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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
4 December 2014 (04.12.2014)

WIPO | PCT

(10) International Publication Number
WO 2014/193943 A1

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

(21) International Application Number:
PCT/US2014/039771

(22) International Filing Date:
28 May 2014 (28.05.2014)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
13/906,162 30 May 2013 (30.05.2013) US

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

(72) Inventors: **BISHNOI, Sandeep**; 29-MITC Colony, Barnala Road, Sirsa, Haryana 125055 (IN). **SRINIVASAN, Anand**; E 303 Mantri Elegance, N.S. Palya, Bannerghatta Road, Bangalore, Karnataka 560076 (IN). **DESHMUKH, Unmesh Anil**; Plot No. 222-A H.B. Estate Near Shiv Mandir, Sonegaon, Nagpur, Maharashtra 440025 (IN).

(74) Agents: **SRIPATHY, Kanchan** 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, IR, 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,

[Continued on next page]

(54) Title: VALUE BASED WINDOWS ON RELATIONS IN CONTINUOUS DATA STREAMS

(57) Abstract: Techniques for managing value-based windows on relations are provided. In some examples, an input relation is generated. The input relation is a bounded set of data records related to an application. A continuous query that identifies the input relation may be received. Additionally, a configurable window operator associated with processing the input relation may be identified. Then, the continuous query may be executed based at least in part on the configurable window operator to generate an output relation. Further, in some instances, the data records of the output relation may be provided based at least in part on execution of the continuous query.

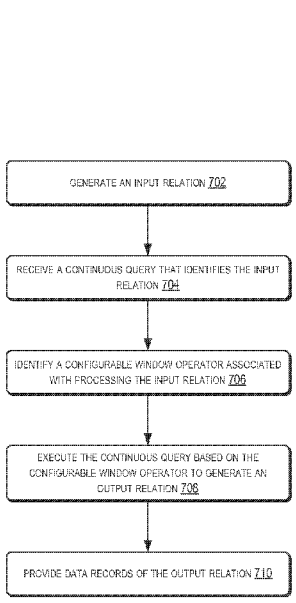


FIG. 7

WO 2014/193943 A1

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:

— *with international search report (Art. 21(3))*

— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments (Rule 48.2(h))*

VALUE BASED WINDOWS ON RELATIONS IN CONTINUOUS DATA STREAMS

BACKGROUND

[0001] In traditional database systems, data is stored in one or more databases usually in the form of tables. The stored data is then queried and manipulated using a data management language such as a structured query language (SQL). For example, a SQL query may be defined and executed to identify relevant data from the data stored in the database. A SQL query is thus executed on a finite set of data stored in the database. Further, when a SQL query is executed, it is executed once on the finite data set and produces a finite static result. Databases are thus best equipped to run queries over finite stored data sets.

[0002] A number of modern applications and systems however generate data in the form of continuous data or event streams instead of a finite data set. Examples of such applications include but are not limited to sensor data applications, financial tickers, network performance measuring tools (e.g. network monitoring and traffic management applications), clickstream analysis tools, automobile traffic monitoring, and the like. Such applications have given rise to a need for a new breed of applications that can process the data streams. For example, a temperature sensor may be configured to send out temperature readings.

[0003] Managing and processing data for these types of event stream-based applications involves building data management and querying capabilities with a strong temporal focus. A different kind of querying mechanism is needed that comprises long-running queries over continuous unbounded sets of data. While some vendors now offer product suites geared towards event streams processing, these product offerings still lack the processing flexibility required for handling today's events processing needs.

Brief Summary

[0004] In some examples, a method is provided for managing value-based windows on relations. The method may include receiving a continuous query that identifies an input relation. In one example, the input relation is a bounded set of data records related to an application. The method may then include identifying a configurable window operator associated with processing the input relation. Additionally, the method may include executing the continuous query based at least in part on the configurable window operator to generate an output relation. In some aspects, the method may also include providing data records of the output relation based at least in part on execution of the continuous query.

[0005] In some examples, the input relation may be an external data source generated based at least in part on an incoming continuous input data stream related to the application. Additionally, in some examples, the input relation may be an external data source generated based at least in part on information related to the application stored in a database of historical data. The input relation may also be an external data source generated based at least in part on one or more archived relations related to the application.

[0006] In some examples, the configurable window operator may be a generic value window operator defined over the input relation and the method may include applying the generic value window operator on an attribute in the input relation to generate the output relation. In other examples, the configurable window operator may be a current hour value window operator defined over the input relation and the method may include applying the current hour value window operator on an attribute in the input relation to generate the output relation. Additionally, in some examples, the configurable window operator may be a current period value window operator defined over the input relation and the method may include applying the current period value window operator on an attribute in the input relation to generate the output relation.

[0007] In some aspects, the method may also include displaying the output relation. In some examples, the output relation may include a subset of the data records from the input

relation whose attribute values lie within a specified range defined by the configurable window operator.

[0008] In some examples, a non-transitory computer-readable medium may be provided. The medium may store a plurality of instructions executable by one or more processors. The instructions, in some examples, may include to generating an input relation and identifying a query configured to process the input relation. Additionally, the instructions may include identifying a configurable window operator associated with processing the input relation and executing the query based at least in part on the configurable window operator to generate an output relation. In some aspects, the instructions may also include providing data records of the output relation based at least in part on execution of the continuous query by displaying the output relation. In some examples, the output relation may include a subset of the data records from the input relation whose attribute values lie within a specified range defined by the configurable window operator.

[0009] In some examples, a system may be provided. The system may include a memory and one or more processors configured to access the memory and execute instructions to generate an input relation. The instructions may also be executed to identify a query configured to process the input relation and identify a configurable window operator associated with processing the input relation. Additionally, the instructions may be executed to execute the query based at least in part on the configurable window operator to generate an output relation.

[0010] In some examples, a computer-implemented apparatus may be provided, which comprises: means for generating an input relation, the input relation being a bounded set of data records related to an application; means for receiving a continuous query that identifies the input relation; means for identifying a configurable window operator associated with processing the input relation; means for executing the continuous query based at least in part on the configurable window operator to generate an output relation; and means for providing data records of the output relation based at least in part on execution of the continuous query.

[0011] In some examples, the input relation may be an external data source generated based at least in part on an incoming continuous input data stream related to the application.

[0012] In some examples, the input relation may be an external data source generated based at least in part on information related to the application stored in a database of historical data.

[0013] In some examples, the input relation may be an external data source generated based at least in part on one or more archived relations related to the application.

[0014] In some examples, the configurable window operator may be a generic value window operator defined over the input relation, and means for executing the continuous query comprises means for applying the generic value window operator on an attribute in the input relation to generate the output relation.

[0015] In some examples, the configurable window operator may be a current hour value window operator defined over the input relation, and means for executing the continuous query comprises means for applying the current hour value window operator on an attribute in the input relation to generate the output relation.

[0016] In some examples, the configurable window operator may be a current period value window operator defined over the input relation, and means for executing the continuous query comprises means for applying the current period value window operator on an attribute in the input relation to generate the output relation.

[0017] In some examples, means for providing the data records comprises means for displaying the output relation, wherein the output relation comprises a subset of the data records from the input relation whose attribute values lie within a specified range defined by the configurable window operator.

[0018] In some examples, a service provider device (1101) may be provided, which comprises: an input relation unit (1102), configured to receive a continuous query that identifies an input relation, wherein the input relation being a bounded set of data records related to an application; a configurable window operator unit (1103) , configured to identify a configurable window operator associated with processing the input relation; an output relation unit (1105), configured to execute the continuous query based at least in part on the configurable window

operator to generate an output relation; and a providing unit (1107), configured to provide data records of the output relation based at least in part on execution of the continuous query.

[0019] In some examples, the input relation may be an external data source generated based at least in part on an incoming continuous input data stream related to an application.

[0020] In some examples, the input relation may be an external data source generated based at least in part on information related to an application stored in a database of historical data.

[0021] In some examples, the input relation may be an external data source generated based at least in part on one or more archived relations related to an application.

[0022] In some examples, the configurable window operator may be a generic value window operator defined over the input relation, and wherein the output relation unit (1105) may be further configured to apply the generic value window operator on an attribute in the input relation to generate the output relation.

[0023] In some examples, the configurable window operator may be a current hour value window operator defined over the input relation, and the output relation unit (1105) may be further configured to apply the current hour value window operator on an attribute in the input relation to generate the output relation.

[0024] In some examples, the configurable window operator may be a current period value window operator defined over the input relation, and the output relation unit (1105) may be further configured to apply the current period value window operator on an attribute in the input relation to generate the output relation.

[0025] In some examples, the providing unit (1107) may be further configured to display the output relation, wherein the output relation comprises a subset of the data records from the input relation whose attribute values lie within a specified range defined by the configurable window operator.

[0026] In some examples, a computer-implemented apparatus may be provided, which comprises: means for generating an input relation; means for identifying a query to process the input relation; means for identifying a configurable window operator associated with processing

the input relation; and means for executing the query based at least in part on the configurable window operator to generate an output relation.

[0027] In some examples, the input relation may be an external data source generated based at least in part on an incoming continuous input data stream related to the application.

[0028] In some examples, the input relation may be an external data source generated based at least in part on information related to the application stored in a database of historical data.

[0029] In some examples, the input relation may be an external data source generated based at least in part on one or more archived relations related to the application.

[0030] In some examples, the configurable window operator may be a generic value window operator defined over the input relation, and means for executing the continuous query comprises means for applying the generic value window operator on an attribute in the input relation to generate the output relation.

[0031] In some examples, the apparatus may further comprise means for providing data records of the output relation based at least in part on execution of the continuous query.

[0032] In some examples, the apparatus may further comprise means for displaying the output relation, the output relation comprises a subset of the data records from the input relation whose attribute values lie within a specified range defined by the configurable window operator.

[0033] In some examples, a service provider device (1201) may be provided, which comprises: an input relation unit (1202), configured to receive and identify a query configured to process an input relation; a configurable window operator unit (1203) configured to identify a configurable window operator associated with processing the input relation; and an output relation unit (1205) configured to execute the query based at least in part on the configurable window operator to generate an output relation.

[0034] In some examples, the input relation may be an external data source generated based at least in part on an incoming continuous input data stream related to the application.

[0035] In some examples, the input relation may be an external data source generated based at least in part on information related to the application stored in a database of historical data.

[0036] In some examples, the input relation may be an external data source generated based at least in part on one or more archived relations related to the application.

[0037] In some examples, the configurable window operator may be a generic value window operator defined over the input relation, and the output relation unit (1205) may be further configured to apply the generic value window operator on an attribute in the input relation to generate the output relation.

[0038] In some examples, the service provider device (1201) may further comprise a providing unit (1207) configured provide data records of the output relation based at least in part on execution of the continuous query.

[0039] In one embodiment, the providing unit (1207) may be further configured to display the output relation, wherein the output relation comprises a subset of the data records from the input relation whose attribute values lie within a specified range defined by the configurable window operator.

[0040] 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

[0041] 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.

[0042] FIG. 1 depicts a simplified example system or architecture in which techniques for managing value-based windows on relations may be implemented.

[0043] FIG. 2 illustrates a simplified block diagram with which features for the management of value-based windows on relations may be described.

[0044] FIG. 3 illustrates exemplary information stored in an input relation and the generation of an output relation when the input relation is processed using a configurable window operator, in accordance with one embodiment of the present disclosure.

[0045] FIG. 4 illustrates exemplary information stored in an input relation and the generation of an output relation when the input relation is processed using a configurable window operator, in accordance with another embodiment of the present disclosure.

[0046] FIG. 5 illustrates exemplary information stored in an input relation and the generation of an output relation when the input relation is processed using a configurable window operator, in accordance with another embodiment of the present disclosure.

[0047] FIG. 6 is a simplified process flow illustrating at least some features of the management of value-based windows on relations described herein, according to at least one example.

[0048] FIG. 7 is another simplified process flow illustrating at least some features of the management of value-based windows on relations described herein, according to at least one example.

[0049] FIG. 8 is a simplified block diagram illustrating components of a system environment that may be used in accordance with an embodiment of the management of value-based windows on relations described herein, according to at least one example.

[0050] FIG. 9 is a simplified block diagram illustrating a computer system that may be used in accordance with embodiments of the management of value-based windows on relations described herein described herein, according to at least one example.

[0051] FIG. 10 illustrates an example flow diagram showing a process for implementing the management of value-based windows on relations described herein, in accordance with another embodiment of the present invention.

[0052] FIG. 11 is a simplified block diagram of a service provider device that may be used in accordance with certain embodiments of the present invention.

[0053] FIG. 12 is a simplified block diagram of a service provider device that may be used in accordance with certain embodiments of the present invention.

DETAILED DESCRIPTION

[0054] 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.

[0055] In some applications, data may take the form of continuous, unbounded data streams, rather than finite stored data sets. Examples of such data streams may include stock ticks in financial applications, performance measurements in network monitoring and traffic management, log records or click-streams in web tracking and personalization, data feeds from sensor applications, network packets and messages in firewall-based security, call detail records in telecommunications, and the like. Due to their continuous nature, these data streams may typically be queried using continuous queries rather than traditional one-time SQL queries.

[0056] In general, 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.

[0057] 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.

[0058] 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.

[0059] 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.

[0060] 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.

[0061] 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.

[0062] As noted, a stream may be the principle source of data that CQL queries may act on. A stream S may be a bag (also referred to as a "multi-set") of elements (s, T), where "s" is in the schema of S and "T" is in the time domain. Additionally, stream elements may be tuple-timestamp pairs, which can be represented as a sequence of timestamped tuple insertions. In other words, a stream may be a sequence of timestamped tuples. In some cases, there may be more than one tuple with the same timestamp. And, the tuples of an input stream may be

requested to arrive at the system in order of increasing timestamps. Alternatively, a relation (also referred to as a "time varying relation," and not to be confused with "relational data," which may include data from a relational database) may be a mapping from the time domain to an unbounded bag of tuples of the schema R. In some examples, a relation may be an unordered, time-varying bag of tuples (i.e., an instantaneous relation). In some cases, at each instance of time, a relation may be a bounded set. It can also be represented as a sequence of timestamped tuples that may include insertions, deletes, and/or updates to capture the changing state of the relation. Similar to streams, a relation may have a fixed schema to which each tuple of the relation may conform. Further, as used herein, a continuous query may generally be capable of processing data of (i.e., queried against) a stream and/or a relation. Additionally, the relation may reference data of the stream.

[0063] 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.

[0064] 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.

[0065] 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.

[0066] 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.

[0067] 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 real-time event processing 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.

[0068] 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.

[0069] 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.

[0070] In some examples, the above concepts may be utilized to leverage the rich real-time and continuous event processing capabilities associated with complex event processing. Several features may be supported such as, but not limited to, archived relations. As such, in order to leverage such features (e.g., rich, real-time and continuous event processing), the system may be configured to transparently deal with startup state and runtime state of relational data. In other words, the system may be configured to manage a query that is non-empty at the instant of its creation (i.e., an archived relation).

[0071] In some examples, an archived relation may be utilized. As such, when a CQL engine sees a query that indicates that it is based on an archived relation; that archived relation may also indicate that there are certain entities it can call to query for historical context, for example. In some examples, a data definition language (DDL) may indicate annotations about the archived relation such as, but not limited to, how do to the query, what are the important columns in the table, and/or where to send the rest of the data. In some examples, once the query is constructed in the CQL engine (e.g., as a graph), the system may analyze the query graph. Additionally, in some aspects, there are certain operators that are stateful, like "distinct," "group aggr," "pattern," and/or "group by." However, stateless operators may just take input and send it to the parent, for example, down-stream operators. So, one approach is to store this entire table here. However, utilizing archived relations, the system may analyze the query graph and decide which of the lowest stateful operator that it can use to query the archive. In some examples, the system (or one

or more computer-implemented methods) may retrieve the state at the lowest stateful operator reached while traversing the graph. For example, the query graph may be analyzed in a topological order from the source. Based at least in part on this first stateful operator, the CQL engine may then determine the optimal amount of data to be fetched in order to initialize the state of the operators for a query defined over an archived relation.

[0072] In at least one non-limiting example, source operators like relation and/or source may come first in the topological traversal with query output and/or root coming last. For example, if the CQL query looks like: select sum(c1) from R1 where c2>c25, the plan for this query may look like: RelationSource → SELECT → GroupAggr. Thus, following the topological order, and since RelationSource and SELECT are both stateless, the lowest stateful operator may be GroupAggr. In this way, the stateful operators of a query (GroupAggr in this example) may enable the query engine to populate the query engine with historical data from a data store prior to receiving streaming data. This may be enabled based at least in part on the fact that the query is analyzing an archived relation and the archived relation has been indicated as such.

[0073] In some examples, a window size for a given archive relation may be specified by a user. A window, in some aspects, in relation to an archived relation, may include a node in a query graph that analyzes or otherwise evaluates incoming active data. In other words, the window may define the amount of active data that may be analyzed and/or processed by the query engine and/or the amount of historical data that will be included in the archived relation.

[0074] At a high level, once a window is applied on a Stream it becomes a Relation and then regular relational logic may be applied, as with relational databases. As tuples arrive and leave the window, the Relation under consideration changes with queries compiled against it emitting results at the same time. CQL may support RANGE (up to nanoseconds granularity), ROWS, PARTITION BY and extensible windows. These windows are examples of stream-to-relation operators. On the other hand, ISTREAM (i.e., insert stream), DSTREAM (i.e., delete stream) and RSTREAM (i.e., relation stream) are relation-to-stream operators. In some examples, a user, developer, and/or manager may set the window size (e.g., via a UI) provided by the query engine or one or more computing systems operating or hosting the query engine. In some examples, a window on a stream may be a time-based range window. For example, a configurable value

window on an archived relation may be specified using window size and the attribute on which the window is calculated. When there is a configurable value window specified on top of archived relation, a snapshot query may be computed and the snapshot tuples which are within window limits may be output. Additionally, after state initialization, the value window may be applied on incoming active data. In some examples, only the incoming active data will be inserted into window whose window attribute's value is differing from current event time for less than the window size.

[0075] As discussed above, in some examples, the processing of continuous data streams using continuous queries may involve applying a window on the stream, specifying that a fixed number of the most recent stream tuples are considered instead of the entire stream history, or specifying that only stream tuples that have arrived within a specified time are considered, instead of the entire stream history. As further discussed above, in some examples, continuous query processing may involve the use of various operators that process incoming continuous streams to convert it to a relation or operators that convert an input relation to output stream data.

[0076] In certain situations, however, a user may set a query that may require the processing of incoming real-time data related to an application with data related to the application that is stored in a relation (e.g., an external data source). If the size of the relation is very large, the amount of data that is needed to be brought into memory to query this data may also be very large. Since the CQL engine may typically be an in-memory real-time event processing engine configured to query or otherwise process incoming real-time information, it may not be scalable to join such a high rate input stream with a big sized relation in limited memory.

[0077] Accordingly, in some embodiments, a window operator may be defined for processing a relation in a continuous query (e.g., a CQL query). In one example, the window operator may be user-configurable. The window operator may enable processing a subset of data records in the relation over a specified range of time. Accordingly, a subset of the data records of the relation may be queried in limited memory rather than processing the entire set of data records of the relation in memory.

[0078] In certain embodiments, an input relation may be generated. In one example, the

input relation may be an external data source that includes a bounded set of data records related to an application. In one example, the input relation may be generated from a continuous input data stream. Alternatively, the input relation may be generated based on a database of historical data related to the application. Additionally, the input relation may initially be generated from one or more archived relations and the remainder of the input relation may be generated based on the incoming streaming data.

[0079] In accordance with one embodiment of the present disclosure, a continuous query (e.g., a CQL query) that identifies the input relation may be received. A window operator defined over the input relation may then be identified. The query may be executed based at least in part on the window operator to generate an output relation. In one embodiment, the output relation may include a subset of data records from the input relation whose attribute values may lie within a specified range defined by the window operator in the input relation.

[0080] 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.

[0081] FIG. 1 depicts a simplified example system or architecture 100 in which techniques for managing value-based windows on relations 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 (e.g., queries may be run against either or both of 110, 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.

[0082] 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.).

[0083] 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.

[0084] 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 management of CQL relations, generation of input relations, configurable window operators associated with the input relations, and the generation of output relations, 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.

[0085] 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.

[0086] 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.

[0087] 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.

[0088] The service provider computers 106 may also contain communications connection(s) 142 that allow the service provider computers 106 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.

[0089] 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 an input relation module 148, a configurable window operator module 150 and an output relation module 152. 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.

[0090] In some examples, the input relation module 148 may be configured to, receive, identify, generate, or otherwise provide one or more input relations. In one embodiment, the input relation 154 is an external data source that includes a bounded set of data records related to an application (e.g., a relation or an archived relation).

[0091] In one example, the input relation 154 may be an external data source that may be generated based on an incoming continuous input data stream that includes a stream of data or events related to the application (e.g., from the data source computers 110), wherein the input relation may include a bounded set of data records that comprise one or more of the event stream entries s1, s2, ..., sN in the continuous data stream. In another example, the input relation 154 may be an external data source that may be generated based on information related to the application stored in a database of historical data (for e.g., the databases 112), wherein the input relation may include a bounded set of data records that include or more event stream entries

(e.g., s1 and/or s2, more entries, or less) that are pre-loaded from Persistence, or a database of historical data.

[0092] In other examples, the input relation 154 may be an external data source that may be generated based on one or more archived relations that may include reference to one or more event stream entries s1, s2, …, sN related to the application. In one example, the input relation 154 may be initially generated from one or more archived relations and the remainder of the input relation may be generated based on the incoming streaming data.

[0093] In one example, the data records in the input relation 154 may include a sequence of time stamped tuples or data records that capture the changing state of the input relation 154. In one embodiment, each data record in the input relation 154 may include an event stream entry that may be represented by the following schema 'S': (<time_stamp>, <attribute(s)>), where <attributes> represents the data portion of the schema and can comprise one or more attributes.

[0094] The configurable window operator module 150 may be configured to define one or more configurable window operators 156 for processing the input relation in a continuous query. There are various ways in which a configurable window operator may be defined for an input relation. In one embodiment, the configurable window operator may be user-configurable. In certain embodiments, and as will be discussed in detail below, the configurable window operator may produce an output relation which includes a subset of data records from the input relation whose attribute values lie in a range specified by the configurable window operator with respect to the timestamp t of a data record in the input relation.

[0095] In one embodiment, a first type of configurable window operator may be defined as a "generic value" window operator over the input relation. In one example, the "generic value" window operator may be expressed as a sub-clause on the input relation 154 as follows:

[0096] RANGE range_val ON attr.

[0097] In certain embodiments, the "generic value" window operator may produce an output relation which includes a subset of data records from the input relation whose attribute "attr" values lie in the specified range, "range_val" with respect to the timestamp t of a corresponding data record in the input relation. An example operation of the manner in which the "generic

value" window operator may be defined over an input relation in a CQL query is discussed in detail below.

[0098] In another embodiment, a second type of configurable window operator may be defined as a "current hour value" window operator on the input relation. In one example, the "current hour value" window operator may be expressed as a sub-clause on an input relation as follows:

[0099] CurrentHour on attr

[00100] In one embodiment, the "current hour value" window operator may produce an output relation which includes a subset of data records from the input relation whose attribute "attr" values lies in the current hour, "CurrentHour" with respect to the timestamp t of a corresponding data record in the input relation. An example operation of the manner in which the "current hour value" window operator may be defined over an input relation in a CQL query is discussed in detail below.

[00101] In another embodiment, a third type of configurable window operator may be defined as a "current period value" window operator on the input relation. In one example, the "current period value" window operator may be expressed as a sub-clause on an input relation as follows:

[00102] CurrentPeriod("t1","t2") on attr

[00103] In one embodiment, the "current period value" window operator may produce an output relation which includes a subset of data records from the input relation whose attribute "attr" values lies in the current time period, "CurrentPeriod" with respect to the timestamp t of a corresponding data record in the input relation. An example operation of the manner in which the "CurrentPeriod" window operator may be defined over an input relation in a CQL query is discussed in detail below.

[00104] Although the above discussion relates to defining three types of configurable window operators over input relations in a CQL query, it is to be appreciated that additional types of window operators may be defined by the configurable window operator module 150 on input relations in a CQL query, in at least some embodiments.

[00105] The output relation module 152 may be configured to generate one or more output relation(s) 158. In one embodiment, the output relation module 152 may be configured to receive a CQL query from a CQL Engine and/or CQ Service. In certain embodiments, the output relation module 152 may identify an input relation in the CQL query, identify a configurable window operator defined over the input relation and execute the CQL query based at least in part on the configurable window operator to generate an output relation. Alternatively, in some examples, when a query (e.g., a CQL query) is identified or received that includes an input relation, the CQL engine and/or CQ Service may parse the query 206 to process the input relation and generate the output relation. In one embodiment, and as will be discussed in detail below, the output relation includes a subset of data records from the input relation whose attribute values lie within a specified range defined by the configurable window operator in the input relation. In certain embodiments, a CQL query may then be received to process the subset of the data records in the output relation in memory.

[00106] FIG. 2 illustrates a simplified block diagram with which features for the management of value-based windows on relations may be described. As shown, FIG. 2 describes at least one implementation of a CQL Engine and/or CQ Service 200 for managing an input relation 154. The CQL Engine and/or CQ Service 200 may initially receive information from an input source 202. In one example, the input source 202 may include the data source computers 110 that receive an incoming continuous input data stream that includes a stream of data or events related to the application. In one example, the CQL Engine and/or CQ Service 200 may then identify an input relation 154, which may be a representation of data from the input source 202. In some examples, and as discussed above, the input source 202 may include a bounded set of data records related to an application. In a certain embodiment, when a query (e.g., a continuous query) is identified or received that includes an input relation 154, the CQL engine 200 may parse the query to identify the configurable window operator 156 in the query and execute the query based at least in part on the configurable window operator to generate an output relation 158. In one embodiment, the CQL engine and/or CQ Service 200 may execute the query by applying the configurable window operator on an attribute in the input relation to generate the output relation. In some examples, the output relation may be a representation of data (e.g., a subset of data records) related to the application. The CQL Engine and/or CQ Service 200 may

then store the output relation in an output destination 204, such as for example, in the databases 112 shown in Fig. 1.

[00107] FIG. 3 illustrates exemplary information stored in an input relation and the generation of an output relation when the input relation is processed using a configurable window operator, in accordance with one embodiment of the present disclosure. As an example, consider an input relation, "CallCentreRecords" that may be generated as follows:

[00108] CREATE RELATION CallCentreRecords(callerName char(50), callTime, timestamp).

[00109] Per this definition, the input relation, "CallCentreRecords" contains two attributes, a first attribute "callerName", a second attribute, "callTime" and a timestamp. In one example, the input relation, "CallCentreRecords" may store continuous data streams related to call detail records from a call center. As used herein, a "call center" typically refers to a service network in which customer service representatives may provide services to customers via telephones. Continuous data streams from a call center may typically include call-by-call information related to a call that is summarized over short time-intervals. Such call-by-call information may include, for example, a timestamp for when a call was recorded or received in the database, a caller name, the time that the call was placed, and the like.

[00110] Referring to the example shown in FIG. 3, in one embodiment, the input relation, "CallCenterRecords", 302, includes columns identifying for each timestamp at which a data record was inserted into the input relation 302, a "callerName" attribute which identifies the name of the caller who placed the call, and a "callTime" attribute which identifies the date and time that the call was placed.

[00111] In certain embodiments, a CQL query may be identified or received that includes the "CallCenterRecords", input relation 302 as follows:

[00112] SELECT * FROM CallCentreRecords [RANGE 1 HOUR ON callTime]

[00113] As per the CQL query shown above, the configurable window operator is identified as a "generic value" window operator 303. The CQl query is then executed based at least in part

on the "generic value" window operator 303 by applying the "RANGE 1 HOUR" window operator on the attribute "callTime" in the "CallCenterRecords", input relation 302 to generate an output relation. An output relation, 304, that is generated as a result of applying the "RANGE 1 HOUR" window operator 303 is also shown in FIG. 3.

[00114] In one embodiment, and as illustrated in FIG. 3, the output relation 304 includes a subset of data records from the input relation 302, whose attribute value, "callTime", lies in the specified range, "1 HOUR" with respect to the timestamp t of the data records. In the example shown in FIG. 3, in one embodiment, the "CallCenterRecords" output relation 304 also includes a column identifying for each data record, a "SIGN" attribute which indicates whether a particular data record from the input relation 302 was inserted into the output relation 304 or removed from the output relation 304 as a result of applying the "generic value" window operator 303 on the input relation 302. The manner in which the "CallCenterRecords" output relation 304 is generated is discussed in detail below.

[00115] Consider the first data record in the "CallCenterRecords" input relation 302. As per the information in the "CallCenterRecords" input relation 302, this record has a timestamp of 21/01/2012 9:00:00 AM and indicates that a call was placed by a caller "Robin" at 21/01/2012 8:30:00 AM. When the "RANGE 1 HOUR" window operator 303 is applied on the "callTime" attribute, it is determined whether the first data record is to be inserted into the CallCenterRecords" output relation 304. In this example, the first data record in the "CallCenterRecords" input relation 302 will be inserted into the "CallCenterRecords" output relation 304 because the call time (8:30:00AM) of this call was placed within a 1 hour range from the current timestamp of 21/01/2012 9:00:00 AM, wherein the 1 hour range from the current timestamp is defined to be (8.00 AM – 9.00 AM) for this data record. In one embodiment, and as illustrated in FIG. 3B, the insertion of this data record into the "CallCenterRecords" output relation 304 is indicated by a + entry in the SIGN column corresponding to the data record.

[00116] Similarly, the second data record in the "CallCenterRecords" input relation 302 which indicates that a call was placed by caller "Michael" at 21/01/2012 8:45:00 AM will also be inserted into the "CallCenterRecords" output relation 304 since the call time (8:45:00AM) of this

call was also placed within a 1 hour range from the current timestamp of 21/01/2012 9:00:00 AM, wherein the 1 hour range from the current timestamp is defined to be (8:00:00 AM – 9:00:00 AM) for this data record.

[00117] The next data record in the "CallCenterRecords" input relation 302 indicates that the current time stamp has now moved to 21/01/2012 10:00:00 AM. It may further be observed that two data records have been inserted into the input relation 302 at the current time stamp of 10:00:00 AM. These records include a caller, "Sandeep" whose call was placed at 21/01/2012, 9:06:00 AM and a caller "Anand" whose call was placed at 21/01/2012 9:30:00 AM. Since, these records lie within the 1 hour range of the current timestamp of 21/01/2012 10:00:00 AM, wherein the 1 hour range from the current timestamp is defined to be (9:00:00 AM – 10:00:00 AM) they are also inserted into the "CallCenterRecords" output relation 304 as indicated by the + operator in the SIGN column next to these data records. However, it may be observed that the data records which include callers, "Robin" and "Michael" now no longer lie within the 1 hour range (9:00:00 AM – 10:00:00 AM)from the current timestamp. Therefore, these records are removed from the "CallCenterRecords" output relation 304 at the current timestamp of 10:00:00 AM. In one embodiment, and as illustrated in FIG. 3, the deletion of these data records from the "CallCenterRecords" output relation 304 is indicated by a - entry in the SIGN column corresponding to the data records.

[00118] Now, the current time stamp has moved to 21/01/2012, 3:00:00 PM. The data records in the "CallCenterRecords" input relation 302 at this time stamp include caller, "Unmesh" whose call was placed at 21/01/2012 11:00:00 AM and caller, "Alex" whose call was placed at 21/01/2012 1:15:00 PM. However, these records are not inserted into the output relation 304 since the call times of these calls were not placed within the 1 hour range of the current timestamp of 21/01/2012, 3:00:00 PM, wherein the 1 hour range from the current timestamp is defined to be (2:00:00 PM – 3:00:00 PM). However caller "Sundar" whose call was placed at 21/01/2012 2:30:00 PM is inserted into the "CallCenterRecords" output relation 304 as this call was placed within the 1 hour range of the current timestamp of 21/01/2012, 3:00:00 PM. Similarly, it may be observed that the data records which include callers, "Sandeep" and "Anand" now no longer lie within the 1 hour range from the current timestamp of 21/01/2012,

3:00:00 PM. Therefore, these records are removed from the "CallCenterRecords" output relation 304 as indicated by a - entry in the SIGN column corresponding to the data records.

[00119] It is to be appreciated that the information listed in the "CallCenterRecords" input relation 302 and the "CallCenterRecords" output relation 304 is merely provided by way of example and is not intended to limit the scope of the present disclosure. One of ordinary skill in the art would recognize many variations, modifications, and alternatives.

[00120] FIG. 4 illustrates exemplary information stored in an input relation and the generation of an output relation when the input relation is processed using a configurable window operator, in accordance with another embodiment of the present disclosure. FIG. 4 illustrates exemplary information stored in a "CallCenterRecords" input relation 402, in accordance with one embodiment of the present disclosure. In the example shown in FIG. 4, in one embodiment, the input relation, "CallCenterRecords", 402, includes columns identifying for each timestamp at which a data record was inserted into the input relation, a "callerName" attribute which identifies the name of the caller who placed the call, and a "callTime" attribute which identifies the date and time that the call was placed.

[00121] In certain embodiments, a CQL query may be identified or received that includes the "CallCenterRecords", input relation 402 as follows:

[00122] SELECT callerName, callTime FROM CallCentreRecords[CurrentHour ON callTime]

[00123] As per the CQL query shown above, the configurable window operator is identified as a "current hour" window operator 403. The CQL query is then executed based at least in part on the "current hour" window operator by applying the "CurrentHour" window operator 403 on the attribute "callTime" in the "CallCenterRecords", input relation 402 to generate an output relation. An output relation 404 that is generated as a result of applying the "CurrentHour" window operator 403 is also shown in FIG. 4.

[00124] In one embodiment, and as illustrated in FIG. 4, the output relation 404 includes a subset of data records from the input relation 402, whose attribute value, "callTime", lies in the current hour with respect to the timestamp t of the data records. In the example shown in FIG. 4,

in one embodiment, the "CallCenterRecords" output relation 404 also includes a column identifying for each data record, a "SIGN" attribute which indicates whether a particular data record from the input relation 402 was inserted into the output relation 404 or removed from the output relation 404 as a result of applying the "CurrentHour" window operator 403 on the input relation 402.

[00125] FIG. 5 illustrates exemplary information stored in an input relation and the generation of an output relation when the input relation is processed using a configurable window operator, in accordance with another embodiment of the present disclosure. FIG. 5 illustrates exemplary information stored in a "CallCenterRecords" input relation 402, in accordance with one embodiment of the present disclosure. In the example shown in FIG. 5, in one embodiment, the input relation, "CallCenterRecords", 502, includes columns identifying for each timestamp at which a data record was inserted into the input relation, a "callerName" attribute which identifies the name of the caller who placed the call, and a "callTime" attribute which identifies the date and time that the call was placed.

[00126] In certain embodiments, a CQL query may be identified or received that includes the "CallCenterRecords", input relation 502 as follows:

[00127] SELECT callerName, callTime FROM CallCentreRecords[CurrentPeriod("0900", "1500" ON callTime]

[00128] As per the CQL query shown above, the configurable window operator is identified as a "current period" window operator 503. The CQL query is then executed based at least in part on the "CurrentPeriod" window operator 503 by applying the "CurrentPeriod" window operator 503 on the attribute "callTime" in the "CallCenterRecords", input relation 502 to generate an output relation. An output relation 504 that is generated as a result of applying the "CurrentPeriod" window operator 503 is also shown in FIG. 5.

[00129] In one embodiment, and as illustrated in FIG. 5, the output relation 504 may include a subset of data records from the input relation 502, whose attribute value, "callTime", lies in the current period, CurrentPeriod("0900", "1500") with respect to the timestamp t of the data records. In the example shown in FIG. 5, in one embodiment, the "CallCenterRecords" output

relation 504 also includes a column identifying for each data record, a "SIGN" attribute which indicates whether a particular data record from the input relation 502 was inserted into the output relation 504 or removed from the output relation 504 as a result of applying the CurrentPeriod("0900", "1500") window operator 503 on the input relation 502.

[00130] The disclosed technique enables the processing of incoming real-time information related to an application stored in an external data source (e.g., relation) using continuous queries (e.g., CQL query). This incoming real-time information is typically processed using an in-memory real-time event processing engine (e.g., a CQL engine). However, as the amount of information related to the application received in real-time increases, the size of the relation that is stored in memory also increases. Processing such a big sized relation in limited memory using continuous queries may not be scalable since this may involve joining a high rate input stream with a big sized relation in limited memory.

[00131] The disclosed technique enables a subset of data records of the relation to be queried in limited memory rather than processing the entire set of data records of the relation in memory by defining a window operator in the continuous query (e.g., a CQL query). A continuous query (e.g., a CQL query) is applied over the input relation for the duration of the window operator to generate an output relation. A set of data records related to the output relation is provided to the user. The window operator is configurable by a user and enables processing a subset of data records in the relation over a specified range of time. Therefore, a subset of data records of the relation may be queried in limited memory rather than processing the entire set of data records of the relation in memory.

[00132] The disclosed technique may be applied to data related to real-world applications that may the form of continuous, unbounded data streams. Examples of such data streams may include stock tickers in financial applications, performance measurements in network monitoring and traffic management, log records or click-streams in web tracking and personalization, data feeds from sensor applications, network packets and messages in firewall-based security, call detail records in telecommunications, and the like.

[00133] FIGS. 6-7 illustrate example flow diagrams showing respective processes 600 and 700 for implementing the management of value-based windows on relations described herein. These processes 600 and 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.

[00134] 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.

[00135] In some examples, the one or more service provider computers 106 (e.g., utilizing at least the input relation module 148, the configurable window operator module 150 or the output relation module 152) shown in at least FIG. 1 (and others) may perform the process 600 of FIG. 6. The process 600 may begin at block 602 by including generating an input relation. At block 604, the process 600 may include identifying a query (e.g., a CQL query) to process the input relation. At block 606, the process 600 may include identifying a configurable window operator associated with processing the input relation. Further, in some examples, the process 600 may end at block 608 by including executing the query based at least in part on the configurable window operator to generate an output relation.

[00136] FIG. 7 illustrates an example flow diagram showing process 700 for implementing the management of value-based windows on relations described herein. The one or more service provider computers 106 (e.g., utilizing at least the input relation module 148, the configurable window operator module 150 or the output relation module 152) shown in at least FIG. 1 (and others) may perform the process 700 of FIG. 7. The process 700 may begin at block 702 by including generating an input relation. In one embodiment, the input relation is a bounded set of data records related to an application. At block 704, the process 700 may include receiving a continuous query (e.g., a CQL query) that identifies the input relation. At block 706, the process 700 may include identifying a configurable window operator associated with processing the input relation. Further, in some examples, the process 700 may include executing the continuous query based at least in part on the configurable window operator to generate an output relation at block 708. Further, the process 700 may end, at block 710, by including providing the data records of the output relation based at least in part on execution of the continuous query.

[00137] FIG. 8 is a simplified block diagram illustrating components of a system environment 800 that may be used in accordance with an embodiment of the present disclosure. As shown, system environment 800 includes one or more client computing devices 802, 804, 806, 808, 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 810 (such as, but not limited to, networks similar to the networks 108 of FIG. 1). In various embodiments, client computing devices 802, 804, 806, and 808 may interact with a server 812 over the networks 810.

[00138] Client computing devices 802, 804, 806, and 808 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 802, 804, 806, and 808 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 810 described below). Although exemplary system environment 800 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 812.

[00139] System environment 800 may include networks 810. Networks 810 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 1510 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.

[00140] System environment 800 also includes one or more server computers 812 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 812 may be adapted to run one or more services or software applications described in the foregoing disclosure. For example, server 812 may correspond to a server for performing processing described above according to an embodiment of the present disclosure.

[00141] Server 812 may run an operating system including any of those discussed above, as well as any commercially available server operating system. Server 812 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.

[00142] System environment 800 may also include one or more databases 814, 816. Databases 814, 816 may reside in a variety of locations. By way of example, one or more of databases 814, 816 may reside on a non-transitory storage medium local to (and/or resident in) server 812. Alternatively, databases 814, 816 may be remote from server 812, and in communication with server 812 via a network-based or dedicated connection. In one set of embodiments, databases 814, 816 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 812 may be stored locally on server 812 and/or remotely, as appropriate. In one set of embodiments, databases 814, 816 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.

[00143] FIG. 9 is a simplified block diagram of a computer system 900 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 900. Computer system 900 is shown comprising hardware elements that may be electrically and/or communicatively coupled via a bus 901. The hardware elements may include one or more central processing units (CPUs) 902, one or more input devices 904 (e.g., a mouse, a keyboard, etc.), and one or more output devices 906 (e.g., a display device, a printer, etc.). Computer system 900 may also include one or more storage devices 908. By way of example, the storage device(s) 908 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.

[00144] Computer system 900 may additionally include a computer-readable storage media reader 912, a communications subsystem 914 (e.g., a modem, a network card (wireless or wired), an infra-red communication device, etc.), and working memory 918, which may include RAM and ROM devices as described above. In some embodiments, computer system 900 may also include a processing acceleration unit 916, which can include a digital signal processor (DSP), a special-purpose processor, and/or the like.

[00145] Computer-readable storage media reader 912 can further be connected to a computer-readable storage medium 910, together (and, optionally, in combination with storage device(s)

908) 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 914 may permit data to be exchanged with network 912 and/or any other computer described above with respect to system environment 900.

[00146] Computer system 900 may also comprise software elements, shown as being currently located within working memory 918, including an operating system 920 and/or other code 922, such as an application program (which may be a client application, Web browser, mid-tier application, RDBMS, etc.). In an exemplary embodiment, working memory 918 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 900 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.

[00147] FIG. 10 illustrates an example flow diagram showing process 1000 for implementing the management of value-based windows on relations described herein, in accordance with another embodiment of the present invention. The one or more service provider computers 106 (e.g., utilizing at least the input relation module 148, the configurable window operator module 150 or the output relation module 152) shown in at least FIG. 1 (and others) may perform the process 1000 of FIG. 10. The process 1000 may begin at block 1002 by including identifying a query (e.g., a continuous query) configured to process an input relation. At block 1004, the process 1000 may include starting the query based on a configurable window operator to generate an output relation. At block 1006, the process 1000 may include generating an input event and sending it to the CQL Engine and/or CQ Service 200. In some examples, the input event may be generated when a new event is received at the CQL Engine and/or CQ Service 200. For example, an input event may include a new call placed by a caller in the "CallCentreRecords" input relation 302 discussed above. At block 1008, the process 1000 may include receiving the input event and the configurable window operator and producing an output event. The output event may include, for example, a data record that is inserted into the

"CallCenterRecords" output relation 304 when the configurable window operator is applied on the "calltime" attribute in the "CallCenterRecords", input relation 302.

[00148] FIG. 11 is a simplified block diagram of a service provider device 1101 that may be used in accordance with certain embodiments of the present invention. The blocks of the service provider device 1101 may be implemented by hardware, software, or a combination of hardware and software to carry out the principles of the invention. It is understood by persons of skill in the art that the blocks described in FIG. 11 may be combined or separated into sub-blocks to implement the principles of the invention as described above. Therefore, the description herein may support any possible combination or separation or further definition of the functional blocks described herein.

[00149] As shown in FIG. 11, the service provider device 1101 may comprise an input relation unit 1102, a configurable window operator unit 1103, an output relation unit 1105, and a providing unit 1107. The input relation unit 1102 may be configured to receive a continuous query that identifies an input relation, wherein the input relation is a bounded set of data records related to an application. The configurable window operator unit 1103 may be configured to identify a configurable window operator associated with processing the input relation. The output relation unit 1105 may be configured to execute the continuous query based at least in part on the configurable window operator to generate an output relation. The providing unit 1107 may be configured to provide data records of the output relation based at least in part on execution of the continuous query. In one embodiment, each unit may be implemented as a processor that performs corresponding process by reading computer program instructions.

[00150] In one embodiment, the input relation may be an external data source generated based at least in part on an incoming continuous input data stream related to an application.

[00151] In one embodiment, the input relation may be an external data source generated based at least in part on information related to an application stored in a database of historical data.

[00152] In one embodiment, the input relation may be an external data source generated based at least in part on one or more archived relations related to an application.

[00153] In one embodiment, the configurable window operator may be a generic value window operator defined over the input relation, and wherein the output relation unit 1105 may be further configured to apply the generic value window operator on an attribute in the input relation to generate the output relation.

[00154] In one embodiment, the configurable window operator may be a current hour value window operator defined over the input relation, and wherein the output relation unit 1105 may be further configured to apply the current hour value window operator on an attribute in the input relation to generate the output relation.

[00155] In one embodiment, the configurable window operator may be a current period value window operator defined over the input relation, and wherein the output relation unit 1105 may be further configured to apply the current period value window operator on an attribute in the input relation to generate the output relation.

[00156] In one embodiment, the providing unit 1107 may be further configured to display the output relation, wherein the output relation comprises a subset of the data records from the input relation whose attribute values lie within a specified range defined by the configurable window operator.

[00157] FIG. 12 is a simplified block diagram of a service provider device 1201 that may be used in accordance with certain embodiments of the present invention. The blocks of the service provider device 1201 may be implemented by hardware, software, or a combination of hardware and software to carry out the principles of the invention. It is understood by persons of skill in the art that the blocks described in FIG. 12 may be combined or separated into sub-blocks to implement the principles of the invention as described above. Therefore, the description herein may support any possible combination or separation or further definition of the functional blocks described herein.

[00158] As shown in FIG.12, the service provider device 1201 may comprises an input relation unit 1202, a configurable window operator unit 1203, and an output relation unit 1205. The input relation unit 1202 may be configured to receive and identify a query configured to process an input relation. The configurable window operator unit 1203 may be configured to

identify a configurable window operator associated with processing the input relation. The output relation unit 1205 may be configured to execute the query based at least in part on the configurable window operator to generate an output relation. In one embodiment, each unit may be implemented as a processor that performs corresponding process by reading computer program instructions.

[00159] In one embodiment, the input relation may be an external data source generated based at least in part on an incoming continuous input data stream related to the application.

[00160] In one embodiment, the input relation may be an external data source generated based at least in part on information related to the application stored in a database of historical data.

[00161] In one embodiment, the input relation may be an external data source generated based at least in part on one or more archived relations related to the application.

[00162] In one embodiment, the configurable window operator may be a generic value window operator defined over the input relation, and the output relation unit 1205 may be further configured to apply the generic value window operator on an attribute in the input relation to generate the output relation.

[00163] In one embodiment, the service provider device 1201 may further comprise a providing unit 1207 configured provide data records of the output relation based at least in part on execution of the continuous query.

[00164] In one embodiment, the providing unit 1207 may be further configured to display the output relation, wherein the output relation comprises a subset of the data records from the input relation whose attribute values lie within a specified range defined by the configurable window operator.

[00165] 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.

[00166] 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. 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.

[00167] 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.

[00168] 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.

[00169] 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 computer-implemented method , comprising:

generating an input relation, the input relation being a bounded set of data records related to an application;

receiving a continuous query that identifies the input relation;

identifying a configurable window operator associated with processing the input relation;

executing the continuous query based at least in part on the configurable window operator to generate an output relation; and

providing data records of the output relation based at least in part on execution of the continuous query.

2. The computer-implemented method of claim 1, wherein the input relation is an external data source generated based at least in part on an incoming continuous input data stream related to the application.

3. The computer-implemented method of claim 1 or 2, wherein the input relation is an external data source generated based at least in part on information related to the application stored in a database of historical data.

4. The computer-implemented method of claim 1 or 2 or 3, wherein the input relation is an external data source generated based at least in part on one or more archived relations related to the application.

5. The computer-implemented method of any one of claims 1-4, wherein the configurable window operator is a generic value window operator defined over the input relation, and wherein executing the continuous query comprises applying the generic value window operator on an attribute in the input relation to generate the output relation.

6. The computer-implemented method of any one of claims 1-5, wherein the configurable window operator is a current hour value window operator defined over the input

relation, and wherein executing the continuous query comprises applying the current hour value window operator on an attribute in the input relation to generate the output relation.

7. The computer-implemented method of any one of claims 1-6, wherein the configurable window operator is a current period value window operator defined over the input relation, and wherein executing the continuous query comprises applying the current period value window operator on an attribute in the input relation to generate the output relation.

8. The computer-implemented method of any one of claims 1-7, wherein providing the data records comprises displaying the output relation, wherein the output relation comprises a subset of the data records from the input relation whose attribute values lie within a specified range defined by the configurable window operator.

9. A non-transitory computer-readable memory 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 generate an input relation;

instructions that cause the one or more processors to identify a query configured to process the input relation;

instructions that cause the one or more processors to identify a configurable window operator associated with processing the input relation; and

instructions that cause the one or more processors to execute the query based at least in part on the configurable window operator to generate an output relation.

10. The computer-readable memory of claim 9, wherein the input relation is an external data source generated based at least in part on an incoming continuous input data stream related to the application.

11. The computer-readable memory of claim 9 or 10, wherein the input relation is an external data source generated based at least in part on information related to the application stored in a database of historical data.

12. The computer-readable memory of claim 9 or 10 or 11, wherein the input relation is an external data source generated based at least in part on one or more archived relations related to the application.

13. The computer-readable memory of any one of claims 9-12, wherein the configurable window operator is a generic value window operator defined over the input relation, and wherein executing the continuous query comprises applying the generic value window operator on an attribute in the input relation to generate the output relation.

14. The computer-readable memory of any one of claims 9-13, further comprising instructions to provide data records of the output relation based at least in part on execution of the continuous query.

15. The computer-readable memory of any one of claims 9-14, further comprising instructions to display the output relation, wherein the output relation comprises a subset of the data records from the input relation whose attribute values lie within a specified range defined by the configurable window operator.

16. 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:

generate an input relation;

identify a query configured to process the input relation;

identify a configurable window operator associated with processing the input relation; and

execute the query based at least in part on the configurable window operator to generate an output relation.

17. The system of claim 16, wherein the input relation is an external data source generated based at least in part on an incoming continuous input data stream related to the application.

18. The system of claim 16 or 17, wherein the configurable window operator is a current period value window operator defined over the input relation, and wherein executing the continuous query comprises applying the current period value window operator on an attribute in the input relation to generate the output relation.

19. The system of claim 16 or 17 or 18, wherein the one or more processors are further configured to execute the plurality of instructions to provide data records of the output relation based at least in part on execution of the continuous query.

20. The system of any one of claims 16-19, wherein the one or more processors are further configured to display the output relation, wherein the output relation comprises a subset of the data records from the input relation whose attribute values lie within a specified range defined by the configurable window operator.

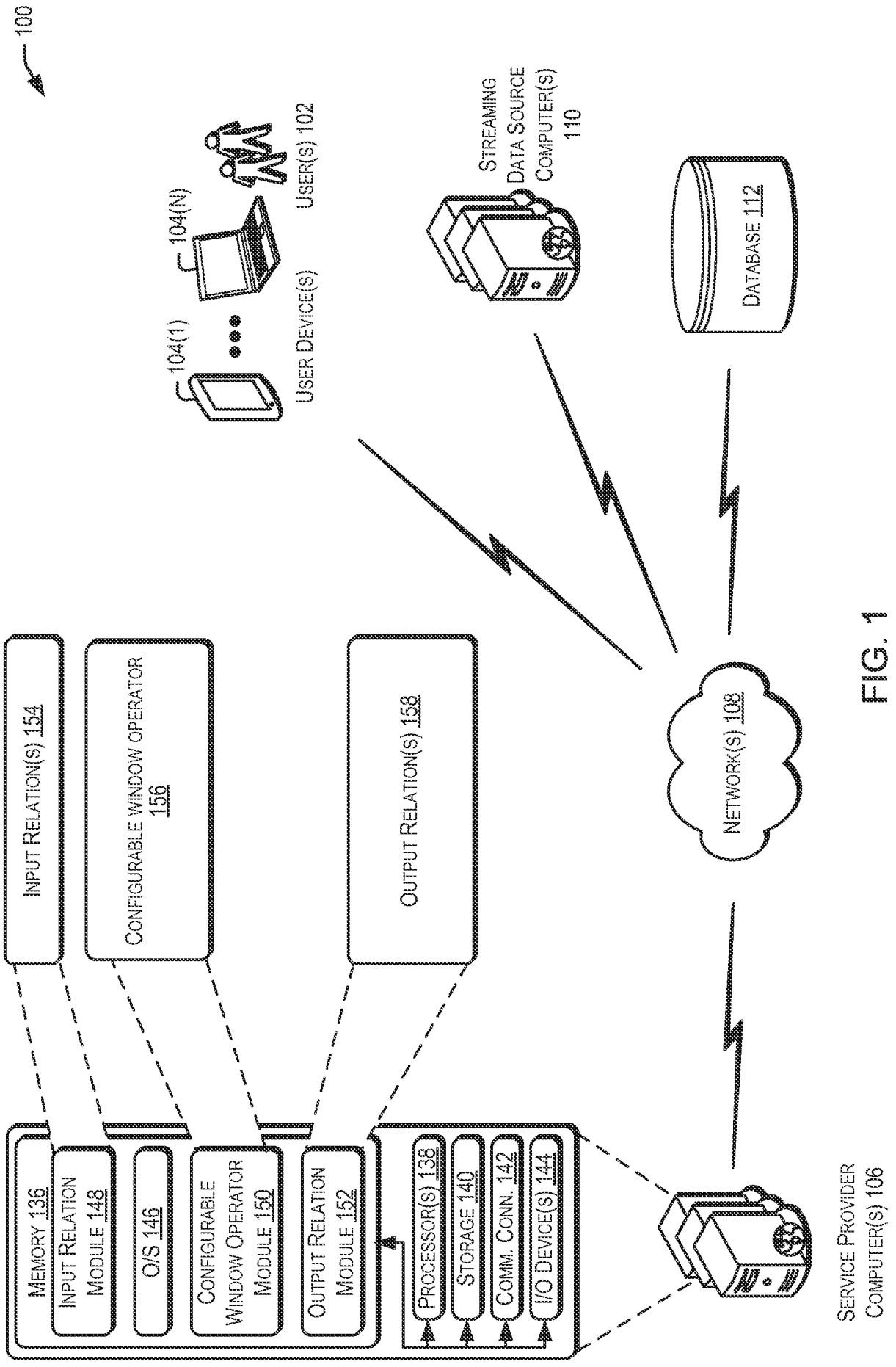

100
104(1)
104(N)
USER DEVICE(S)
USER(S) 102
STREAMING DATA SOURCE COMPUTER(S) 110
DATABASE 112
NETWORK(S) 108
SERVICE PROVIDER COMPUTER(S) 106
MEMORY 136
INPUT RELATION MODULE 148
O/S 146
CONFIGURABLE WINDOW OPERATOR MODULE 150
OUTPUT RELATION MODULE 152
PROCESSOR(S) 138
STORAGE 140
COMM. CONN. 142
I/O DEVICE(S) 144
INPUT RELATION(S) 154
CONFIGURABLE WINDOW OPERATOR 156
OUTPUT RELATION(S) 158

FIG. 1

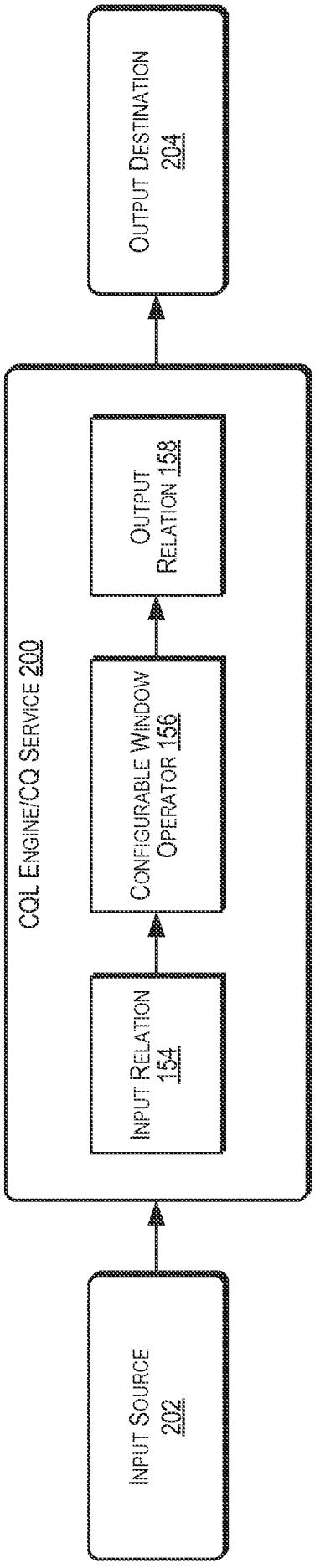
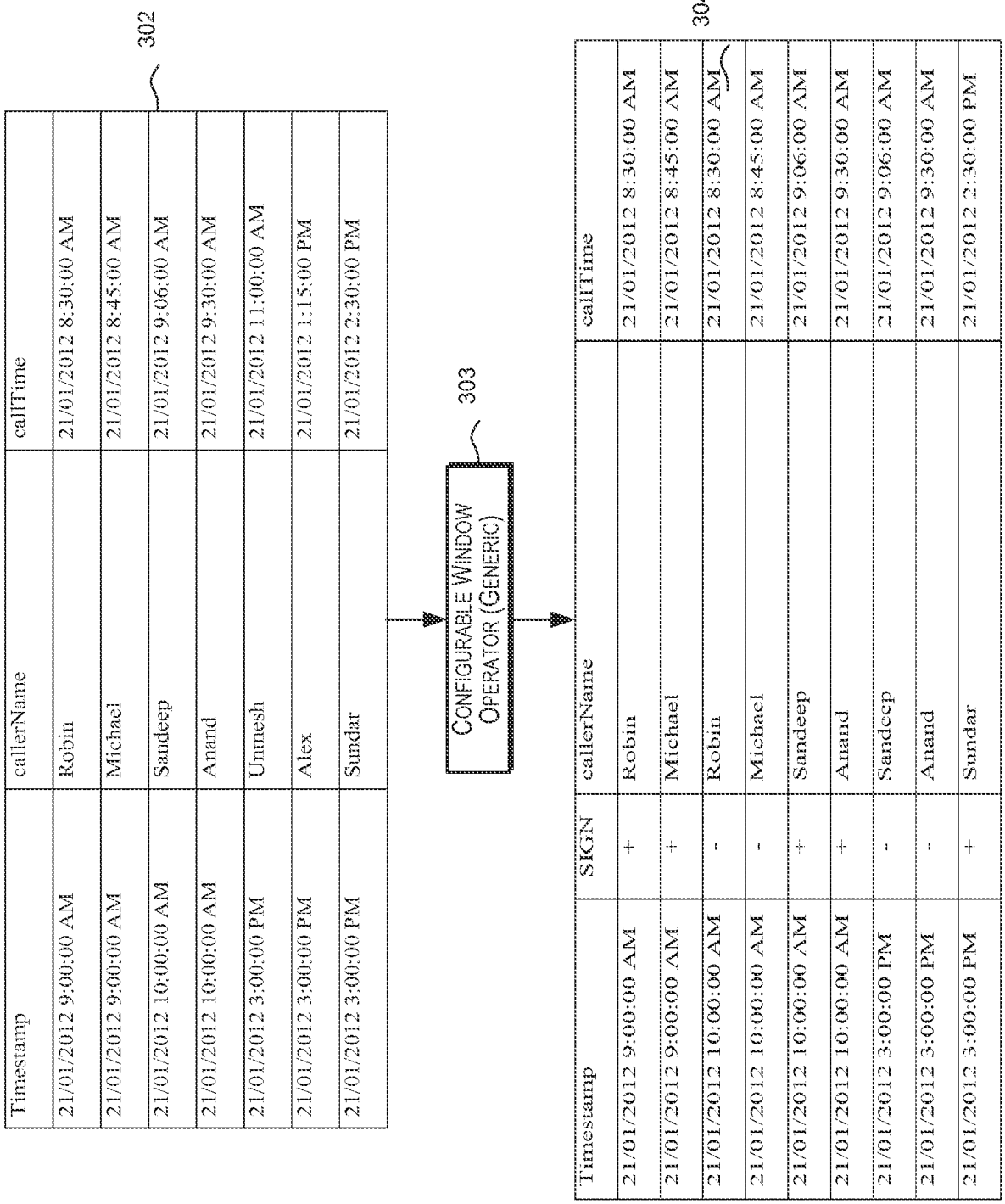

INPUT SOURCE 202
CQL ENGINE/CQ SERVICE 200
INPUT RELATION 154
CONFIGURABLE WINDOW OPERATOR 156
OUTPUT RELATION 158
OUTPUT DESTINATION 204

FIG. 2

+



Timestamp	callerName	callTime
21/01/2012 9:00:00 AM	Robin	21/01/2012 8:30:00 AM
21/01/2012 9:00:00 AM	Michael	21/01/2012 8:45:00 AM
21/01/2012 10:00:00 AM	Sandeep	21/01/2012 9:06:00 AM
21/01/2012 10:00:00 AM	Anand	21/01/2012 9:30:00 AM
21/01/2012 3:00:00 PM	Unmesh	21/01/2012 11:00:00 AM
21/01/2012 3:00:00 PM	Alex	21/01/2012 1:15:00 PM
21/01/2012 3:00:00 PM	Sundar	21/01/2012 2:30:00 PM

Timestamp	SIGN	callerName	callTime
21/01/2012 9:00:00 AM	+	Robin	21/01/2012 8:30:00 AM
21/01/2012 9:00:00 AM	+	Michael	21/01/2012 8:45:00 AM
21/01/2012 10:00:00 AM	-	Robin	21/01/2012 8:30:00 AM
21/01/2012 10:00:00 AM	-	Michael	21/01/2012 8:45:00 AM
21/01/2012 10:00:00 AM	+	Sandeep	21/01/2012 9:06:00 AM
21/01/2012 10:00:00 AM	+	Anand	21/01/2012 9:30:00 AM
21/01/2012 3:00:00 PM	-	Sandeep	21/01/2012 9:06:00 AM
21/01/2012 3:00:00 PM	-	Anand	21/01/2012 9:30:00 AM
21/01/2012 3:00:00 PM	+	Sundar	21/01/2012 2:30:00 PM

FIG. 3

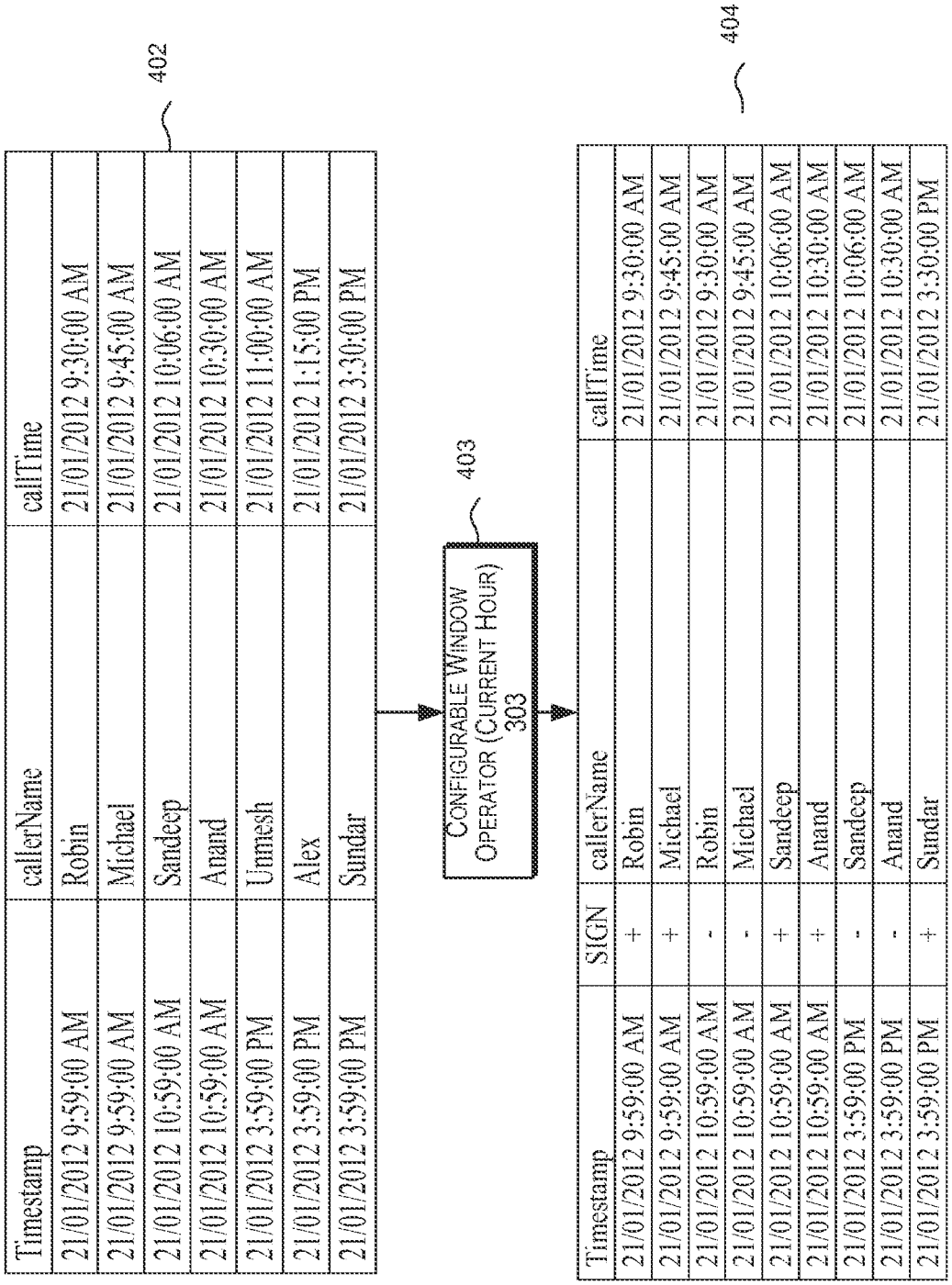
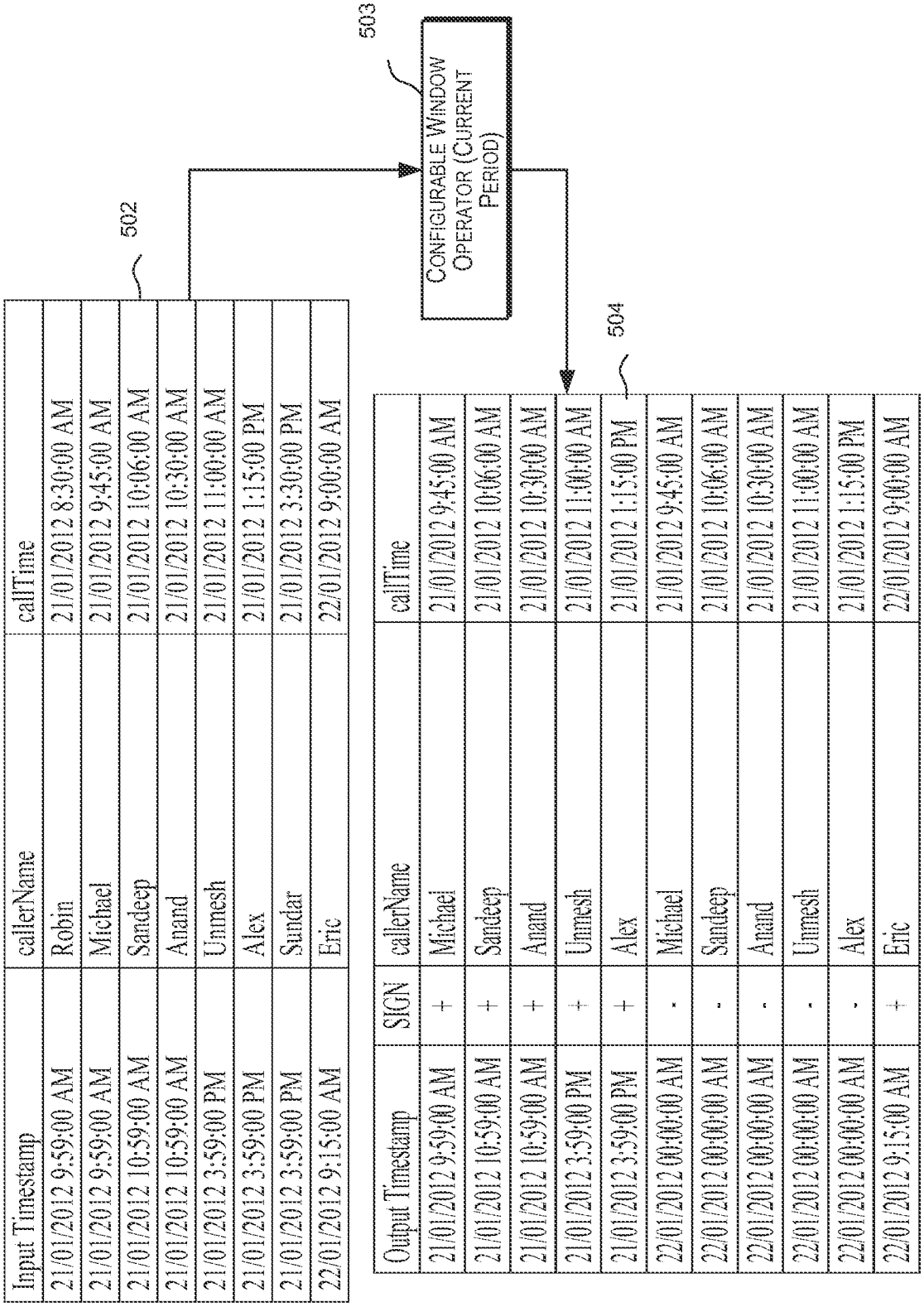

402
Timestamp
callerName
callTime
21/01/2012 9:59:00 AM
Robin
21/01/2012 9:30:00 AM
21/01/2012 9:59:00 AM
Michael
21/01/2012 9:45:00 AM
21/01/2012 10:59:00 AM
Sandeep
21/01/2012 10:06:00 AM
21/01/2012 10:59:00 AM
Anand
21/01/2012 10:30:00 AM
21/01/2012 3:59:00 PM
Unmesh
21/01/2012 11:00:00 AM
21/01/2012 3:59:00 PM
Alex
21/01/2012 1:15:00 PM
21/01/2012 3:59:00 PM
Sundar
21/01/2012 3:30:00 PM
403
CONFIGURABLE WINDOW OPERATOR (CURRENT HOUR) 303
404
Timestamp
SIGN
callerName
callTime
21/01/2012 9:59:00 AM
+
Robin
21/01/2012 9:30:00 AM
21/01/2012 9:59:00 AM
+
Michael
21/01/2012 9:45:00 AM
21/01/2012 10:59:00 AM
-
Robin
21/01/2012 9:30:00 AM
21/01/2012 10:59:00 AM
-
Michael
21/01/2012 9:45:00 AM
21/01/2012 10:59:00 AM
+
Sandeep
21/01/2012 10:06:00 AM
21/01/2012 10:59:00 AM
+
Anand
21/01/2012 10:30:00 AM
21/01/2012 3:59:00 PM
-
Sandeep
21/01/2012 10:06:00 AM
21/01/2012 3:59:00 PM
-
Anand
21/01/2012 10:30:00 AM
21/01/2012 3:59:00 PM
+
Sundar
21/01/2012 3:30:00 PM

FIG. 4



Input Timestamp	callerName	callTime
21/01/2012 9:59:00 AM	Robin	21/01/2012 8:30:00 AM
21/01/2012 9:59:00 AM	Michael	21/01/2012 9:45:00 AM
21/01/2012 10:59:00 AM	Sandeep	21/01/2012 10:06:00 AM
21/01/2012 10:59:00 AM	Anand	21/01/2012 10:30:00 AM
21/01/2012 3:59:00 PM	Unmesh	21/01/2012 11:00:00 AM
21/01/2012 3:59:00 PM	Alex	21/01/2012 1:15:00 PM
21/01/2012 3:59:00 PM	Sundar	21/01/2012 3:30:00 PM
22/01/2012 9:15:00 AM	Eric	22/01/2012 9:00:00 AM

Output Timestamp	SIGN	callerName	callTime
21/01/2012 9:59:00 AM	+	Michael	21/01/2012 9:45:00 AM
21/01/2012 10:59:00 AM	+	Sandeep	21/01/2012 10:06:00 AM
21/01/2012 10:59:00 AM	+	Anand	21/01/2012 10:30:00 AM
21/01/2012 3:59:00 PM	+	Unmesh	21/01/2012 11:00:00 AM
21/01/2012 3:59:00 PM	+	Alex	21/01/2012 1:15:00 PM
22/01/2012 00:00:00 AM	-	Michael	21/01/2012 9:45:00 AM
22/01/2012 00:00:00 AM	-	Sandeep	21/01/2012 10:06:00 AM
22/01/2012 00:00:00 AM	-	Anand	21/01/2012 10:30:00 AM
22/01/2012 00:00:00 AM	-	Unmesh	21/01/2012 11:00:00 AM
22/01/2012 00:00:00 AM	-	Alex	21/01/2012 1:15:00 PM
22/01/2012 9:15:00 AM	+	Eric	22/01/2012 9:00:00 AM

FIG. 5

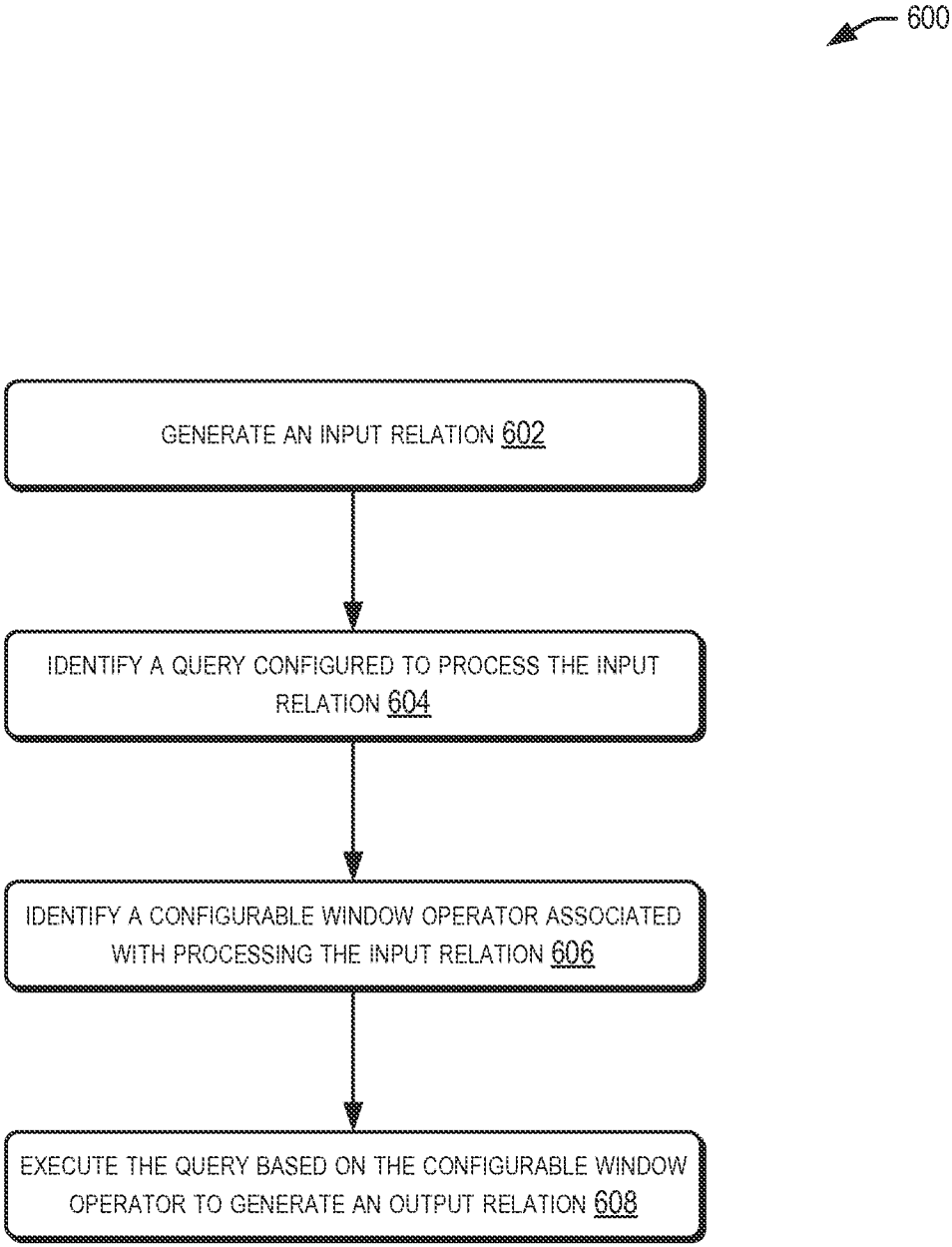

600
GENERATE AN INPUT RELATION 602
IDENTIFY A QUERY CONFIGURED TO PROCESS THE INPUT RELATION 604
IDENTIFY A CONFIGURABLE WINDOW OPERATOR ASSOCIATED WITH PROCESSING THE INPUT RELATION 606
EXECUTE THE QUERY BASED ON THE CONFIGURABLE WINDOW OPERATOR TO GENERATE AN OUTPUT RELATION 608

FIG. 6

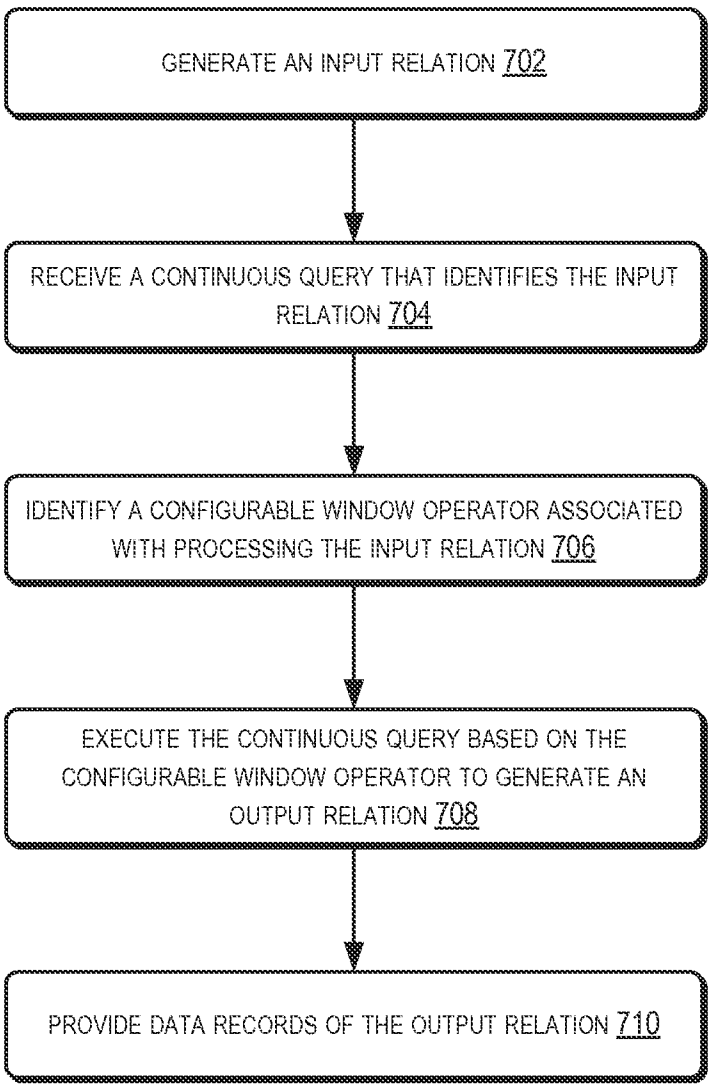

700
GENERATE AN INPUT RELATION 702
RECEIVE A CONTINUOUS QUERY THAT IDENTIFIES THE INPUT RELATION 704
IDENTIFY A CONFIGURABLE WINDOW OPERATOR ASSOCIATED WITH PROCESSING THE INPUT RELATION 706
EXECUTE THE CONTINUOUS QUERY BASED ON THE CONFIGURABLE WINDOW OPERATOR TO GENERATE AN OUTPUT RELATION 708
PROVIDE DATA RECORDS OF THE OUTPUT RELATION 710

FIG. 7

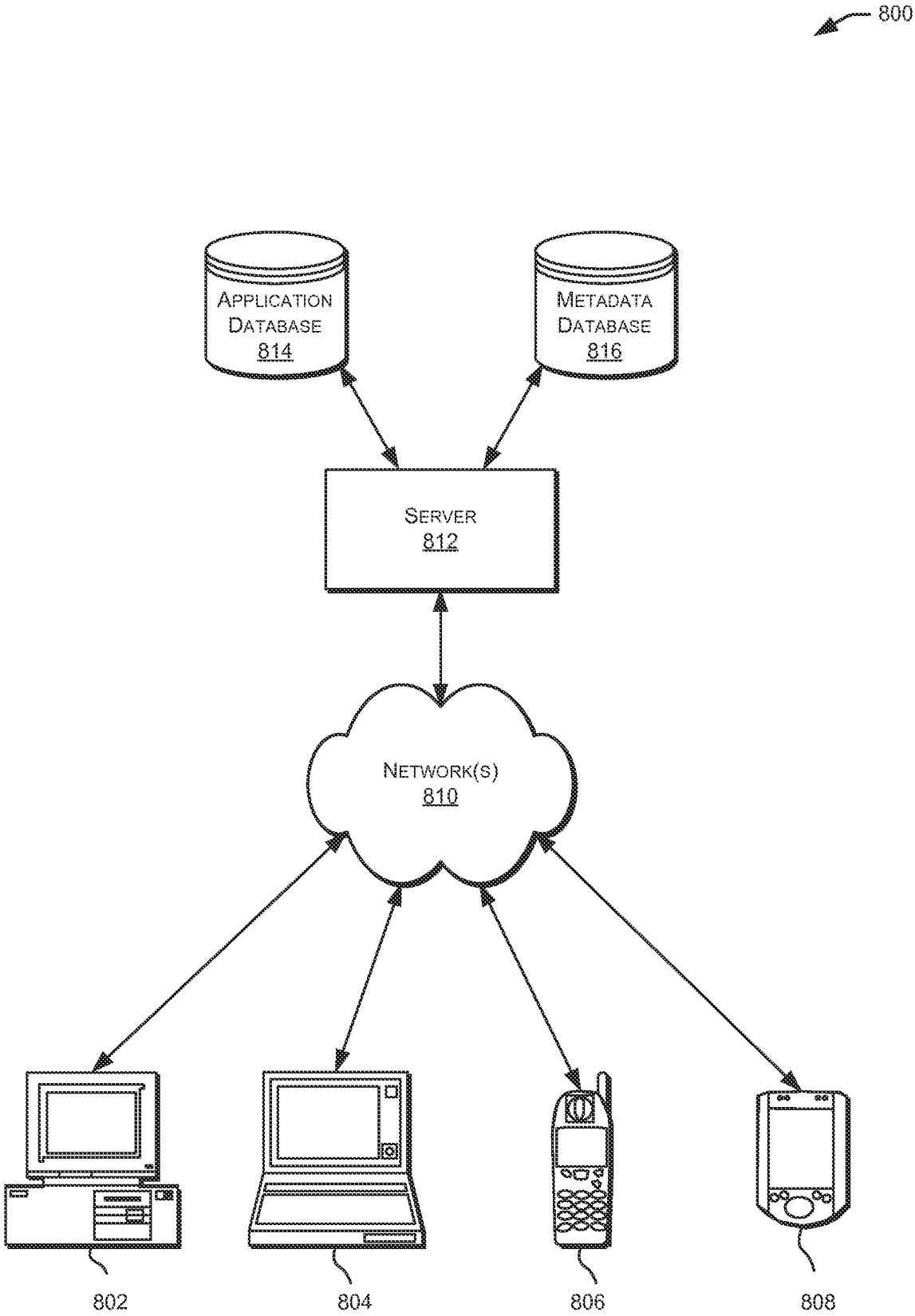

800
APPLICATION DATABASE 814
METADATA DATABASE 816
SERVER 812
NETWORK(S) 810
802
804
806
808

FIG. 8

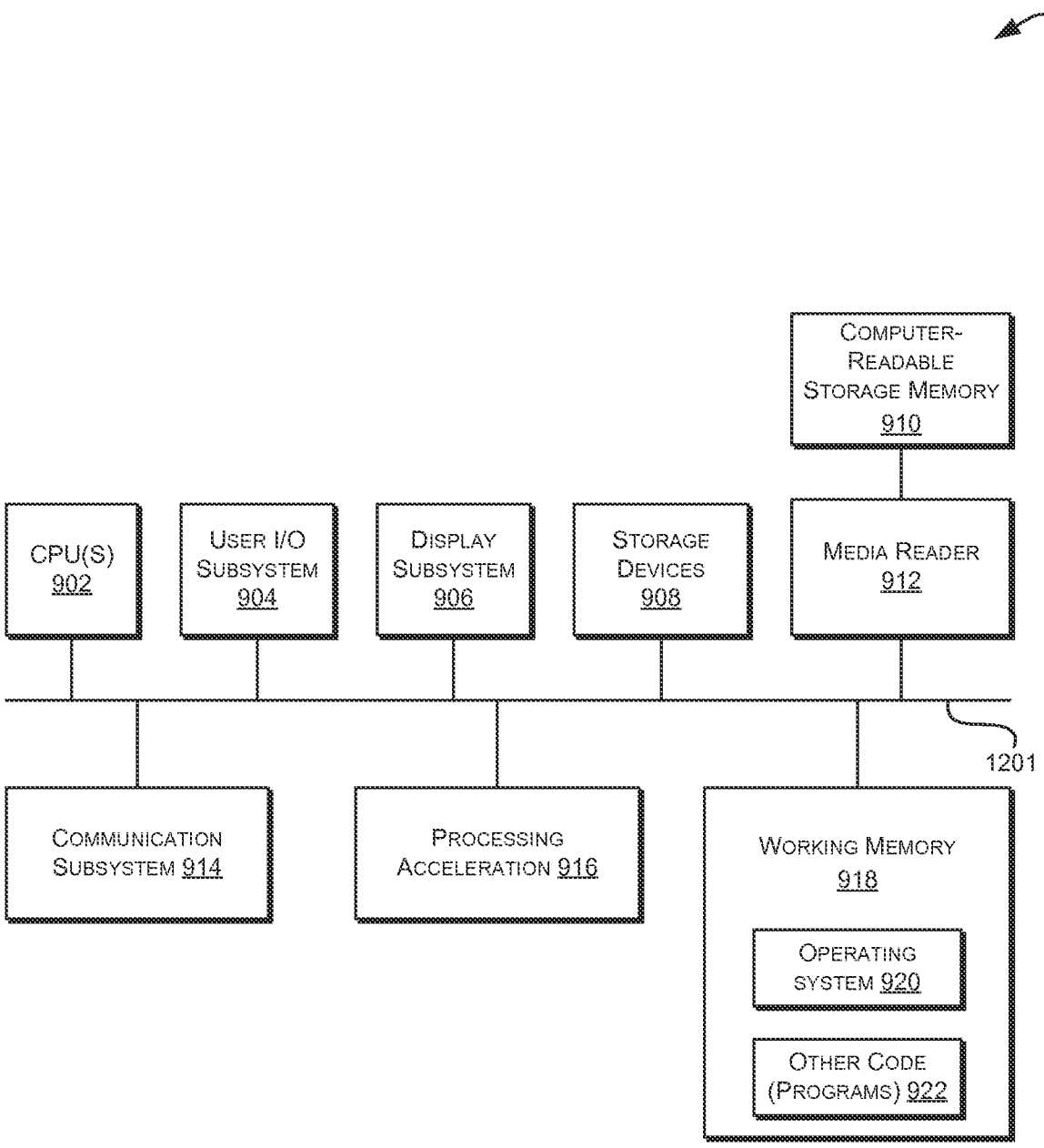

900
COMPUTER-READABLE STORAGE MEMORY 910
CPU(S) 902
USER I/O SUBSYSTEM 904
DISPLAY SUBSYSTEM 906
STORAGE DEVICES 908
MEDIA READER 912
1201
COMMUNICATION SUBSYSTEM 914
PROCESSING ACCELERATION 916
WORKING MEMORY 918
OPERATING SYSTEM 920
OTHER CODE (PROGRAMS) 922

FIG. 9

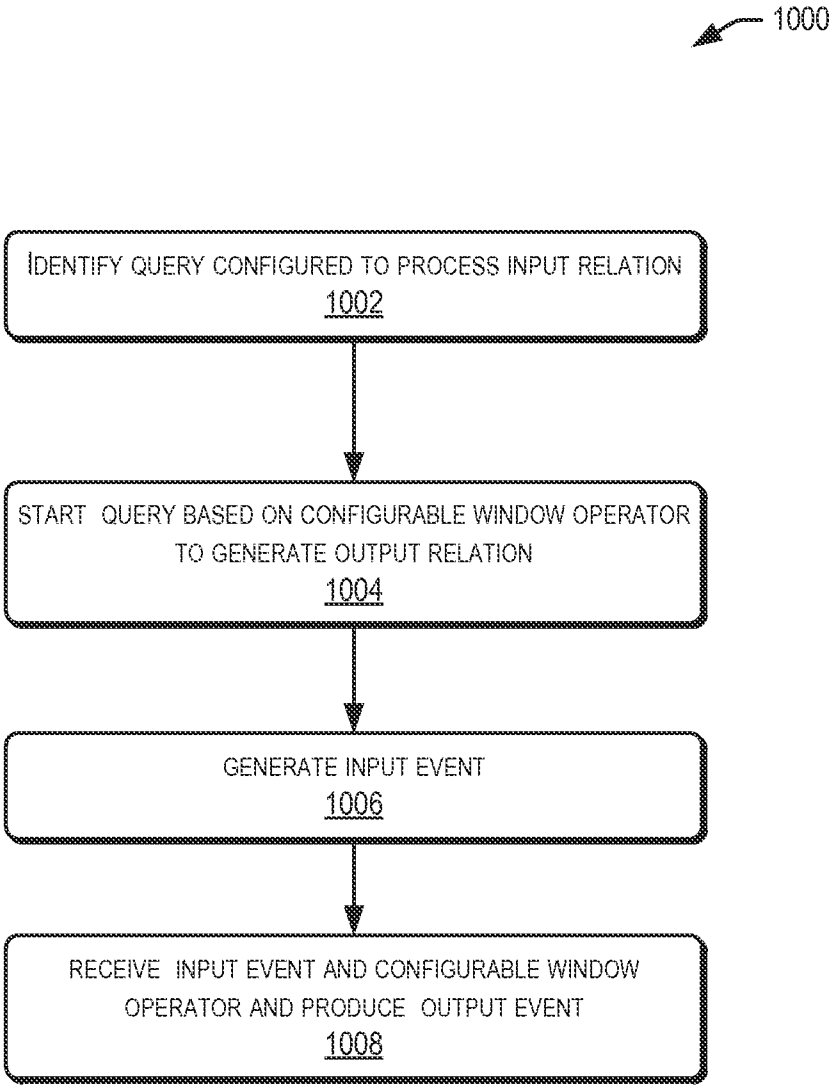

1000
IDENTIFY QUERY CONFIGURED TO PROCESS INPUT RELATION
1002
START QUERY BASED ON CONFIGURABLE WINDOW OPERATOR TO GENERATE OUTPUT RELATION
1004
GENERATE INPUT EVENT
1006
RECEIVE INPUT EVENT AND CONFIGURABLE WINDOW OPERATOR AND PRODUCE OUTPUT EVENT
1008

FIG. 10

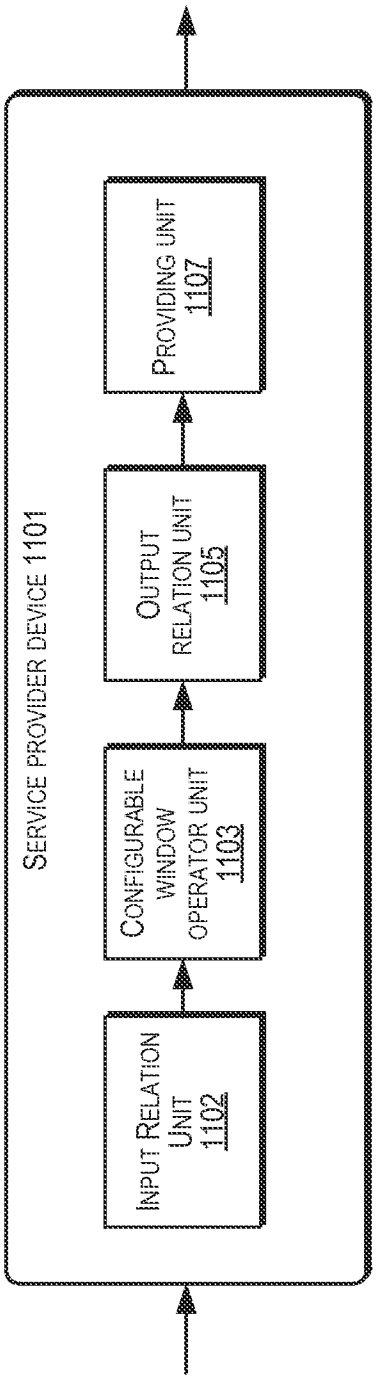

SERVICE PROVIDER DEVICE 1101
INPUT RELATION UNIT 1102
CONFIGURABLE WINDOW OPERATOR UNIT 1103
OUTPUT RELATION UNIT 1105
PROVIDING UNIT 1107

FIG. 11

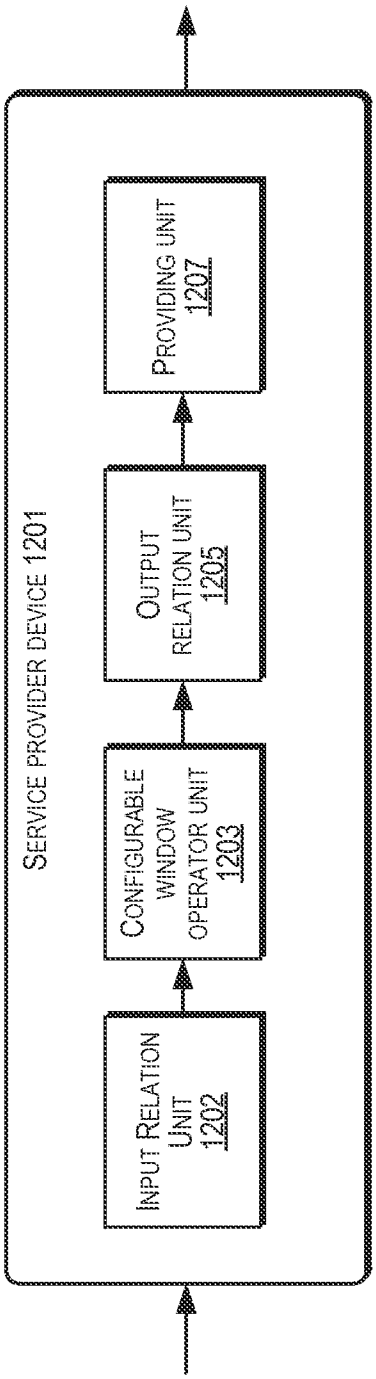

SERVICE PROVIDER DEVICE 1201
INPUT RELATION UNIT 1202
CONFIGURABLE WINDOW OPERATOR UNIT 1203
OUTPUT RELATION UNIT 1205
PROVIDING UNIT 1207

FIG. 12

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2014/039771

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06F17/30
ADD.

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
EPO-Internal

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	ARVIND ARASU ET AL: "The CQL continuous query language: semantic foundations and query execution", THE VLDB JOURNAL ; THE INTERNATIONAL JOURNAL ON VERY LARGE DATA BASES, SPRINGER, BERLIN, DE, vol. 15, no. 2, 22 July 2005 (2005-07-22), pages 121-142, XP019431176, ISSN: 0949-877X page 124 - page 130 page 141 table 1 -----	1-20

☐ Further documents are listed in the continuation of Box C.

☐ See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
16 September 2014	24/09/2014

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Yotova, Polina