150 may also support queries that combine continuous queries with a historical BI logical query (e.g., this may be another example of a tactical query). For example, when historical KPI data is used within a continuous query model, the historical BI logical query may run with a schedule and the result may get passed to the CQL engine 202 as a stream. In some cases, implementation of the hybrid query 150 may involve a syntax that includes both CQL features and historical BI logical query (e.g., on a schedule) features. In some examples, the syntax may include the following examples:

create query <fully qualified query name > as <cql>
with <fully qualified stream name> as <historical bi logical sql> <schedule>

Additionally, the stream name may match with the stream in a "where" clause of the continuous query. The stream name may also be fully qualified. The query may be given in the quoted string. The query may be opaque to the CQ service 206 and the CQ service 206 may not attempt to parse it. Further, a schedule syntax may be the same as the one used in the scheduled tactical query without an "expire" clause.

[0065] In some aspects, the hybrid query 150 may include three parts, the continuous part, the tactical part, and the schedule part. The following hybrid query 150 illustrates one non-limiting example which sends an alert if the average call processing time is bigger than the average call processing time from yesterday where the average call processing time is calculated every midnight.

```
create query CALLCENTER.callcenter4hkpi as
istream(
select avg(callProcessingTime) as measure,
        avg(callProcessingTime) -  max(B.prevDayCPT) as actualDeviation,
        max(B.allowedDeviation) as allowedDeviation from
CALLCENTER.CallCenterFact_DO[range 480 hour on callClosedTime] as A,
        CALLCENTER.CC4HistKPI[rows 1] as B
where callStatus = 'CLOSED'
```

```
)
with CALLCENTER.CC4HistKPI as
"select avg(callProcessingTime) as prevDayCPT,
        stddev(callProcessingTime) as allowedDeviation
from CALLCENTER.CallCenterFact_DO
where callStatus = 'CLOSED' AND
TIMESTAMPDIFF(SQL_TSI_DAY,  callClosedTime, CURRENT_TIMESTAMP)>0 "
refresh on "0:0" every 1 day
```

[0066] With the above hybrid continuous query 150 example, the following operations may be occurring:

1. A stream, CALLCENTER.CC4HistKPI is created using a ddl, 'create stream CALLCENTER.CC4HistKPI(prevDayCPT double, allowedDeviatin double)'.

2. The cql part is registered to the CQL engine as the regular cql.

3. The SQL part runs with a scheduler in the beam product following the given schedule, '0:0 AM every 1 days'.

4. The result from the SQL run is pushed to the CQL engine by the beam product as the stream.

5. The join part in the cql part handles joining between the stream and relation.

[0067] FIG. 3 depicts a simplified flow diagram showing one or more techniques 300 for implementing the hybrid execution of continuous and scheduled queries, according to one example. In FIG. 3, the service provider computers 106 are again shown in communication with the users 102 and/or user devices 104 via the networks 108. Additionally, in some examples, the service provider computers 106 may include or be in communication (e.g., via the networks 108) with one or more event processor computers 302 and/or databases 304. While techniques 300 are shown in FIG. 3 in a particular order (including arbitrary sequence numbers), it should be understood that no particular order is necessary and that one or more steps or parts of the

techniques 300 may be omitted, skipped, and/or reordered. In at least one non-limiting example, the one or more service provider computers 106 described above with reference to FIGS. 1 and 2 may receive queries and/or data processing requests (e.g., KPI and/or BI data) from the user devices 104. The queries and/or data requests may be configured to request processing (e.g., retrieval, storage, deletion, etc.) of database data (e.g., data stored by the databases 304) and/or streaming event data (e.g., data being received in real-time from the event processors 302). Additionally, in some examples, the service provider computers 106 may also generate a stream and/or register the stream with a continuous query engine (e.g., but not limited to, the CQL engine 202 of FIG. 2), a service (e.g., but not limited to, the CQ service 206 of FIG. 2), and/or the event processors 302. The stream may be configured to retrieve or otherwise collect real-time KPI data from the event processors 302.

[0068] In some examples, the service provider computers 106 may also provide or implement one or more tactical queries (e.g., an SQL query or the like) to retrieve historical data from the database 304 or another storage system. In some examples, the query may be provided or otherwise implemented based at least in part on a schedule. Based at least in part on the tactical query, the service provider computers 106 may receive data from the database 304. In other words, the service provider computers 106 may receive the query results. The service provider computers 106 may then include the tactical query result in the stream such that the continuous query may be processed based at least in part on that tactical query result. The service provider computers 106 may then receive data from the stream (i.e., the continuous query result) and provide the data and/or acknowledgement of processing to the user computers 104.

[0069] FIG. 4 illustrates a simplified block diagram 400 with which features of the tactical query to continuous query conversion may be described. As noted above, in some examples, an active data service (ADS) 402 may be executed by the query conversion module 149 of FIG. 1 and may include one or more programming layers (also referred to as modules). The implementation of these features (i.e., the layers and/or the options noted above), may be managed by the DC layer 404. In some examples, the DC layer 404 maybe a binding component of an application development framework (ADF) and may be authored in a JDev Environment. The DC layer 404 may be created at DT@RT and may be equipped with a mechanism to change queries dynamically and re-construct the binding definition on the fly to support run time slice

and dice. Furthermore, in some examples, if the dimension or the filter have been changed, these changes may have been sent to DC and, in turn, DC may prepare a new model which may include the changes and then may send the request to the backend for execution. After the results are returned, DC may bind the new model data which may include real time visualization changes that entail both x-axis (usually the time dimension) and the data series (which may be changed by the new filter), it may also be the conduit to convert a tactical query to a continuous query. In some cases, personalization may be applied so each instance of the dashboard can support its own slice and dice.

[0070] The DC layer 404 may be configured to pick up the values form the pageDef of the business view. The DC layer 404 may also get access of the pageDef of the business view to which it is bound to through the ADF context. It may then make changes to an in-memory modifier 405 and set it up to the next later to fetch the data and return it to the business view. The ADS 402 may also include a UI layer 406, a Meta Data Service 408, a Report Cache layer 410, and/or a Common Query layer 412. The UI layer 406 may provide options to the user 102 for setting up new filters. The UI layer 406 may also capture the information provided by the user 102 and set it up in the metadata files in an appropriate format that can be understood by the next layer. The Meta Data Service layer 408 may be configured to allow the update of the underlying pageDef of the business views to be updated and the values captured from the UI layer 406. Once these changes are made, a request may be ordered which may allow this information to be passed to the next layer for processing. In some examples, the Report Cache layer 410 may listen to the DC layer 404 to get the appropriate modifiers and returns back the actual data 414 after fetching it using the Common Query layer 412.

[0071] In some examples, a user 102 may input data 414 to the UI layer 406. This data 414 may be properly structured into information that is requested to be passed on the information to the Data Control layer 404 through the pageDef files of business view. The DC layer 404, through the ADF context, may get the handle of the pageDef to which it is bound and, and may also get the requested information. For applying filters, a Filter node may be added to the pageDef file which depicts the information gathered from the user. A sample follows:

```
<Filter xmlns="http://www.beam.com/datacontrol/personalization">
    <Branch name="Root" type="ALL">
```

```
<Branch type="ANY">
  <Entry type="EQ">
    <Node type="COLUMN">_COMPONENTNAME</Node>
    <Node type="STRING">ABC</Node>
  </Entry>
                <Entry type="EQ">
    <Node type="COLUMN">_DEPARTMENT</Node>
    <Node type="STRING">xyz</Node>
  </Entry>
</Branch>
<Branch type="ANY">
    <Entry type="EQ">
      <Node type="COLUMN">_DEPARTMENT</Node>
      <Node type="STRING">xyz</Node>
    </Entry>
</Branch>
</Branch>
</Filter>
```

[0072] In some examples, this node may not be part of the pageDef of a view in ADF. Thus, in order for it to fit into an extensible markup language (XML) file, the user of a custom namespace may be implemented (e.g., highlighted in the node above). This node may then be picked up by the DC layer 404 and set in the in-memory modifier 405 to depict the changes done by the user 102 on top of the modifier that already existed for that query in the business view. When this modifier gets sent to the report cache layer 410 to return the actual data 414 back to the DC layer 404, which may in turn return the data to the UI layer 406.

[0073] Similarly, for making group changes, and making views active, the XML of a pageDef may be modified with values of new groups and a "ChangeEventPolicy" attribute may be set to "push," which may look as follows:

```
<graph IterBinding="QueryIterator" id="Query" xmlns="http://xmlns.oracle.com/adfm/dvt"
type="AREA_VERT_ABS" ChangeEventPolicy="push">
```

```
<graphDataMap convert="false" leafOnly="true">
    <groups>
        <item value="_COUNTRY"/>
    </groups>
    <series>
        <data>
            <item value="SUM_SALARY"/>
        </data>
    <item value="_DEPARTMENT"/>
    </series>
</graphDataMap>
</graph>
```

[0074] However, this node may not be a part of the ADF framework either. As such, in some examples, it may be used as it is with changes in the group's node to depict the change in the groups done by the user 102 in the UI layer 406. Then a similar approach may be followed to return the data 414 to the UI layer 406 from the DC layer 404. Additionally, in some examples, setting up the active data properties may include the following XML node used in the pageDef file:

```
<ActiveDataProperties xmlns="http://www.beam.com/datacontrol/activeData">
    <ActiveDataCollapse enabled="true" interval="56" timeunit="seconds"/>
    <ActiveDataWindow enabled="true" type="Fixed">
    <RangeLength timeunit="seconds">45</RangeLength>
    <UpdateInterval timeunit="seconds">34</UpdateInterval>
    </ActiveDataWindow>
</ActiveDataProperties>
```

[0075] Again, this node may not be part of the ADF framework. As such, a custom namespace may be utilized. Additionally, when the slice and dice feature changes the groups it may be internally changing the shape of the query. The structure definition inside the DC layer 404 may be based at least in part on the shape of the query. At the time the DC layer 404 gets

access to the pageDef, the structure definition may already be created to the original query, so the DC layer 404 may request to modify the query definition in memory to match the new groups and also recreate dynamically the new structure definition that is expected to be bound to the pageDef that will be accessing/using this instance of the DC layer 404. Then, when the DC layer 404 opens a view on the RC layer 410, it may also use the new query definition that was modified in at the in-memory modifier 405. In some examples, this may ensure that the layers of the ADS 402 will be in sync.

[0076] In some examples, when a filter is applied it may change the query definition but not the shape of the query, consequently the structure definition may not change. The filter may change only the query definition in memory where a "where clause" and "parameters" of the query may change. This may get propagated to the RC layer 410 as well. If the DC layer 404 already has a view open with the previous query definition it may request the RC layer 410 to close the old view with the old query definition and open a new view on the RC layer 410 with the new query definition. Additionally, when the attribute "ChangeEventPolicy" is set to "push," the DC layer 404 may read the ActiveDataProperties node to create an appropriate ViewSetBuilder with the window extension values that is expected by the RC layer 410 when an active data push view is created. At this point the DC layer 404 may close the old tactical view on the RC layer 410 and may create a new active data definition to open a new view on the RC layer 410. After opening the new view on the RC layer 410, the DC layer 404 may register an active data listener onto the RC layer 410 for receiving delta changes. Once the RC layer 410 starts to push data in to the DC listener, the DC layer 404 may start to push data to ADF layer as well.

[0077] Further, personalization features may be implemented using a Meta Data Service (MDS) framework. A filter may be setup in a web.xml file of the application before the adfBindings filter. This filter may set up the SessionOptionsFactory which, on every page request (or a subset of the page requests), returns the customization layer which is currently active at that moment. Also at any moment a site layer may be active, but for the personalization feature the user layer may be dynamically made active. For this, an option(ModeContext) may be maintained which tells the SessionOptionsFactory that the user layer may be made active or not. Once the proper layers in the MDS framework are made active then the changes related to filters,

groups and active data may be made to the pageDef files of business views which are specific for each user and whenever that pageDef is requested then the right one may be returned to the UI layer 406 and also to DC layer 404 to allow this personalization feature to work.

[0078] FIGS. 5-7 illustrate example flow diagrams showing respective processes 500, 600, and 700 for implementing the hybrid execution of continuous and scheduled queries described herein. These processes 500, 600, 700 are illustrated as logical flow diagrams, each operation of which represents a sequence of operations that can be implemented in hardware, computer instructions, or a combination thereof. In the context of computer instructions, the operations represent computer-executable instructions stored on one or more computer-readable storage media that, when executed by one or more processors, perform the recited operations. Generally, computer-executable instructions include routines, programs, objects, components, data structures and the like that perform particular functions or implement particular data types. The order in which the operations are described is not intended to be construed as a limitation, and any number of the described operations can be combined in any order and/or in parallel to implement the processes.

[0079] Additionally, some, any, or all of the processes may be performed under the control of one or more computer systems configured with executable instructions and may be implemented as code (e.g., executable instructions, one or more computer programs, or one or more applications) executing collectively on one or more processors, by hardware, or combinations thereof. As noted above, the code may be stored on a computer-readable storage medium, for example, in the form of a computer program comprising a plurality of instructions executable by one or more processors. The computer-readable storage medium may be non-transitory.

[0080] In some examples, the one or more service provider computers 106 (e.g., utilizing at least one of the hybrid query module 148 of FIG. 1 and/or the continuous query service 206 of FIG. 2) shown in FIGS. 1-3 may perform the process 500 of FIG. 5. The process 500 may begin by including initializing a query engine with relational data from a first source at 502. In some examples, the first source may be a database (e.g., a relational database). At 504, the process 500 may end by enabling the query engine to provide query results based at least in part on the relational data and streaming data from a second source. The second source may be an event

processor or other computing system capable of providing real-time and/or streaming data (e.g., complex events, KPI data, etc.).

[0081] FIG. 6 illustrates an example flow diagram showing process 600 for implementing the hybrid execution of continuous and scheduled queries described herein. The one or more service provider computers 106 (e.g., utilizing at least one of the hybrid query module 148 of FIG. 1 and/or the continuous query service 206 of FIG. 2) shown in FIGS. 1-3 may perform the process 600 of FIG. 6. The process 600 may begin at 602 by including generating a stream associated with a first source at 602. In some examples, the first source may be configured as a complex event processing engine or other event stream. At 604, the process 600 may include receiving a first query result from a second source. The second source may be configured as a database or other storage system (e.g., a relational database or the like). As such, the first query result may be received based at least in part on implementation of a tactical query. At 606, the process 600 may also include including (or providing) the first query result from the second source into the stream generated at 602. The process 600 may then end, at 608, by including providing a second query result based at least in part on the generated stream and the first query result. In other words, the second query result may be based on the inclusion of the first query result into the stream. This second query result may be provided to a user (e.g., via a user interface), to a processing engine, and/or to a service provider configured to monitor, alert, and/or provide information associated with business events or the like.

[0082] FIG. 7 illustrates an example flow diagram showing process 700 for implementing the hybrid execution of continuous and scheduled queries described herein. The one or more service provider computers 106 (e.g., utilizing at least one of the hybrid query module 148 of FIG. 1 and/or the continuous query service 206 of FIG. 2) shown in FIGS. 1-3 may perform the process 700 of FIG. 7. The process 700 may begin by including generating a stream associated with an event processor (e.g., a complex event processor or the like) at 702. At 704, the process 700 may include registering the generated stream with a query engine (e.g., the CQL engine 702 of FIG. 2). The process 700 may also include providing a first query based at least in part on a schedule to a data source at 706. The data source may be a relational database or other storage system capable of providing historical data in response to queries. At 708, the process 700 may include receiving a first query result from the data source based at least in part on the first query provided. Additionally, in some examples, the process 700 may include including the first query

result from the data source in the registered stream at 710. At 712, the process 700 may end by including providing a second query result based at least in part on the registered stream and the first query result. In other words, and as noted above, the second query result may be based at least in part on a combination of the scheduled tactical query and the continuous query (i.e., instantiated by a stream).

[0083] FIGS. 8-10 illustrate example flow diagrams showing respective processes 800, 900, and 1000 for implementing the tactical query to continuous query conversion described herein. These processes 800, 900, 1000 are illustrated as logical flow diagrams, each operation of which represents a sequence of operations that can be implemented in hardware, computer instructions, or a combination thereof. In the context of computer instructions, the operations represent computer-executable instructions stored on one or more computer-readable storage media that, when executed by one or more processors, perform the recited operations. Generally, computer-executable instructions include routines, programs, objects, components, data structures and the like that perform particular functions or implement particular data types. The order in which the operations are described is not intended to be construed as a limitation, and any number of the described operations can be combined in any order and/or in parallel to implement the processes.

[0084] Additionally, some, any, or all of the processes may be performed under the control of one or more computer systems configured with executable instructions and may be implemented as code (e.g., executable instructions, one or more computer programs, or one or more applications) executing collectively on one or more processors, by hardware, or combinations thereof. As noted above, the code may be stored on a computer-readable storage medium, for example, in the form of a computer program comprising a plurality of instructions executable by one or more processors. The computer-readable storage medium may be non-transitory.

[0085] In some examples, the one or more service provider computers 106 (e.g., utilizing at least one of the query conversion module 149 of FIG. 1 and/or the active data service 402 of FIG. 4) shown in FIGS. 1-4 may perform the process 800 of FIG. 8. The process 800 may begin by including configuring a query engine with a tactical query at 802. In some examples, the tactical query may be configured to pull data on behalf of the query engine. In other examples, the tactical query may be configured to enabling receipt of pulled data. At 804, the process 800 may end by including enabling conversion of a tactical query to a continuous query. In some

examples, the conversion may be enabled at runtime. Additionally, in some aspects, the conversion may include at least configuring a listening service to receive data pushed from the continuous query or by the continuous query. Further, the continuous query may, instead, configure the data to be pushed from a stream to the listening service and/or the query engine.

[0086] FIG. 9 illustrates an example flow diagram showing process 900 for implementing the tactical query to continuous query conversion described herein. The one or more service provider computers 106 (e.g., utilizing at least one of the query conversion module 149 of FIG. 1 and/or the active data service 402 of FIG. 4) shown in FIGS. 1-4 may perform the process 900 of FIG. 9. The process 900 may begin at 902 by including determining a tactical query for querying business event data of a user from a database. At 904, the process 900 may include converting the tactical query to a continuous query configured to enable pushing of streaming business event data of the user to a query engine. Further, at 906, the process 900 may end by including providing a user interface configured to display the active visualization based at least in part on data pushed to the query engine.

[0087] FIG. 10 illustrates an example flow diagram showing process 1000 for implementing the tactical query to continuous query conversion described herein. The one or more service provider computers 106 (e.g., utilizing at least one of the query conversion module 149 of FIG. 1 and/or the active data service 402 of FIG. 4) shown in FIGS. 1-4 may perform the process 1000 of FIG. 10. The process 1000 may begin by including determining a tactical query configured to enable pulling business event data from a database to a query engine at 1002. The determination may be based at least in part on a query received from a user. At 1004, the process 1000 may include receiving a request associated with generating an active visualization of the data. The process 1000 may also convert the tactical query into a continuous query at 1006. The conversion may take place at runtime. At 1008, the process 1000 may include registering a listening service to receive pushed data (e.g., from a stream). At 1010, the process 1000 may receive a request to apply a filter to or change dimension of the continuous query (e.g., at runtime). Further, the process 1000 may end at 1012, where the process 1000 may include providing a UI configured to display the active visualization of the pushed data.

[0088] Illustrative methods and systems for implementing the hybrid execution of continuous and scheduled queries are described above. Some or all of these systems and methods may, but

need not, be implemented at least partially by architectures and processes such as those shown at least in FIGS. 1-10 above.

[0089] FIG. 11 is a simplified block diagram illustrating components of a system environment 1100 that may be used in accordance with an embodiment of the present disclosure. As shown, system environment 1100 includes one or more client computing devices 1102, 1104, 1106, 1108, which are configured to operate a client application such as a web browser, proprietary client (e.g., Oracle Forms), or the like over one or more networks 1110 (such as, but not limited to, networks similar to the networks 108 of FIGS. 1 and 3). In various embodiments, client computing devices 1102, 1104, 1106, and 1108 may interact with a server 1112 over the networks 1110.

[0090] Client computing devices 1102, 1104, 1106, 1108 may be general purpose personal computers (including, by way of example, personal computers and/or laptop computers running various versions of Microsoft Windows and/or Apple Macintosh operating systems), cell phones or PDAs (running software such as Microsoft Windows Mobile and being Internet, e-mail, SMS, Blackberry, or other communication protocol enabled), and/or workstation computers running any of a variety of commercially-available UNIX or UNIX-like operating systems (including without limitation the variety of GNU/Linux operating systems). Alternatively, client computing devices 1102, 1104, 1106, and 1108 may be any other electronic device, such as a thin-client computer, Internet-enabled gaming system, and/or personal messaging device, capable of communicating over a network (e.g., network 1110 described below). Although exemplary system environment 1100 is shown with four client computing devices, any number of client computing devices may be supported. Other devices such as devices with sensors, etc. may interact with server 1112.

[0091] System environment 1100 may include networks 1110. Networks 1110 may be any type of network familiar to those skilled in the art that can support data communications using any of a variety of commercially-available protocols, including without limitation TCP/IP, SNA, IPX, AppleTalk, and the like. Merely by way of example, network 1110 can be a local area network (LAN), such as an Ethernet network, a Token-Ring network and/or the like; a wide-area network; a virtual network, including without limitation a virtual private network (VPN); the Internet; an intranet; an extranet; a public switched telephone network (PSTN); an infra-red

network; a wireless network (e.g., a network operating under any of the IEEE 802.11 suite of protocols, the Bluetooth protocol known in the art, and/or any other wireless protocol); and/or any combination of these and/or other networks.

[0092] System environment 1100 also includes one or more server computers 1112 which may be general purpose computers, specialized server computers (including, by way of example, PC servers, UNIX servers, mid-range servers, mainframe computers, rack-mounted servers, etc.), server farms, server clusters, or any other appropriate arrangement and/or combination. In various embodiments, server 1112 may be adapted to run one or more services or software applications described in the foregoing disclosure. For example, server 1112 may correspond to a server for performing processing described above according to an embodiment of the present disclosure.

[0093] Server 1112 may run an operating system including any of those discussed above, as well as any commercially available server operating system. Server 1112 may also run any of a variety of additional server applications and/or mid-tier applications, including HTTP servers, FTP servers, CGI servers, Java servers, database servers, and the like. Exemplary database servers include without limitation those commercially available from Oracle, Microsoft, Sybase, IBM and the like.

[0094] System environment 1100 may also include one or more databases 1114, 1116. Databases 1114, 1116 may reside in a variety of locations. By way of example, one or more of databases 1114, 1116 may reside on a non-transitory storage medium local to (and/or resident in) server 1112. Alternatively, databases 1114, 1116 may be remote from server 1112, and in communication with server 1112 via a network-based or dedicated connection. In one set of embodiments, databases 1114, 1116 may reside in a storage-area network (SAN) familiar to those skilled in the art. Similarly, any necessary files for performing the functions attributed to server 1112 may be stored locally on server 1112 and/or remotely, as appropriate. In one set of embodiments, databases 1114, 1116 may include relational databases, such as databases provided by Oracle, that are adapted to store, update, and retrieve data in response to SQL-formatted commands.

[0095] FIG. 12 is a simplified block diagram of a computer system 1200 that may be used in accordance with embodiments of the present disclosure. For example service provider computers 106 may be implemented using a system such as system 1200. Computer system 1200 is shown comprising hardware elements that may be electrically and/or communicatively coupled via a bus 1201. The hardware elements may include one or more central processing units (CPUs) 1202, one or more input devices 1204 (e.g., a mouse, a keyboard, etc.), and one or more output devices 1206 (e.g., a display device, a printer, etc.). Computer system 1200 may also include one or more storage devices 1208. By way of example, the storage device(s) 1208 may include devices such as disk drives, optical storage devices, and solid-state storage devices such as a random access memory (RAM) and/or a read-only memory (ROM), which can be programmable, flash-updateable and/or the like.

[0096] Computer system 1200 may additionally include a computer-readable storage media reader 1212, a communications subsystem 1214 (e.g., a modem, a network card (wireless or wired), an infra-red communication device, etc.), and working memory 1218, which may include RAM and ROM devices as described above. In some embodiments, computer system 1200 may also include a processing acceleration unit 1216, which can include a digital signal processor (DSP), a special-purpose processor, and/or the like.

[0097] Computer-readable storage media reader 1212 can further be connected to a computer-readable storage medium 1210, together (and, optionally, in combination with storage device(s) 1208) comprehensively representing remote, local, fixed, and/or removable storage devices plus storage media for temporarily and/or more permanently containing computer-readable information. Communications system 1214 may permit data to be exchanged with network 1212 and/or any other computer described above with respect to system environment 1200.

[0098] Computer system 1200 may also comprise software elements, shown as being currently located within working memory 1218, including an operating system 1220 and/or other code 1222, such as an application program (which may be a client application, Web browser, mid-tier application, RDBMS, etc.). In an exemplary embodiment, working memory 1218 may include executable code and associated data structures used for relying party and open authorization-related processing as described above. It should be appreciated that alternative

embodiments of computer system 1200 may have numerous variations from that described above. For example, customized hardware might also be used and/or particular elements might be implemented in hardware, software (including portable software, such as applets), or both. Further, connection to other computing devices such as network input/output devices may be employed.

[0099] Storage media and computer readable media for containing code, or portions of code, can include any appropriate media known or used in the art, including storage media and communication media, such as but not limited to, volatile and non-volatile (non-transitory), removable and non-removable media implemented in any method or technology for storage and/or transmission of information such as computer readable instructions, data structures, program modules, or other data, including RAM, ROM, EEPROM, flash memory or other memory technology, CD-ROM, digital versatile disk (DVD) or other optical storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, data signals, data transmissions, or any other medium which can be used to store or transmit the desired information and which can be accessed by a computer.

[00100] Although specific embodiments of the disclosure have been described, various modifications, alterations, alternative constructions, and equivalents are also encompassed within the scope of the disclosure. The various modifications, alternations, alternative constructions, and equivalents include relevant combinations of the disclosed features. Embodiments of the present disclosure are not restricted to operation within certain specific data processing environments, but are free to operate within a plurality of data processing environments. Additionally, although embodiments of the present disclosure have been described using a particular series of transactions and steps, it should be apparent to those skilled in the art that the scope of the present disclosure is not limited to the described series of transactions and steps.

[00101] Further, while embodiments of the present disclosure have been described using a particular combination of hardware and software, it should be recognized that other combinations of hardware and software are also within the scope of the present disclosure. Embodiments of the present disclosure may be implemented only in hardware, or only in software, or using combinations thereof.

[00102] The specification and drawings are, accordingly, to be regarded in an illustrative rather than a restrictive sense. It will, however, be evident that additions, subtractions, deletions, and other modifications and changes may be made thereunto without departing from the broader spirit and scope. Illustrative methods and systems for providing features of the present disclosure are described above. Some or all of these systems and methods may, but need not, be implemented at least partially by architectures such as those shown in FIGS. 1-12 above.

[00103] In one implementation, there is provided a method, comprising: configuring, based at least in part on an indication of business event data to be displayed, a query engine with a tactical query; and enabling, based at least in part on a request, conversion of the tactical query to a continuous query.

[00104] In another implementation, there is provided a method comprising: determining a tactical query for querying business event data of a user from a database; converting the tactical query to a continuous query configured to enable pushing of streaming business event data of the user to a query engine; and providing a user interface configured to display the active visualization based at least in part on data pushed to the query engine.

[00105] In yet another implementation, there is provided an apparatus configured to perform any of the previous described method. The apparatus can be implemented in a variety of manners, such as firmware, software, hardware, any combination thereof, etc.

[00106] In yet another implementation, there is provided an apparatus, comprising: query conversion module, configured to convert a tactical query a tactical query for querying business event data of a user from a database into a continuous query, and hybrid query module, configured to combine continuous queries and tactical queries. The apparatus can be implemented in a variety of manners, such as firmware, software, hardware, any combination thereof, etc.

[00107] Although embodiments have been described in language specific to structural features and/or methodological acts, it is to be understood that the disclosure is not necessarily limited to the specific features or acts described. Rather, the specific features and acts are disclosed as illustrative forms of implementing the embodiments. Conditional language, such as, among others, "can," "could," "might," or "may," unless specifically stated otherwise, or otherwise

understood within the context as used, is generally intended to convey that certain embodiments could include, while other embodiments do not include, certain features, elements, and/or steps. Thus, such conditional language is not generally intended to imply that features, elements, and/or steps are in any way required for one or more embodiments or that one or more embodiments necessarily include logic for deciding, with or without user input or prompting, whether these features, elements, and/or steps are included or are to be performed in any particular embodiment.

CLAIMS

WHAT IS CLAIMED IS:

1. A system, comprising:

a memory storing a plurality of instructions; and

one or more processors configured to access the memory, wherein the one or more processors are further configured to execute the plurality of instructions to at least:

configure, based at least in part on an indication of event data to be displayed, a query engine with a tactical query;

enable, based at least in part on a request, conversion of the tactical query to a continuous query.

2. The system of claim 1, wherein the conversion is enabled at runtime.

3. The system of claim 1 or 2, wherein the conversion includes at least configuring a listening service to receive data pushed from the continuous query.

4. The system of claim 3, wherein the continuous query is configured to push data from a stream to the listening service associated with the query engine.

5. The system of any one of claims 1 to 4, wherein the tactical query is configured to pull data on behalf of the query engine.

6. The system of any one of claims 1 to 5, wherein the request is received from a user associated with the data to be displayed.

7. The system of claim 6, wherein the request is received via a user interface configured to display the data to be displayed to the user.

8. The system of claim 6, wherein the request includes at least one of a data window, a data range, a filter value, or a dimension change associated with the user.

9. A computer-readable program storing a plurality of instructions executable by one or more processors, the plurality of instructions comprising:

instructions that cause the one or more processors to determine a tactical query for querying event data of a user from a database;

instructions that cause the one or more processors to convert the tactical query to a continuous query configured to enable pushing of streaming event data of the user to a query engine; and

instructions that cause the one or more processors to provide a user interface configured to display an active visualization based at least in part on data pushed to the query engine.

10. The computer-readable program of claim 9, wherein the tactical query is converted to the continuous query at runtime of the query engine.

11. The computer-readable program of claim 9 or 10, wherein the tactical query is converted to the continuous query based at least in part on a request to activate a visualization of the event data.

12. The computer-readable program of claim 11, wherein the visualization is activated only for a user associated with the request to activate the active visualization.

13. The computer-readable program of claim 11, wherein the request to activate the active visualization includes at least a request to collapse the event data into one or more objects and update the one or more objects based at least in part on a user-specified time interval.

14. The computer-readable program of claim 11, wherein the request to activate the visualization includes at least an indication of a user-specified time window for at least one of displaying or updating the event data.

15. A method, comprising:

determining a tactical query configured to enable pulling event data from a database to a query engine;

receiving a request, from a user associated with the event data, associated with generating an active visualization of the event data;

converting the tactical query to a continuous query configured to enabling streaming event data of the user to be pushed to the query engine; and

providing a user interface configured to display an active visualization based at least in part on the streaming event data pushed to the query engine.

16. The method of claim 15, wherein the converting is based at least in part on the request associated with generating the active visualization of the event data.

17. The method of claim 15 or 16, further comprising registering a listening system of the query engine to receive the data pushed by the continuous query.

18. The method of any one of claims 15 to 17, wherein the tactical query is determined based at least in part on a request, from the user, to display the event data.

19. The method of any one of claims 15 to 18, further comprising receiving a request to apply a filter to the continuous query or to change a dimension of the continuous query.

20. The method of any one of claims 15 to 19, wherein the request associated with generating the active visualization of the event data includes at least one of a sliding window configuration or a time range configuration specified by the user.

21. A method, comprising:

configuring, based at least in part on an indication of event data to be displayed, a query engine with a tactical query;

enabling, based at least in part on a request, conversion of the tactical query to a continuous query.

22. A method, comprising:

determining a tactical query for querying event data of a user from a database;

converting the tactical query to a continuous query configured to enable pushing of streaming event data of the user to a query engine; and

providing a user interface configured to display an active visualization based at least in part on data pushed to the query engine.

23. The method of claim 22, wherein the tactical query is configured to query only a single element of the database.

24. An apparatus, comprising:

query conversion module, configured to convert a tactical query a tactical query for querying event data of a user from a database into a continuous query, and

hybrid query module, configured to combine continuous queries and tactical queries.

25. An apparatus configured to perform the method according to any of claims 15-20.

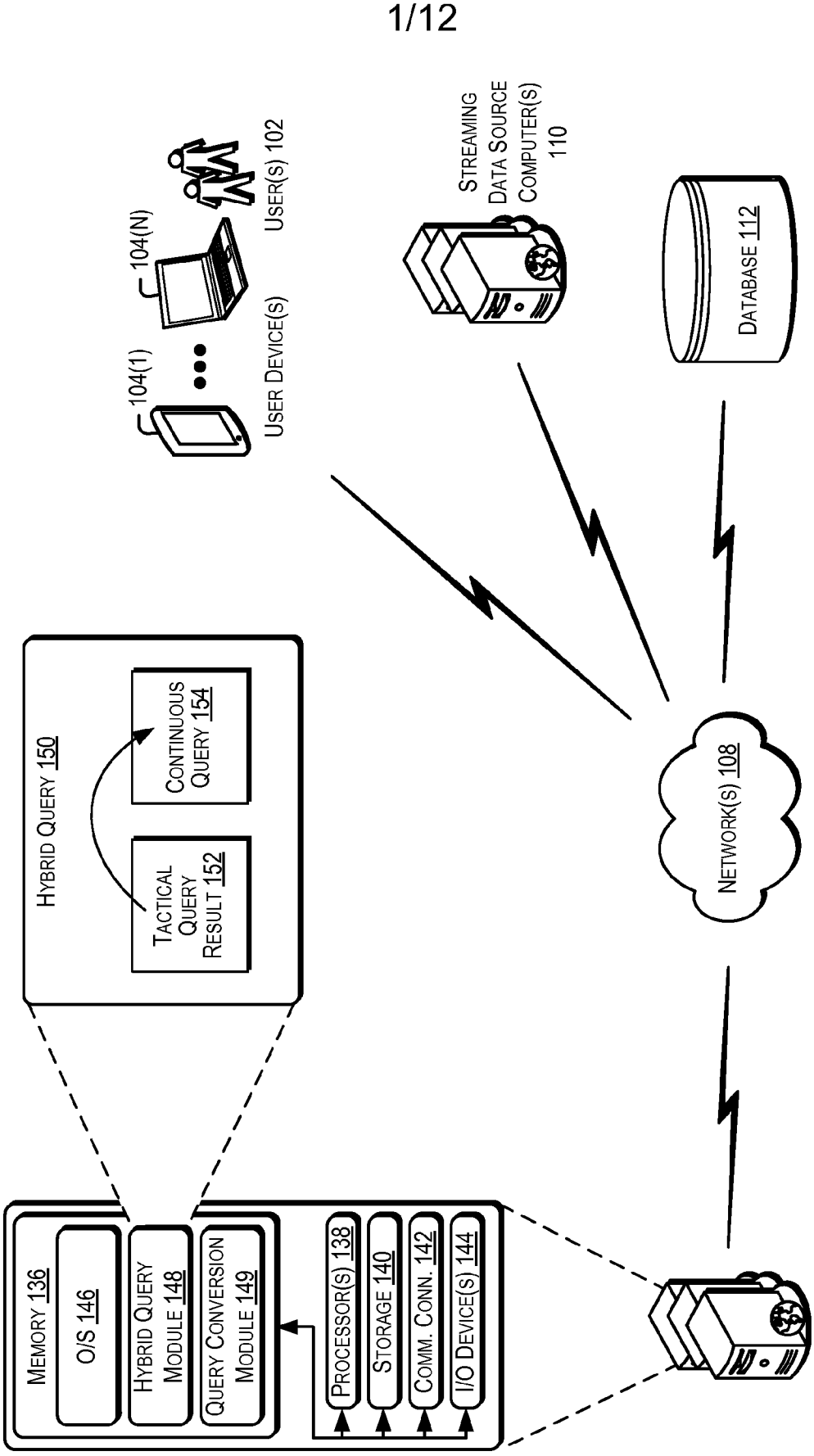
100
USER(S) 102
104(N)
104(1)
USER DEVICE(S)
STREAMING DATA SOURCE COMPUTER(S) 110
DATABASE 112
HYBRID QUERY 150
CONTINUOUS QUERY 154
TACTICAL QUERY RESULT 152
NETWORK(S) 108
MEMORY 136
O/S 146
HYBRID QUERY MODULE 148
QUERY CONVERSION MODULE 149
PROCESSOR(S) 138
STORAGE 140
COMM. CONN. 142
I/O DEVICE(S) 144
SERVICE PROVIDER COMPUTER(S) 106

FIG. 1

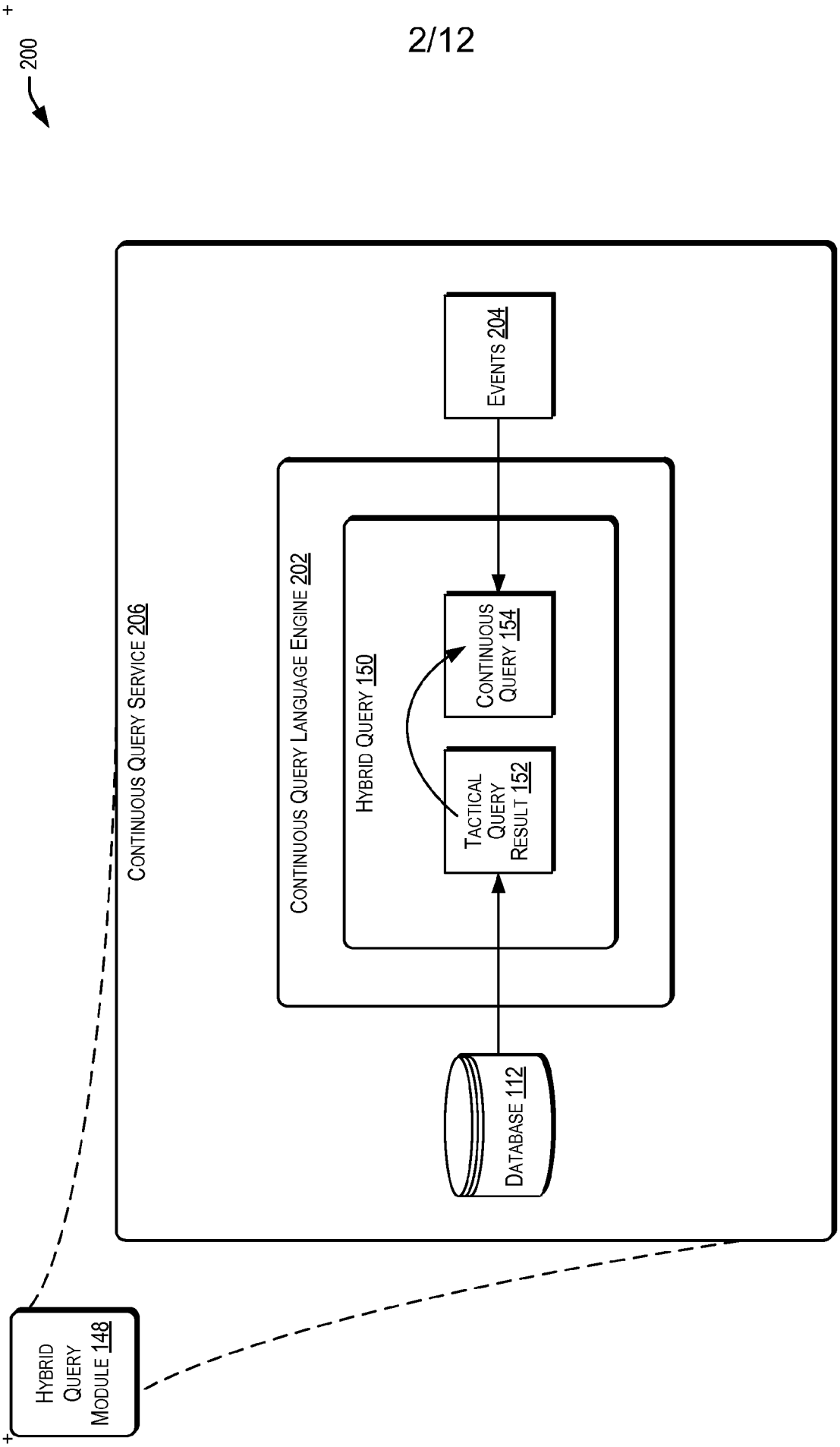
200
200
CONTINUOUS QUERY SERVICE 206
CONTINUOUS QUERY LANGUAGE ENGINE 202
HYBRID QUERY 150
EVENTS 204
CONTINUOUS QUERY 154
TACTICAL QUERY RESULT 152
DATABASE 112
HYBRID QUERY MODULE 148

FIG. 2

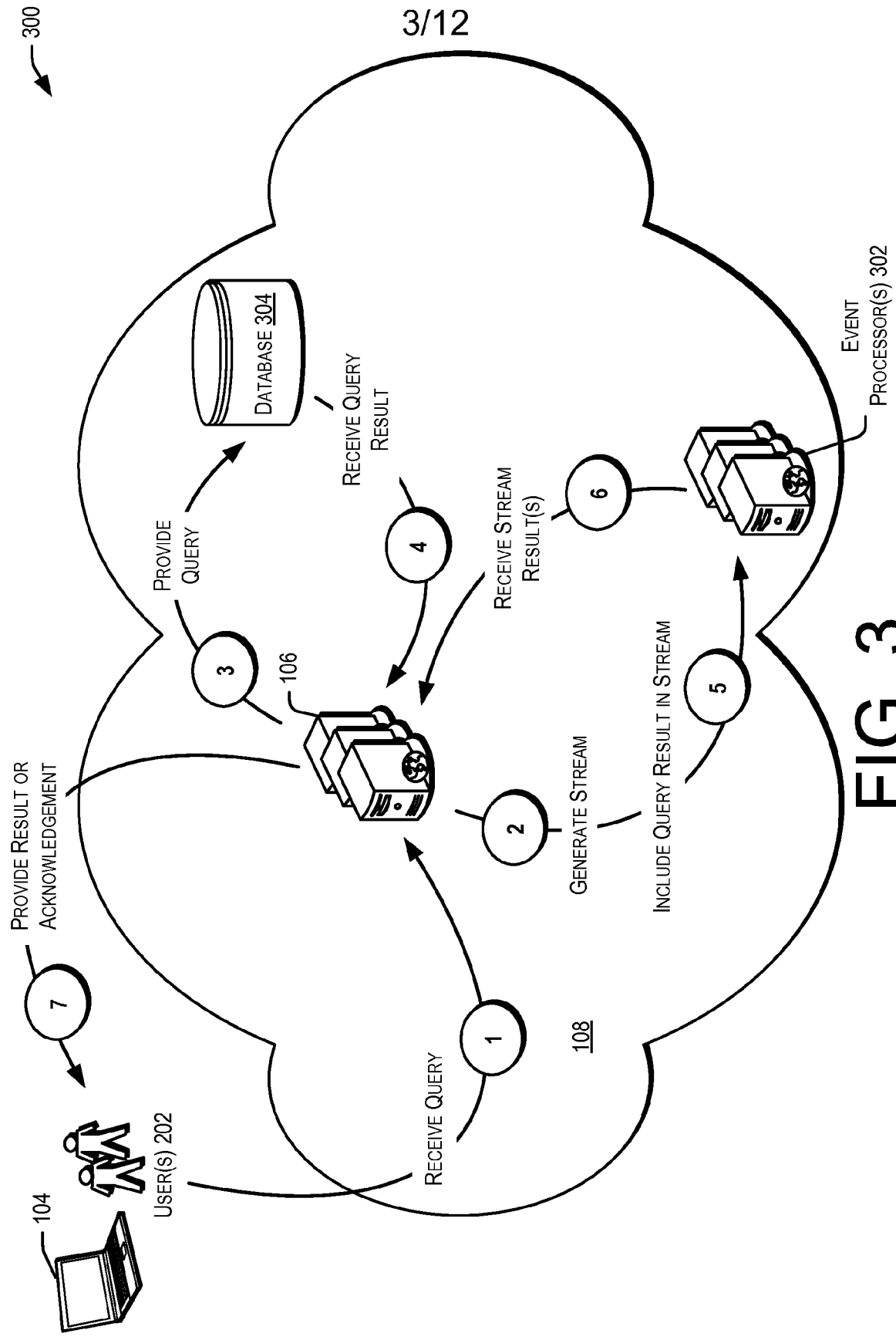
300
DATABASE 304
RECEIVE QUERY RESULT
PROVIDE QUERY
RECEIVE STREAM RESULT(S)
EVENT PROCESSOR(S) 302
106
PROVIDE RESULT OR ACKNOWLEDGEMENT
GENERATE STREAM
INCLUDE QUERY RESULT IN STREAM
RECEIVE QUERY
108
USER(S) 202
104
1
2
3
4
5
6
7

FIG. 3

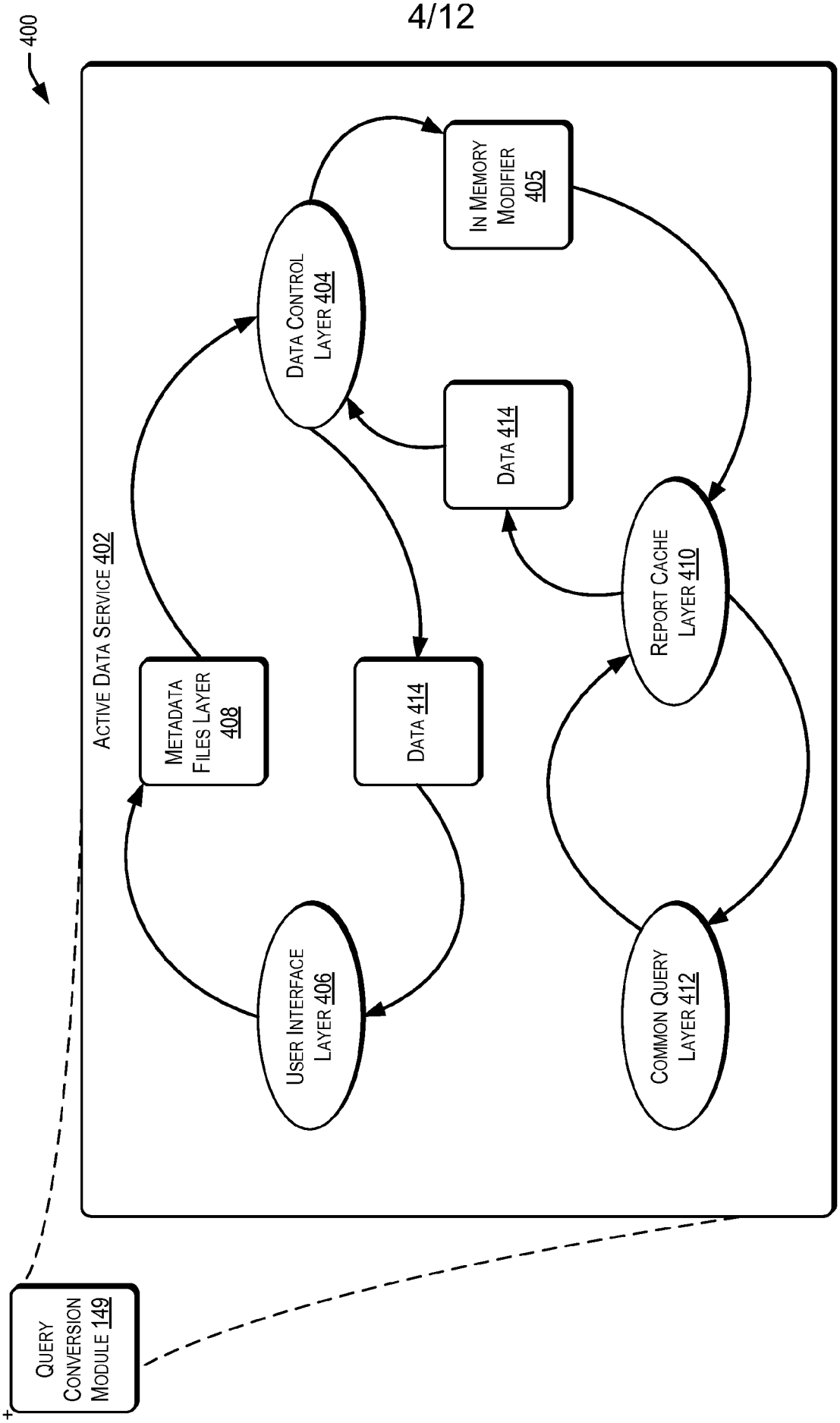
400
ACTIVE DATA SERVICE 402
METADATA FILES LAYER 408
DATA CONTROL LAYER 404
IN MEMORY MODIFIER 405
DATA 414
DATA 414
REPORT CACHE LAYER 410
USER INTERFACE LAYER 406
COMMON QUERY LAYER 412
QUERY CONVERSION MODULE 149

FIG. 4

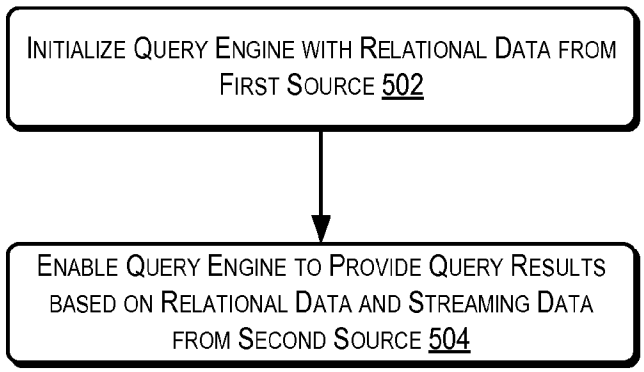
500
INITIALIZE QUERY ENGINE WITH RELATIONAL DATA FROM FIRST SOURCE 502
ENABLE QUERY ENGINE TO PROVIDE QUERY RESULTS BASED ON RELATIONAL DATA AND STREAMING DATA FROM SECOND SOURCE 504

FIG. 5

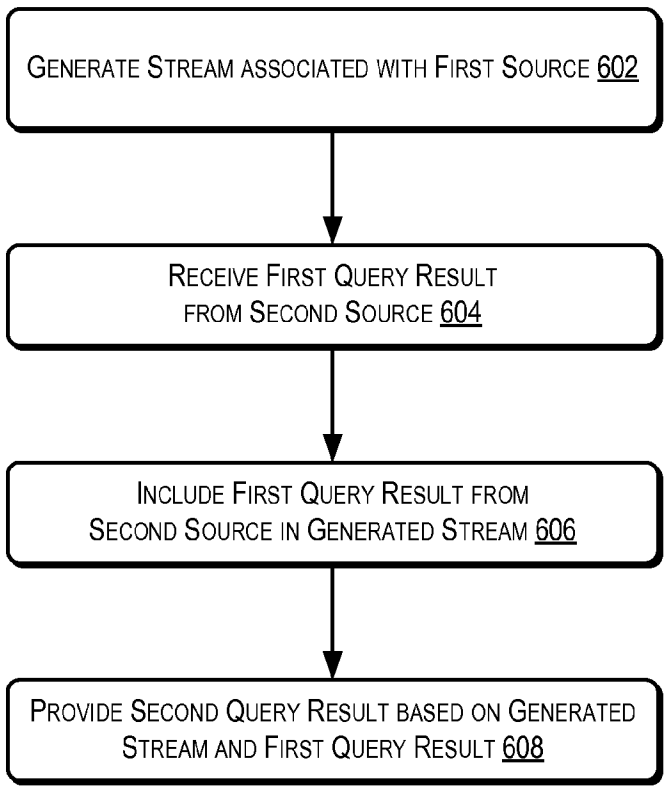
600
GENERATE STREAM ASSOCIATED WITH FIRST SOURCE 602
RECEIVE FIRST QUERY RESULT FROM SECOND SOURCE 604
INCLUDE FIRST QUERY RESULT FROM SECOND SOURCE IN GENERATED STREAM 606
PROVIDE SECOND QUERY RESULT BASED ON GENERATED STREAM AND FIRST QUERY RESULT 608

FIG. 6

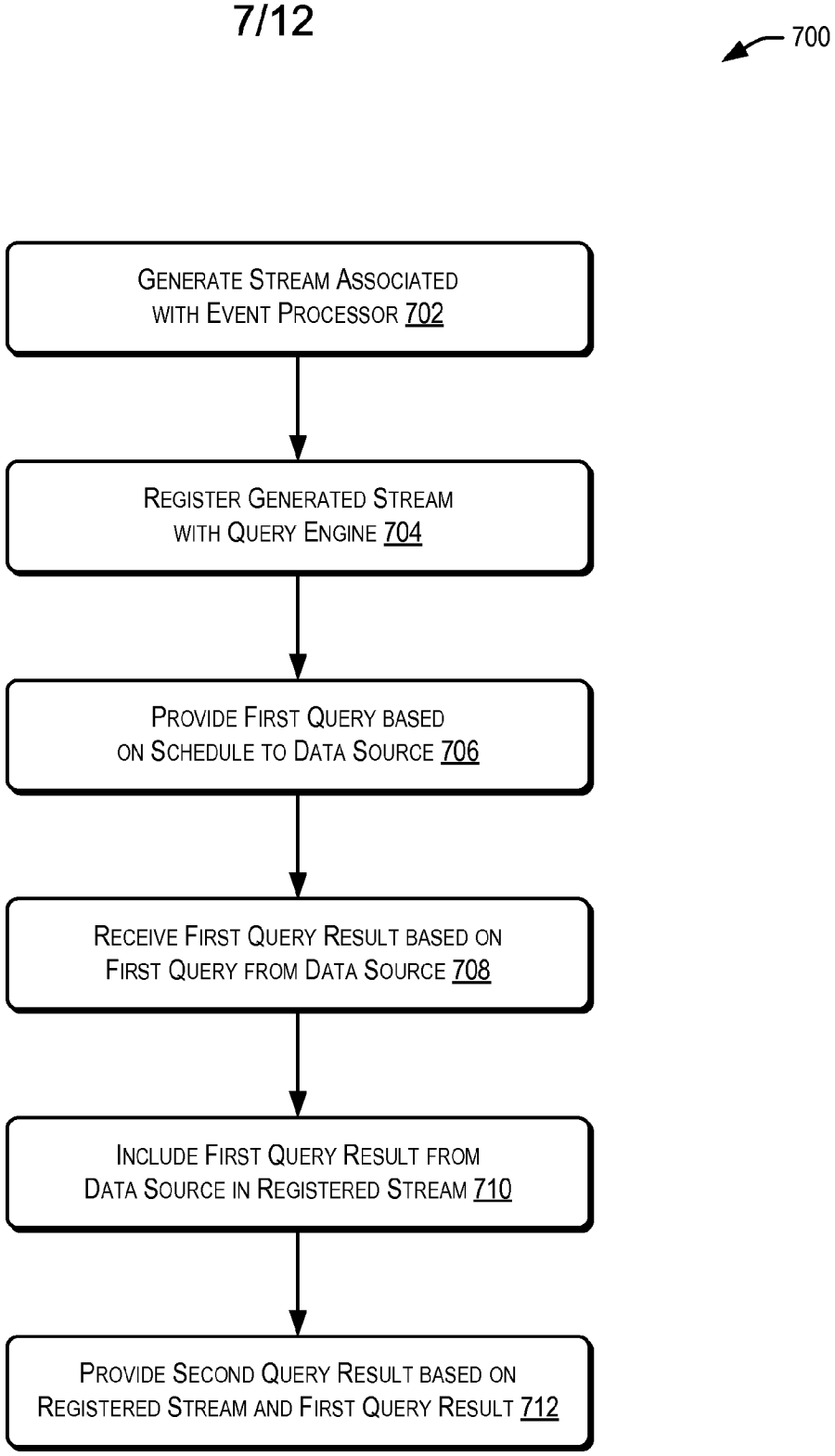
700
GENERATE STREAM ASSOCIATED WITH EVENT PROCESSOR 702
REGISTER GENERATED STREAM WITH QUERY ENGINE 704
PROVIDE FIRST QUERY BASED ON SCHEDULE TO DATA SOURCE 706
RECEIVE FIRST QUERY RESULT BASED ON FIRST QUERY FROM DATA SOURCE 708
INCLUDE FIRST QUERY RESULT FROM DATA SOURCE IN REGISTERED STREAM 710
PROVIDE SECOND QUERY RESULT BASED ON REGISTERED STREAM AND FIRST QUERY RESULT 712

FIG. 7

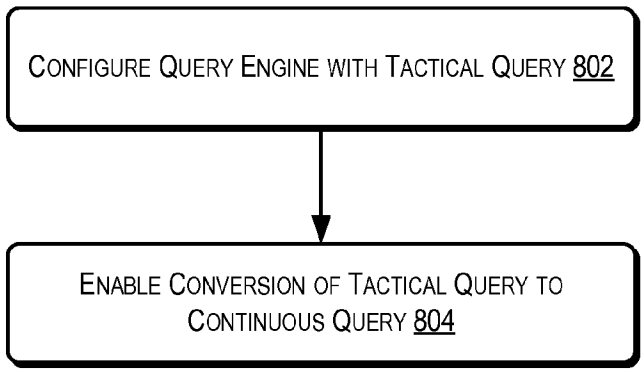
800
CONFIGURE QUERY ENGINE WITH TACTICAL QUERY 802
ENABLE CONVERSION OF TACTICAL QUERY TO CONTINUOUS QUERY 804

FIG. 8

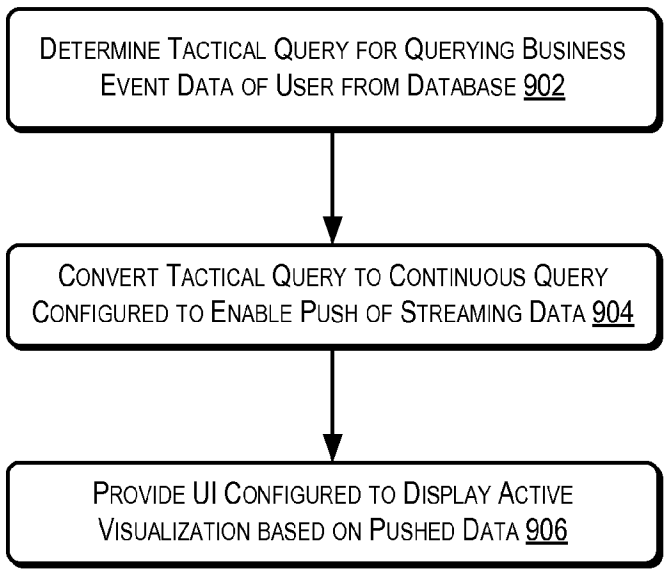
900
DETERMINE TACTICAL QUERY FOR QUERYING BUSINESS EVENT DATA OF USER FROM DATABASE 902
CONVERT TACTICAL QUERY TO CONTINUOUS QUERY CONFIGURED TO ENABLE PUSH OF STREAMING DATA 904
PROVIDE UI CONFIGURED TO DISPLAY ACTIVE VISUALIZATION BASED ON PUSHED DATA 906

FIG. 9

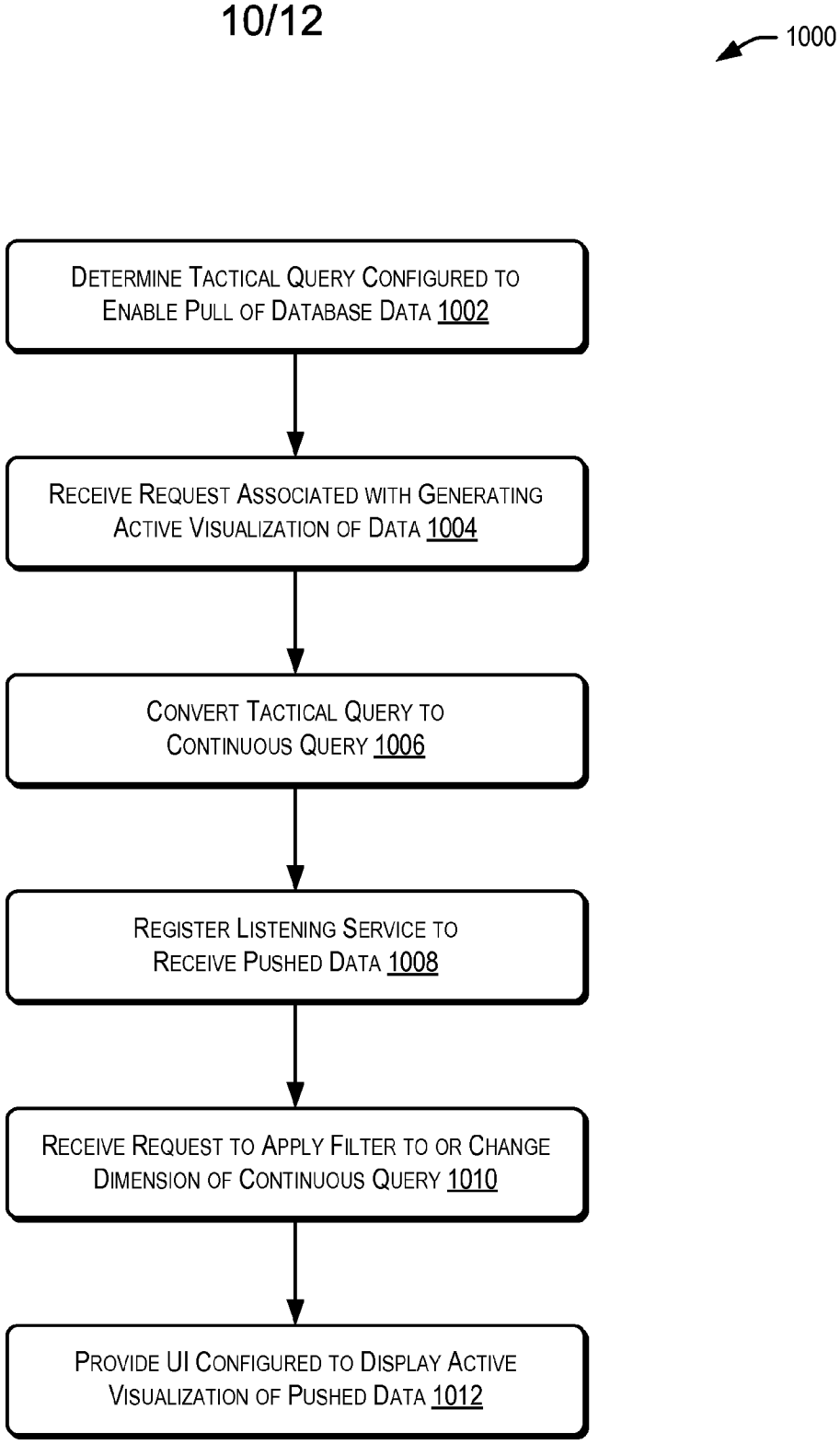
1000
DETERMINE TACTICAL QUERY CONFIGURED TO ENABLE PULL OF DATABASE DATA 1002
RECEIVE REQUEST ASSOCIATED WITH GENERATING ACTIVE VISUALIZATION OF DATA 1004
CONVERT TACTICAL QUERY TO CONTINUOUS QUERY 1006
REGISTER LISTENING SERVICE TO RECEIVE PUSHED DATA 1008
RECEIVE REQUEST TO APPLY FILTER TO OR CHANGE DIMENSION OF CONTINUOUS QUERY 1010
PROVIDE UI CONFIGURED TO DISPLAY ACTIVE VISUALIZATION OF PUSHED DATA 1012

FIG. 10

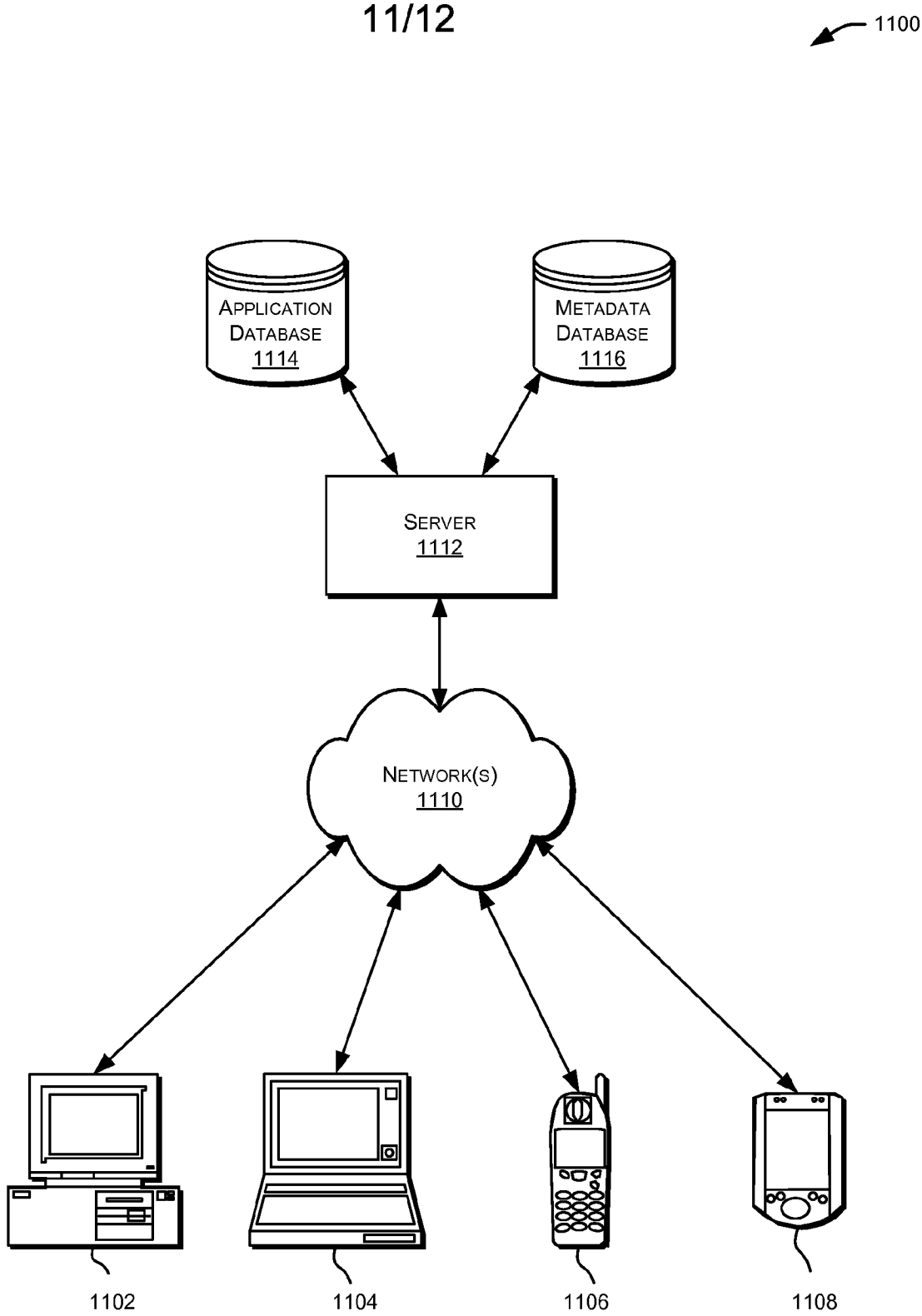
1100
APPLICATION DATABASE 1114
METADATA DATABASE 1116
SERVER 1112
NETWORK(S) 1110
1102
1104
1106
1108

FIG. 11

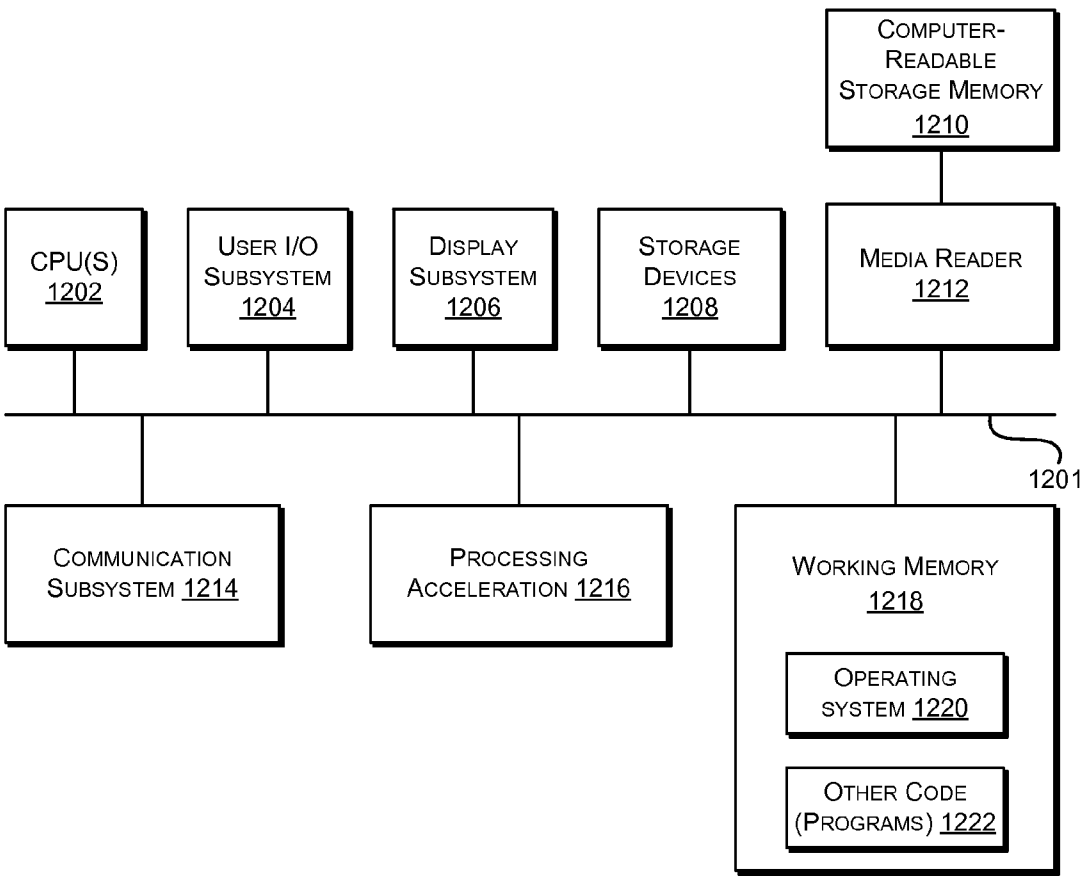
1200
COMPUTER-READABLE STORAGE MEMORY 1210
CPU(S) 1202
USER I/O SUBSYSTEM 1204
DISPLAY SUBSYSTEM 1206
STORAGE DEVICES 1208
MEDIA READER 1212
1201
COMMUNICATION SUBSYSTEM 1214
PROCESSING ACCELERATION 1216
WORKING MEMORY 1218
OPERATING SYSTEM 1220
OTHER CODE (PROGRAMS) 1222

FIG. 12