



(22) Date de dépôt/Filing Date: 2005/09/13

(41) Mise à la disp. pub./Open to Public Insp.: 2007/03/13

(51) Cl.Int./Int.Cl. *G06Q 10/00* (2006.01),
G06F 17/30 (2006.01)

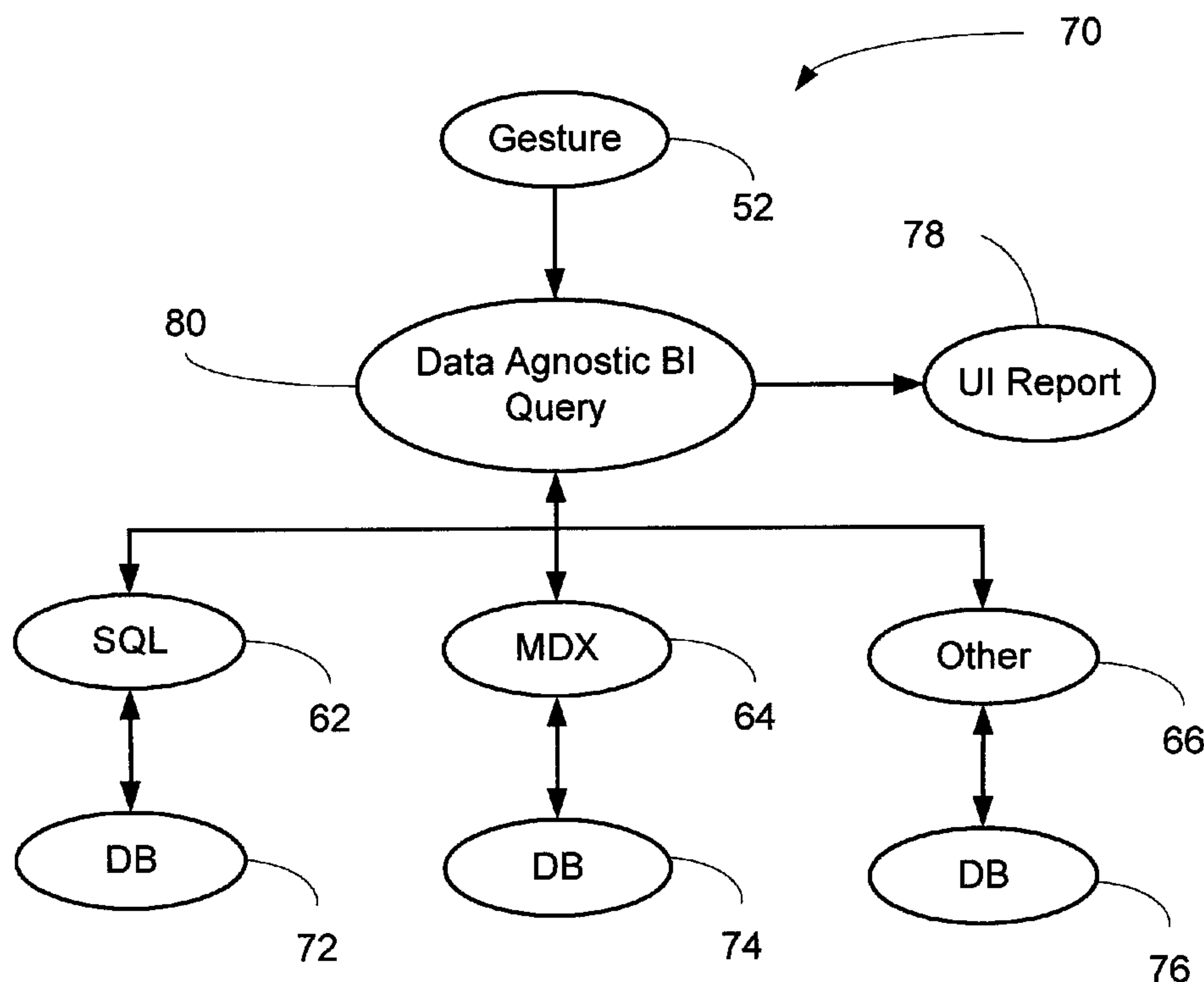
(71) Demandeur/Applicant:
COGNOS INCORPORATED, CA

(72) Inventeurs/Inventors:
AZIZI, SOUFIANE, CA;
POTTER, MIKE C., CA

(74) Agent: GOWLING LAFLEUR HENDERSON LLP

(54) Titre : SYSTEME ET METHODE DE DEMANDE DE RENSEIGNEMENTS D'AFFAIRES INDEPENDANTE DES
DONNEES

(54) Title: SYSTEM AND METHOD OF DATA AGNOSTIC BUSINESS INTELLIGENCE QUERY



(57) **Abrégé/Abstract:**

A data agnostic business intelligence query system and method are provided. The system comprises a query set component for defining data to be retrieved from a database, and a query result definition component for describing the dimensional structure of a result set for the data to be retrieved. The method comprising the steps of translating the gesture into a data agnostic business intelligence query, divide data agnostic business intelligence query into sub-queries grouped into data source query types, and sending each sub-query to a data source query engine based upon its data source query type.

Abstract

A data agnostic business intelligence query system and method are provided. The system comprises a query set component for defining data to be retrieved from a database, and a query result definition component for describing the dimensional structure of a result set for the data to be retrieved. The method comprising the steps of translating the gesture into a data agnostic business intelligence query, divide data agnostic business intelligence query into sub-queries grouped into data source query types, and sending each sub-query to a data source query engine based upon its data source query type.

System and Method of Data Agnostic Business Intelligence Query

FIELD OF THE INVENTION

[0001] The invention relates generally to data access middleware, and in particular to a
 5 system and method of data agnostic business intelligence query.

BACKGROUND OF THE INVENTION

[0002] Many organizations use data stores for storing business data, such as financial data and operational data. In order to assist business users to examine their data, various
 10 data analyzing applications are proposed. Those data analyzing applications provide various views or reports of data to users. The data analyzing applications have query engines that access the data stores to obtain desired data. Some data analyzing applications have online analytical processing (OLAP) engines to provide multidimensional views of data.

15 [0003] Data extraction, conversion, transformation, and integration are database issues. Their solutions rely on low-level query languages: relational (such as structured query language (SQL)), multidimensional (MDX), or proprietary enterprise resource planning (ERP) application programming interfaces (APIs). Business intelligence (BI) users, systems, and applications use tools that support the following tasks:

- 20 • Reporting on a wide range of data extracted from various types of database systems.
- Ad-hoc querying of data residing in relational, multi dimensional, and ERP databases.
- 25 • Analysis and exploration of data residing in relational, multi dimensional, and ERP databases.
- Integration of data from multiple data sources into a single report or analysis session.

[0004] BI systems can have data from relational databases, dimensional databases, and ERP APIs. In such scenarios, the BI system would extract information from each of the

data sources and then merge the results into a report. However, the extraction of information from each data source is different. The BI system or a query author is presented with a query language that is tied to a specific database technology. The user interface is required to be aware of the type of data source it is reporting against and the query language or query tools used vary with the data source type. The user can be presented with a user interface that uses a semantic layer to insulate him from knowledge of low level query syntax, such as SQL or MDX. However, the user experience is inconsistent across data source types.

[0005] There is a need for a better way of providing a query that is operable for a plurality of data sources.

SUMMARY OF THE INVENTION

In accordance with an embodiment of the present invention, there is provided a data agnostic business intelligence query system. The system comprises a query set component for defining data to be retrieved from a database, and a query result definition component for describing the dimensional structure of a result set for the data to be retrieved.

In accordance with another embodiment of the present invention, there is provided a data agnostic business intelligence query method. The method comprising the steps of translating the gesture into a data agnostic business intelligence query, divide data agnostic business intelligence query into sub-queries grouped into data source query types, and sending each sub-query to a data source query engine based upon its data source query type.

BRIEF DESCRIPTION OF THE DRAWINGS

[0006] These and other features of the invention will become more apparent from the following description in which reference is made to the appended drawings wherein:

[0007] Figure 1 shows a typical data access environment.

[0008] Figure 2 shows in a diagram a non-data agnostic approach to merging data source queries.

[0009] Figure 3 shows in a block diagram a data agnostic business intelligence query system, in accordance with an embodiment of the present invention.

[0010] Figure 4 shows in a tree diagram an example of a data agnostic BI query approach, in accordance with an embodiment of the data agnostic business intelligence query system.

[0011] Figure 5 shows in a flowchart an example of a method of data agnostic business intelligence query, in accordance with an embodiment of the data agnostic business intelligence query system.

[0012] Figure 6 shows in a diagram a representation of a shaped result set, in accordance with an embodiment of the data agnostic business intelligence query system.

[0013] Figure 7 shows in a diagram an example of the organization of the rowsets in the result set, in accordance with an embodiment of the data agnostic business intelligence query system.

[0014] Figure 8 shows in a diagram an example of the organization of the row edge rowsets, in accordance with an embodiment of the data agnostic business intelligence query system.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

[0015] Figure 1 shows a typical data access environment 10 for processing data. Typically, data is stored in a database 12. A database server 14 (such as a structured query language (SQL) server or a multidimensional (MDX) server) accesses the raw data stored in the database 12. A report server 16 is used to generate reports on the raw data and instruct the database server 14 to obtain information pertaining to the raw data in the database 12. An end user uses a client application 18, running on a client server, to facilitate report server 16 operations. Typically, a report server 16 has a query engine 20 for universal data access (UDA).

[0016] Data extraction, conversion, transformation, and integration are all database problems. Their solutions rely on low-level query languages: relational (such as SQL), multidimensional (MDX), or proprietary ERP APIs. BI users, systems, and applications use tools that support the following tasks:

- Reporting on a wide range of data extracted from various types of database systems.
- Ad-hoc querying of data residing in relational, multi dimensional, and ERP databases.
- 5 • Analysis and exploration of data residing in relational, multi dimensional, and ERP databases.
- Integration of data from multiple data sources into a single report or analysis session.

[0017] Figure 2 shows in a tree diagram a non-data agnostic approach 50 to merging data
 10 source queries. Gestures 52 are translated into data source queries such as SQL 62 or MDX 64. The queries are used to generate reports 68 that are then merged into a user interface (UI) report 58.

[0018] Figure 3 shows in a block diagram a data agnostic business intelligence query
 15 system 100, in accordance with an embodiment of the present invention. The data agnostic business intelligence query system 100 is suitable for fulfilling the BI user, system, or application without the need to use the low-level query languages and without the need to tailor the application for a specific data source technology. The data agnostic business intelligence query system 100 comprises a query set component 102 for defining the data to be retrieved from the database, and a query result definition component 104
 20 for describing the shape, or dimensional structure, of the result set to be returned for rendering.

[0019] Figure 4 shows in a tree diagram an example of a data agnostic BI query approach
 70, in accordance with an embodiment of the data agnostic business intelligence query system 100. Gestures 52 are translated by the data agnostic business intelligence query system 80 and sent to respective data source queries 54, 56, 66 to retrieve data from the
 25 respective data sources 72, 74, 76. The retrieved information is processed and compiled into a report sent to the UI 78.

[0020] Figure 5 shows in a flowchart an example of a method of data agnostic BI query
 (200), in accordance with an embodiment of the data agnostic business intelligence query
 30 system 100. The method (200) begins with translating gestures into a data agnostic BI query (202). Next, the data agnostic query is divided into respective data source queries

(204). Each data source query is sent to the respective data source query engine for processing (206). The processed data is compiled into a report (208). The method is done. Other steps may be added to this method.

[0021] The data agnostic BI query is a high a level query language supported for any data agnostic BI application. Complex business queries are expressed easily in this query language. A data agnostic BI query relies on the metadata model it is based on. It provides functionality for professional report authoring, casual ad-hoc querying, and sophisticated business analysis. To address the requirements of a business user, the data agnostic BI query provides powerful query capabilities with a minimum of specifications. This implies that the data agnostic business intelligence query system 100 interprets many defaults rules in a sensible way. A single data agnostic BI query can span multiple data source technologies and can be resolved by the query framework 100 and its stack of software components at the coordination, planning, and execution layers into multiple SQL, MDX, and vendor specific queries.

[0022] The data agnostic BI query has the following features:

- It is declarative.
- It is simple enough that known database techniques for query optimization, cost estimation, and query rewriting could be extended to this query.
- It provides functionality for professional report authoring, casual ad-hoc querying, and sophisticated business analysis against various data source technologies through a consistent user experience.

[0023] The data agnostic BI query specification is encapsulated within a querySet section of the Query Service API <execute> command. This command represents a request that is submitted to the query framework, i.e., a query engine, by one of its clients. When the command is a request to retrieve the result set for the enclosed querySet, data results are returned as specified by the data agnostic BI query result set API.

[0024] A querySet has one or more named queries (or query components 102) and one or more named queryResultDefinitions (QRDs) 104. A query 102 in the querySet defines the data to be retrieved from the data source while a queryResultDefinition 104 defines the result set structure to be returned. In most cases, the query simply relies on the

metadata model referenced in its source. The QRD 104 is the syntactic representation of the result set expected from the execution of a data agnostic BI query.

[0025] The QRD 104 is the main mechanism for query framework clients to tie a particular query to a particular result set. In a querySet, each QRD 104 is based on a single query that which it references. Multiple QRDs 104 in the same querySet can reference the same query 102. This allows query authors to use the same query 102 in a crosstab and a chart result sets for example. This also allows the data agnostic business intelligence query system 100 to execute a single query against a data provider and structure the results in multiple ways. A query framework API MasterDataset is returned for each queryResultDefinition specified in a querySet.

[0026] The data agnostic business intelligence query system 100 provides the ability to provide a query language that is not tailored to the data source technology that is meant to query. The data agnostic business intelligence query system 100 may be implemented as a translator in a query framework that provides the ability to build various types of BI user experiences for reporting, ad-hoc querying, and analysis that can use the query language in a consistent manner across various data source technologies. Furthermore, the query framework provides the ability to extract, convert, transform, and integrate data from multiple data sources and multiple data source types into a single report or analysis session using this high level data agnostic query language.

[0027] The query result definition (QRD) 104, which is part of the data agnostic query, is a data agnostic high level definition of a rendered result set. It allows a BI system to express the structure of the results of a data agnostic query for rendering purposes.

[0028] Advantageously, a high-level query language with rich semantics allows a business intelligence (BI) system user and/or a user interface (UI) software layer to pose BI queries to a query engine in a manner that is independent of the type of database from which the results of the query are retrieved.

[0029] Advantageously, a data agnostic query language with minimum specification allows a BI system user to perform reporting, ad-hoc querying, analysis and exploration on top of a large array of data base technologies (relational, rollup, OLAP, HOLAP, ERP) without the need to understand SQL, MDX, or other low level query languages tied to a

specific data base technology. The user experience is seamless and consistent across BI capabilities and across data source technologies.

The Query Set Component 102

5 [0030] The query set component 102 comprises a source 106 for defining the metadata upon which the query is based, and a selection 108 for identifying the metadata upon which the query is based. A query does not define the structure or presentation of the retrieved data. Optionally, the query set component can also comprise filters, dimension information, and query hints. Preferably, each query is identified by a name attribute that
10 is unique to the querySet.

[0031] The source 106 defines the metadata upon which the query is based. The source will mostly be a model reference, but the data agnostic business intelligence query system 100 supports referencing other queries as well. In addition, the data agnostic business intelligence query system 100 supports direct queries against an underlying data source
15 technology such as MDX, SQL, or a vendor interface that can be encoded within the specification. The source 106 could be the outcome of a query operation involving one or more queries followed by a unary, binary or nary commands. The result is a projection of query items that can be used by the selection and the queryResultDefinition 104.

[0032] An sqlQuery is an explicit definition of a SQL select, exec or call statement that
20 returns a row based result. The sql element contains the SQL definition as expressed in an SQL format. While not required to execute, each column in the result is preferably set to be described by a queryItem element in the projectionList so that these queryItems may be referenced in the selection and or queryResultDefinition.

[0033] An mdxQuery is an explicit definition of an MDX statement that returns a
25 multidimensional result. The mdx element contains the MDX definition as expressed in an MDX format. The projectionList describes the projected queryItems that can be used in the selection and queryResultDefinition. The dimension information describes the cube result. Queries in the query set that reference this mdxQuery and use it as a source can use the dimension information as their default dimension info. They can also
30 override, restrict, or extend it.

[0034] Query set operations combine the results of two or more queries into a single result. UNION, INTERSECT, and EXCEPT (MINUS) operations on two or more queries result in a projection list upon which other queries can be based.

5 [0035] A join operation defines a relationship between query subjects in a metadata model. Typically, these relationships are defined in the metadata model. This element is typically used to define the relationships between database tables in non-modeled data sources during a modeling application import.

[0036] The selection 108 identifies the metadata elements upon which the query is based.

10 [0037] A dataItem represents a set of data values or members. The data values or members that correspond to a dataItem are defined by an expression element. The content of an expression element is specified in accordance with the data agnostic query expression grammar. Most often, a dataItem expression refers to a query item from a metadata model. Logical constructs, arithmetic operators, other query operators, and unified functions representing both relational and set (dimensional) algebra may be
15 defined in the more complex use cases.

[0038] Aggregate functions such as total(), minimum(), maximum(), count(), average() are special query operations. While they can be specified in the dataItem expressions, these operators are typically specified using the aggregation rules discussed in the next section.

20 [0039] Each dataItem is identified by a name that is unique to the selection in which the dataItem is defined. It can be aliased with an alias that can be more meaningful than its name if the client application chooses to do so. References to other data items in the same selection are permissible, whether unqualified or qualified by the query name in which the dataItem is defined. Such references imply that the expression associated with
25 the dataItem is used in place of where it is referenced. Aggregate operations of the referenced dataItem are not transferred with the expression. For example:

```

30 <query name="sampleQuery">
    <source> ...</source>
    <selection autoSummary="true">
        <dataItem name="Amt" aggregate="sum">
            <expression>[NS].[Product].[UnitPrice] * [Qty]</expression>
        </dataItem>
        <dataItem name="Qty" aggregate="sum">

```



```

                    <expression>[NS].[OrderDetail].[Quantity]</expression>
                </dataItem>
            ...
        </selection>
5      ...
    </query>

```

[0040] The expression for the “Amt” item refers to the “Qty” item. In one embodiment of the data agnostic business intelligence query system 100, the actual “Amt” expression that would be executed resembles:

```

10  <expression>[NS].[Product].[UnitPrice] * [NS].[OrderDetail].[Quantity]</expression>

```

[0041] Note that the aggregate operator that is implicit with the “Qty” item (aggregate attribute is “sum”) is not part of the resulting expression.

[0042] References to a dataItem from another query must be qualified with the name of query in which the dataItem is defined. Following the syntax conventions currently employed, each name is enclosed in square brackets; for example, “[query].[item]”. Such references can be used anywhere that a query item reference from a metadata model is valid. The expression of the referenced dataItem is used in the in place of the query item reference. For example:

```

20  <querySet>
    <query name="SubQuery">
        <source>
            <model name="Model"/>
        </source>
        <selection autoSummary="true">
25      <dataItem name="Unit Price" aggregate="sum">
            <expression>[Model].[Product].[UnitPrice]</expression>
        </dataItem>
        <dataItem name="Qty" aggregate="sum">
30      <expression>[Model].[OrderDetail].[Quantity]</expression>
        </dataItem>
        ...
    </selection>
    </query>
35  <query name="ParentQuery">
    <source>
        <queryRef refQuery="SubQuery"/>
    </source>
    <selection autoSummary="true">
40      <dataItem name="Amt" aggregate="sum">

```

```

                                <expression>[SubQuery].[Unit Price] *
[SubQuery].[Qty]</expression>
                                </dataItem>
                                ...
5      </selection>
      </query>
      ...
</querySet>

```

[0043] The dataItem may define the aggregation rules to be applied to the expression via the aggregate and rollupAggregate attributes. The aggregation rules suggest an aggregate function to wrap the expression when the dataItem are summarized. Each attribute may specify an explicit aggregate function [automatic, summarize, none, calculated, total, minimum, maximum, average, count]. The expression itself may define the aggregate function [calculated], or the appropriate function may be derived from the underlying metadata model. In addition, aggregation may be inhibited [none], in which case the dataItem is grouped instead of summarized. Default aggregate rule is derived from the underlying metadata model. If the rollupAggregate rule is omitted, it defaults to the aggregate specification, if any; otherwise, it is also derived from the underlying metadata model.

[0044] The “automatic” and “summarize” aggregation types are reduced to the other options in accordance with defined aggregation rules.

[0045] In one embodiment of the data agnostic business intelligence query system 100, examples of aggregation types includes “none”, “calculated” and “total to “count”. “none” means that no aggregation is supposed to be applied. “calculated” means that the expression content drives the expression aggregation. “total” to “count” are the standard aggregation types.

[0046] The aggregation context expression of a dataItem having one of these aggregation types (directly or as a results of interpretation of “automatic” or “summarize” aggregation types) consists of the corresponding aggregation function applied to the dataItem’s expression. For example, the dataItem defined as:

```

<dataItem name="Qty" aggregate="total">
    <expression>[GO].[OrderDetail].[Quantity]</expression>
</dataItem>

```

will have the aggregation context expression:

total([GO].[OrderDetail].[Quantity])

[0047] The aggregate attribute of a dataItem is ignored for an OLAP source, because the OLAP source has reduced the original data by applying this type of aggregation during building of the cube.

5 [0048] In the data agnostic query, the selection 106 element by itself does not specify any result set that can be consumed by a client of the data agnostic business intelligence query system 100. A queryResultDefinition is used for that purpose. In the limited sense that a selection defines a data extract that can be operated on internally within the query framework system, this data extract may be sorted in the sense that the set of data values
10 or members represented by a dataItem may be sorted. The sort attribute on each dataItem may specify an ascending or descending sequence, or it may inhibit sorting on the values of that dataItem. This intermediary data extract that is represented by the selection will be sorted according to the specifications on each dataItem, and nested in the order of the data item in the selection list. The default is unsorted. This sorting is in essence a pre-
15 sort. It is the groupSort of the QRD that affects the final sort of data values in the result set of the query.

[0049] In the data agnostic query, the selection by itself does not specify any result set that can be consumed by a client of the data agnostic business intelligence query system 100. A queryResultDefinition is used for that purpose. In the limited sense that a
20 selection defines a data extract that can be operated on internally within the query framework, this data extract may be grouped and summarized automatically – an all-or-nothing operation that is controlled by the autoSummary attribute. When enabled, all non-additive dataItems will be grouped into a single summary level, and the additive and semi-additive dataItems are summarized. The result set will contain a single row for each
25 unique combination of the non-additive dataItem values, and an aggregate value for each additive or semi-additive dataItem. When disabled, the individual database records will be extracted as they appear in the database. The default is enabled (“true”). When the data item expression identifies a single member value or a specific member set, the autoSummary attribute has no meaning.

30 [0050] A query may contain one or more filters that eliminate data values or members from the result set and potentially affect the values of calculations. Each filter element contains at least one filterExpression. Two or more filterExpressions specified within a

filter are conjoined via AND operators. Multiple filter specifications are also conjoined via AND operators. Any filter or filterExpression may be designated as optional, in which case it is not applied when no values are provided for the parameters to which the filter or filterExpression refers.

5 [0051] Logically, one can think of the set of related queries in a querySet as blocks of operations and transformations performed on a data stream. In this logical representation, the querySet can be visualized as a tree of query operations where each node, represented by a <query>, performs operations and transformations on an input data stream defined in its source section then feeds the resulting output data stream to the next query node,
 10 which uses it as an input data stream. At the end of this process, a QRD 104 is defined to represent the structure of the last output data stream for authoring purposes. Filtering and aggregation are two special query operations performed by a query node in this logical tree. It is important to clearly specify their order. To do so, a detail filter is defined which is applied at the input data stream of a query node and hence before any
 15 calculations and aggregations are performed in that node. A summary filter is also defined, which is performed after aggregations. This summary filter is logically equivalent to the detail filter of the next query node that consumes the output data stream of the current node.

[0052] A query author can control the order in which filtering and aggregation should
 20 occur by using this mechanism in the data agnostic query querySet (i.e., query based on query also known as subquery). In one embodiment, some sensible defaults and interpretations are provided for cases where the query author would like a minimum specification in a single query. The author might not seek a granular control over desired query operations expressions.

25 [0053] The concept of a Tabular filter can be set to pre or post summary to control the order of filtering and aggregations in the context of the Tabular query. The Tabular query output is a logical data stream that is used in the Cube query, which applies, among other operations, a pivot transformation to the input stream. A filter is also specified in the cube section. One proposed solution was to add an “apply” attribute is added to the Cube
 30 filter with values “pre and post aggregation” to clearly specify the order or timing of the filter with respect to aggregations and calculations performed in the cube query. The output data stream of the Cube query is “fed” to the Summary section of the query,

another embodiment adds a filter to the Summary query. This filter is applied after all cube query operations, and hence post cube aggregations. This problem illustrates an inherent deficiency in a non-data agnostic BI query. A non-data agnostic query specification is not flexible and general enough to clearly specify an arbitrary sequence of query operations. It reliance on a “Tabular to Cube to Summary” logical operations limits its semantics to three stages of macro query operations. The data agnostic query generalizes this paradigm to any number of stages that the author wishes to specify. Advantageously, the author also has control of the sequence of operations and in particular as to the order of filtering and aggregation.

10 [0054] Without the optional level attribute, a detailFilter defines filters that are applied to the source of a query, before any aggregates are calculated. If the selection is summarized (see autoSummary), this filter inhibits source data values or members from participating in the calculation of the aggregate values; otherwise, it inhibits source data values or members from appearing in the data extract represented by the selection.

15 [0055] The detailFilter can optionally specify the level at which the filter is applied. If unspecified, the overall (or root) level of a dimension is assumed.

[0056] Without the optional level attribute, a summaryFilter defines filters that are applied after aggregates are calculated, also known as a post-aggregation filter. Logically, while the detailFilter is applied to the input data stream of a query, the summary filter is applied to its output. This distinction and the timing of the filter operation are critical only with respect to the aggregate calculation operation. For example, the final output of the query operations represented by a query is not affected by whether we sort then filter or conversely, we filter then sort. Performance requirements dictate that the latter is chosen during query planning; however, one sequence or the other does not affect the result set.

25 [0057] Typically, in most practical cases, the query author specifies a single query in the querySet to define the data to be retrieved from the database, and a single QRD 104 to define the result set structure. Headers and Footers are specified in the QRD 104 that represent aggregations at various nesting levels of the result set. In these cases, a detailFilter is applied to the data values (rows or members) in the data source, while a summaryFilter is applied to the footer or header values, which represent aggregate calculations. The summaryFilter can optionally specify the level at which the filter is

applied. If unspecified, the overall (or root) level of a dimension is assumed. Calculations at and above the specified levels are subject to the filter conditions (i.e., their values can be changed due to the filter condition).

[0058] Dimension information augments the selection. It is optional and is specified by
5 an advanced query author when

- There is no dimension information available in the source.
- The author wishes to override the dimension information in the source.
- The author wishes to extend or restrict dimension information in the source.

10 [0059] The intent of dimension information is not to define the presentation of the information, but to help query planning. In other words it can be considered a form of hint. If the dimension information is omitted then dimension information is used from the source if available. If not available, it will be defaulted by the query framework system.

[0060] A data agnostic query will undergo a series of transformations before SQL, MDX,
15 and or vendor specific APIs are produced and sent to the database. For example, a join strategy must be derived from the underlying metadata. In addition, the generated query may be optimized to better retrieve the first N rows rather than all rows, push most operations to the database, or automatically sort based upon group by structure. These algorithms may be controlled through rowLimit, executionOptimization,
20 queryProcessing, autoSort, joinOptimization and subjectOrdering hints.

The Query Result Definition Component 104

[0061] The query result definition (QRD) component 104 describes the shape, or the dimensional structure, of the result set to be returned for rendering. It is generally
25 generated from the layout specification and is used to assist the rendering operation by delivering the data to be iterated in the expected form. The QRD 104 unambiguously specifies a result set structure and represents a meta-model of the data agnostic query result set API.

[0062] In non-data agnostic query architecture, there is a disconnect between the manner
30 in which queries were posed in a request to a common query engine and how data is

returned via the query set API. The intent with the data agnostic query result set API is to align it with the data agnostic query specification such that there is a correspondence between the structure of the queryResultDefinition of the data agnostic query and the objects presented in the master/partial datasets of the result set API.

- 5 [0063] The QRD 104 can be specified either as one of the available templates or as a set of named canonical edges. The template specification is meant to provide the authoring tools and the software developer kit (SDK) with a simple specification for the most common use cases. The QRD 104 can contain optional master-detail links, generated from the layout containment relationships, which define the master and detail contexts of the relationships. The master-detail links can be specified equivalently in the QRD 104 of the master or detail query.

[0064] Simple list, grouped list, and cross tab results can be specified in a QRD 104 in a unified manner using the canonical edge specification. Simple and grouped list results have a single edge. A cross tab result has two or more edges (row, column, section 1 to section N). These edges are uniquely named. The order in which the edges are specified in the QRD 104 is also the order in which they appear in the result set. The edge information in the result set contains the unique name of the edge as specified in the QRD 104. A query framework 100 client can use the edge's unique name to relate the edges specified in the QRD 104 and the edges returned in the result set. A cross tab with an empty row or column edge can be specified with a named empty edge <edge name="row"/>. A single edge cross tab and a grouped list with no details are represented by the same canonical edge specification. The result sets for a single edge cross tab and a grouped list with no detail columns are also represented by the same result set API structure.

- 25 [0065] An edge has a list of one or more edgeGroups. These are the outer-most groups in the edge. They represent member sets (or data values) that are unioned together. Each one of these edgeGroups has one or more valueSets (that are also unioned within the edgeGroup), and a one or more edgeGroups that are nested or cross joined within the valueSets. In other words, the edge represents an arbitrary shaped result set, that is composed by stitching and intersecting sets of members. Figure 6 shows in a diagram a representation of a shaped result set 300, in accordance with an embodiment of the data

agnostic business intelligence query system 100. An example of a QRD 104 for this case is the following:

```

5 <queryResultDefinition xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="E:\bering\bering_v5_specs\main\QuerySpec\V5Quer
yResultDefinition.xsd" name="SampleQRD" refQuery="Don'tCare">
    <edges>
        <edge name="axis0">
            <edgeGroups>
                <edgeGroup>
10             <valueSets>
                    <valueSet refDataItem="G1"/>
                </valueSets>
                <edgeGroups>
                    <edgeGroup>
15             <valueSets>
                    <valueSet
refDataItem="G3"/>
                    <valueSet
20             refDataItem="G4"/>
                </valueSets>
                <edgeGroups>
                    <edgeGroup>
                    <valueSets>
                        <valueSet
25             refDataItem="G6"/>
                    </valueSets>
                    </edgeGroup>
                </edgeGroups>
                </edgeGroup>
30             </edgeGroups>
            </edgeGroup>
            <edgeGroup>
                <valueSets>
                    <valueSet refDataItem="G2"/>
35             </valueSets>
                <edgeGroups>
                    <edgeGroup>
                        <valueSets>
                            <valueSet
40             refDataItem="G5"/>
                        </valueSets>
                    </edgeGroups>
                    <edgeGroup>
                        <valueSets>
                            <valueSet
45             refDataItem="G7"/>
                        </valueSets>
                    </edgeGroup>
                </edgeGroups>
            </edgeGroup>
        </edgeGroups>
    </edge>
</edges>
</queryResultDefinition>

```



```

5
    </edgeGroup>
    </edgeGroups>
    </edgeGroup>
    </edgeGroups>
    </edgeGroup>
    </edgeGroups>
    </edge>
    </edges>
</queryResultDefinition>

```

10 [0066] The data agnostic query result set API presents each group as a rowset that can be iterated using an IQSSetIterator object. Group headers and footers are presented as separate rowsets that can be accessed within context of a group's corresponding rowset. The name of the rowset corresponding to a group is unique and is that of the dataItemRef of the valueSet that represents the level key of the group in the QRD 104, making it clear

15 to the client application how the data in the result set corresponds to the layout specification.

[0067] Just as the data agnostic query specification maintains a consistent approach to specifying groups in both list and cross tab queries, the data agnostic query result set API presents a single approach to iterating values in a list report as when iterating values

20 along the edges of a cross tab report. In the data agnostic query result set API, list reports are accessed via the same IQSSetIterator class that is used to navigate the edges of cross tab result sets. At any grouping level (represented by a separate rowset), header and footer values may be obtained at any time. All of the detail rows of a report are contained in the single, innermost rowset named "details".

25 [0068] In the data agnostic query result set API, there are no restrictions on how rowsets are related. The result set is instead restricted by what can be authored in the data agnostic query specification.

[0069] An edgeGroup represents an arbitrary shaped set of members (data values) on an edge. A flat list of non-nested edge groups in an edge specification can be used to

30 represent the unioning of member sets. Each group can have one or more valueSets that represent the group's members (based on a caption key and associated body attributes), an optional header and/or footer, a sort, and suppression. Each group can also have one or more nested groups.

[0070] An explorer-mode cross tab edge can be specified by a set of nested edge groups. By nesting and unioning edge groups, a query framework client can specify a reporter-mode crosstab edge.

5 [0071] A grouped list report can be specified by a set of nested edge groups with the inner most edge group representing the details. This special group is not keyed on any level (i.e., it valueSet has not refDataItem attribute) and its body references the detail columns as level attributes.

10 [0072] A valueSet, also known as a memberSet, defines a collection of values or members to be returned for an edgeGroup. It represents a (nesting) level in an explorer style edge. The refDataItem attribute of this element represents the "key" associated with the level. The name attribute identifies the valueSet within the QRD 104, and is unique within the scope of the QRD 104.

15 [0073] The groupHeader and groupFooter child elements of the valueSet element define a set of data values or members that represents a summary of the group members. In one embodiment, group header footer rules include:

- Any valueset (group, groupHeader, groupFooter, groupBody) can be empty. If so then they result in a row with no columns and a cell coordinate of -1.
- If Header and/or Footers have no dataItem then the value of the query item representing the caption key of the parent group for the out-most group, "Summary" can be used (translated).
- Headers and Footers can only contain a dataitem of the following
 1. A dataitem referring to a set of measures. This means an aggregate is created for each measure specified in the header and footer. This is the only way to specify aggregation.

And/Or

2. Dataitem(s) that is/are found in the direct parent Valueset of the current valueset. i.e for the below QRD you can see that the groupFooter contains the dataItem of the parent valueSet
 30 .."Country"

[0074] For example, in one embodiment:

RIGHT Country is a direct parent of City

```

5  <edgeGroup>
    <valueSets>
        <valueSet refDataItem="Country">
            </valueSet>
        </valueSets>
    <edgeGroups>
        <edgeGroup>
            <valueSets>
                <valueSet refDataItem="City">
                    <groupFooter>
                        <dataItemRef
10  refDataItem="Country"/>
                    </groupFooter>
                </valueSet>
            </valueSets>
        </edgeGroup>
    </edgeGroups>
20  </edgeGroup>

```

WRONG Province not a direct parent of City.. Country is

```

25  <edgeGroup>
    <valueSets>
        <valueSet refDataItem="Country">
            </valueSet>
        </valueSets>
    <edgeGroups>
        <edgeGroup>
            <valueSets>
                <valueSet refDataItem="City">
                    <groupFooter>
                        <dataItemRef refDataItem="Province"/>
30  </groupFooter>
                </valueSet>
            </valueSets>
        </edgeGroup>
    </edgeGroups>
35  </edgeGroup>
40  </edgeGroup>

```

WRONG City is not a direct parent of City .. Country is

```

5  <edgeGroup>
    <valueSets>
      <valueSet refDataItem="Country">
        </valueSet>
      </valueSets>
    <edgeGroups>
      <edgeGroup>
        <valueSets>
          <valueSet refDataItem="City">
            <groupFooter>
              <dataItemRef refDataItem="City"/>
            </groupFooter>
          </valueSet>
        </valueSets>
      </edgeGroup>
    </edgeGroups>
  </edgeGroup>

```

20 WRONG Footer dataItem not a direct parent of City.. Province is a direct parent not Country

```

25 <edgeGroup>
    <valueSets>
      <valueSet refDataItem="Country">
        </valueSet>
      </valueSets>
    <edgeGroups>
      <edgeGroup>
        <valueSets>
          <valueSet refDataItem="Province">
            </valueSet>
          </valueSets>
        <edgeGroups>
          <edgeGroup>
            <valueSets>
              <valueSet refDataItem="City">
                <groupFooter>
                  <dataItemRef
35 refDataItem="Country"/>
                </groupFooter>
              </valueSet>
            </valueSets>
          </edgeGroup>
        </edgeGroups>
      </edgeGroup>
    </edgeGroups>
  </edgeGroup>
45 </edgeGroups>

```


</edgeGroup>

[0075] The groupBody child element of the valueSet element defines the attributes to be returned for each member in the group.

5 [0076] The groupSort child element of the valueSet element defines the sort order for the group members within a context defined by the entire result set. A query author can define a sort using projected and non projected items. The groupSort can reference a data item from the associated query even if the data item was not used in QRD 104. For a detail group (i.e., a group with a valueSet that has no data item reference and has a group body reference a list of items) the order of the groupSort items dictates the order in which
10 the details are sorted.

[0077] The suppression child element of the valueSet element defines the suppression mode applied to the group members. This mode is specified by one of the child elements and results in removing some members from the group within a context defined by the entire result set.

15 [0078] If a calculation does not explicitly reference a measure, and one is required to resolve the calculation (e.g., top 10 customers, but no measure is referenced), a measure must be chosen to resolve the expression. In one embodiment, rules include:

20 The rules are that if a calculation along an edge requires a measure and one is not specified, data access and modeling (DA&M) obtains a measure reference in the following order. If there is only a single measure referenced explicitly in the entire QRD 54, DA&M uses this measure. Otherwise, if there is one and only one edge containing a reference to a single measure (and nothing else), it uses that measure. Otherwise, DA&M will use the default measure as defined by the underlying data
25 source.

[0079] In one embodiment, queryResultDefinition templates represent a choice of one of three basic templates that cover the most common report types (lists, cross tabs, charts). They are meant to provide authoring tools and the SDK with simple specifications for the most common use cases.

30

Use Cases

Simple List

[0080] One basic data agnostic query that may be specified is the Simple List. The result set may contain summary or detail database rows (autoSummary). In both cases, the result set structure is the same as defined by the QRD 104. One grouping level may be specified. Any aggregate specifications are applicable only to the lowest grouping level in a summary query – since there's only one grouping level, control break aggregates at various grouping levels are not supported. The next example is of a Simple List report containing [Order year], [Order method], and [Quantity].

10

<i>Order year</i>	<i>Order method</i>	<i>Quantity</i>
2000	E-mail	86,884
2000	Fax	34,462
2000	Mail	54,874
2000	Sales visit	135,262
2001	E-mail	122,350
2001	Fax	41,558
2001	Mail	43,672
2001	Sales visit	191,578
2002	E-mail	139,086
2002	Fax	39,824
2002	Mail	25,684
2002	Sales visit	208,858

[0081] The QRD for this example (using canonical edge specification) is the following

```

15      <queryResultDefinitions>
          <queryResultDefinition name="rs1" refQuery="query1">
              <edges>
                  <edge name="edge0">
                      <edgeGroups>
                          <edgeGroup>
                              <valueSets>
20                                  <valueSet>
                                      <groupBy>
                                          <dataItemRef
refDataItem="Order Year"/>
                                          <dataItemRef
25 refDataItem="Order Method"/>
                                          <dataItemRef
refDataItem="Quantity"/>

```



```

5
10
15
20
25
30

```

</groupBody>
 </valueSet>
 </valueSets>
 </edgeGroup>
 </edgeGroups>
 </edge>
 </edges>
 </queryResultDefinition>
 </queryResultDefinitions>
 </querySet>
 [0082] The QRD for this example (using the list template specification) is the following
 <queryResultDefinitions>
 <queryResultDefinition name="rs1" refQuery="query1">
 <resultTemplate>
 <listResult>
 <details>
 <dataItemRef refDataItem="Order Year"/>
 <dataItemRef refDataItem="Order
 Method"/>
 <dataItemRef refDataItem="Quantity"/>
 </details>
 </listResult>
 </resultTemplate>
 </queryResultDefinition>
 </queryResultDefinitions>
 </querySet>

Grouped List

[0083] The next example is of a list report containing [Order year], [Order method], and
 [Quantity], grouped by [Order year] and with a report level summary.

<i>Order year</i>	<i>Order method</i>	<i>Quantity</i>
2000	E-mail	86,884
	Fax	34,462
	Mail	54,874
	Sales visit	135,262
2000		311,482
2001	E-mail	122,350
	Fax	41,558
	Mail	43,672
	Sales visit	191,578
2001		399,158

2002	E-mail	139,086
	Fax	39,824
	Mail	25.684
	Sales visit	208,858
2002		413,452
Summary		1,123,872

[0084] The QRD for this example (using canonical edge specification) is the following

```
<queryResultDefinition name="groupedList" refQuery="some query">
  <edges>
5    <edge name="edge0">
      <edgeGroups>
        <edgeGroup>
          <valueSets>
            <valueSet refDataItem="Order Year">
10              <groupFooter>
                <dataItemRef
refDataItem="Quatity"/>
              </groupFooter>
            </valueSet>
          </valueSets>
15        </edgeGroup>
      </edgeGroups>
    </edge>
  </edges>
  <edgeGroups>
    <edgeGroup>
      <valueSets>
        <valueSet>
20          <groupBody>
            <dataItemRef
refDataItem="Order Method"/>
            <dataItemRef
refDataItem="Quantity"/>
25          </groupBody>
        </valueSet>
      </valueSets>
    </edgeGroup>
  </edgeGroups>
30 </edge>
</edges>
</queryResultDefinition>
```

[0085] Figure 7 shows in a diagram an example of the organization of the rowsets in the
35 result set 320, in accordance with an embodiment of the data agnostic business
intelligence query system 100. Note that the [Order method] rowset contains both the

[Order method] and [Quantity] data sets from the QRD 104. Also, the [Order year]_Footer rowset contains one row of data that represents the report summary.

Crosstab

- 5 [0086] A cross tab result presents a grid of summarized data values: effectively, it is an intersection of two Grouped List results. A QRD 104 with two or more edges defines a cross tab result. Aggregates at various intersections are calculated automatically. This example is the same as the previous example, except the data is presented as a cross tab.

		Quantity
2000	E-mail	86,884
	Fax	34,462
	Mail	54,874
	Sales visit	135,262
	2000	311,482
2001	E-mail	122,350
	Fax	41,558
	Mail	43,672
	Sales visit	191,578
	2001	399,158
2002	E-mail	139,086
	Fax	39,824
	Mail	25,684
	Sales visit	208,858
	2002	413,452
Order Year		1,123,872

10

[0087] The QRD 104 for this example (using canonical edge specification) is the following

```

15  <queryResultDefinition name="groupedList" refQuery="some query">
      <edges>
        <edge name="edge0">
          <edgeGroups>
            <edgeGroup>
              <valueSets>
                <valueSet refDataItem="Order Year">
20                <valueSet refDataItem="Order Year">
                  <groupFooter>

```

```

5
10
15
20
25
30
    <dataItemRef
refDataItem="Quatity"/>
    </groupFooter>
    </valueSet>
    </valueSets>
    <edgeGroups>
    <edgeGroup>
    <valueSets>
    <valueSet
refDataItem="Order Method">
    <groupFooter>
    <dataItemRef
refDataItem="Order Year"/>
    </groupFooter>
    </valueSet>
    </valueSets>
    </edgeGroup>
    </edgeGroups>
    </edgeGroup>
    </edgeGroups>
    </edge>
    <edge name="edge1">
    <edgeGroups>
    <edgeGroup>
    <valueSets>
    <valueSet refDataItem="Quantity"></valueSet>
    </valueSets>
    </edgeGroup>
    </edgeGroups>
    </edge>
    </edges>
</queryResultDefinition>

```

[0088] The representation of the rowsets is identical to the previous example, except that the [Order method] rowset no longer contains the [Quantity] column – those values are now contained within a cell rowset iterator.

Crosstab 2

[0089] The first example presents simple cross tab report with [Country] nested within [Product] along the row edge and [Quantity] along the column edge. Sub-totals are calculated for each product.

		Quantity
Product1	USA	1000

	Canada	500
	France	2000
	Total	3500
Product2	USA	3000
	Canada	2000
	France	500
	Total	5500

[0090] The QRD 104 for this example (using canonical edge specification) is the following:

```
5  <queryResultDefinition name="groupedList" refQuery="some query">
    <edges>
        <edge name="rows">
            <edgeGroups>
                <edgeGroup>
                    <valueSets>
10                 <valueSet refDataItem="Product"/>
                    </valueSets>
                    <edgeGroups>
                        <edgeGroup>
                            <valueSets>
15                             <valueSet
                                refDataItem="Country">
                                    <groupFooter>
                                        </groupFooter>
20                                     </valueSet>
                                    </valueSets>
                                </edgeGroup>
                            </edgeGroups>
                        </edgeGroup>
                    </edgeGroups>
                </edge>
                <edge name="columns">
                    <edgeGroups>
                        <edgeGroup>
                            <valueSets>
30                             <valueSet refDataItem="Quantity"></valueSet>
                            </valueSets>
                        </edgeGroup>
                    </edgeGroups>
                </edge>
            </edges>
35 </queryResultDefinition>
```

[0091] Notice the empty group footer for the country valueSet. It indicates that the rowset corresponding to this footer should have zero columns, which is a valid case. Consumers of this result set will use the existence of this empty rowset to form grouping breaks for example when rendering such a result set.

- 5 [0092] Figure 8 shows in a diagram an example of the organization of the row edge rowsets 340, in accordance with an embodiment of the data agnostic business intelligence query system 100.

[0093] The systems and methods according to the present invention may be implemented by any hardware, software or a combination of hardware and software having the
10 functions described above. The software code, either in its entirety or a part thereof, may be stored in a computer readable memory. Further, a computer data signal representing the software code that may be embedded in a carrier wave may be transmitted via a communication network. Such a computer readable memory and a computer data signal are also within the scope of the present invention, as well as the hardware, software and
15 the combination thereof.

[0094] While particular embodiments of the present invention have been shown and described, changes and modifications may be made to such embodiments without departing from the true scope of the invention.

WHAT IS CLAIMED IS:

1. A data agnostic business intelligence query system for data agnostic querying, the system comprising:
 - 5 a query set component for defining data to be retrieved from a database; and
 - a query result definition component for describing the dimensional structure of a result set for the data to be retrieved.
2. The data agnostic business intelligence query system as claimed in claim 1, wherein
 - 10 the query set component includes:
 - a source element for defining metadata upon which the data agnostic query is based; and
 - a selection element for identifying metadata elements upon which the data agnostic query is based.
 - 15
3. The data agnostic business intelligence query system as claimed in claim 3, wherein the query set component further includes one or more of:
 - a name attribute for uniquely identifying the query set component;
 - a filter element for eliminating data values or members from a result set;
 - 20 a dimension information element for augmenting the selection element; and
 - a query hint element for transforming the data agnostic query.
4. A method of data agnostic business intelligence query, the method comprising the steps of:
 - 25 translating the gesture into a data agnostic business intelligence query;
 - divide data agnostic business intelligence query into sub-queries grouped into data source query types; and
 - sending each sub-query to a data source query engine based upon its data source query type.
 - 30
5. The method as claimed in claim 4, wherein the step of dividing includes the steps of:
 - locating a data item type in a query specification; and

allocating a sub-query associated with the data item type to a data source query type group.

6. The method as claimed in claim 4, further comprising the steps of:
 - 5 receiving a data set result from the data source query engine; and
 - compiling the data set result into a report.

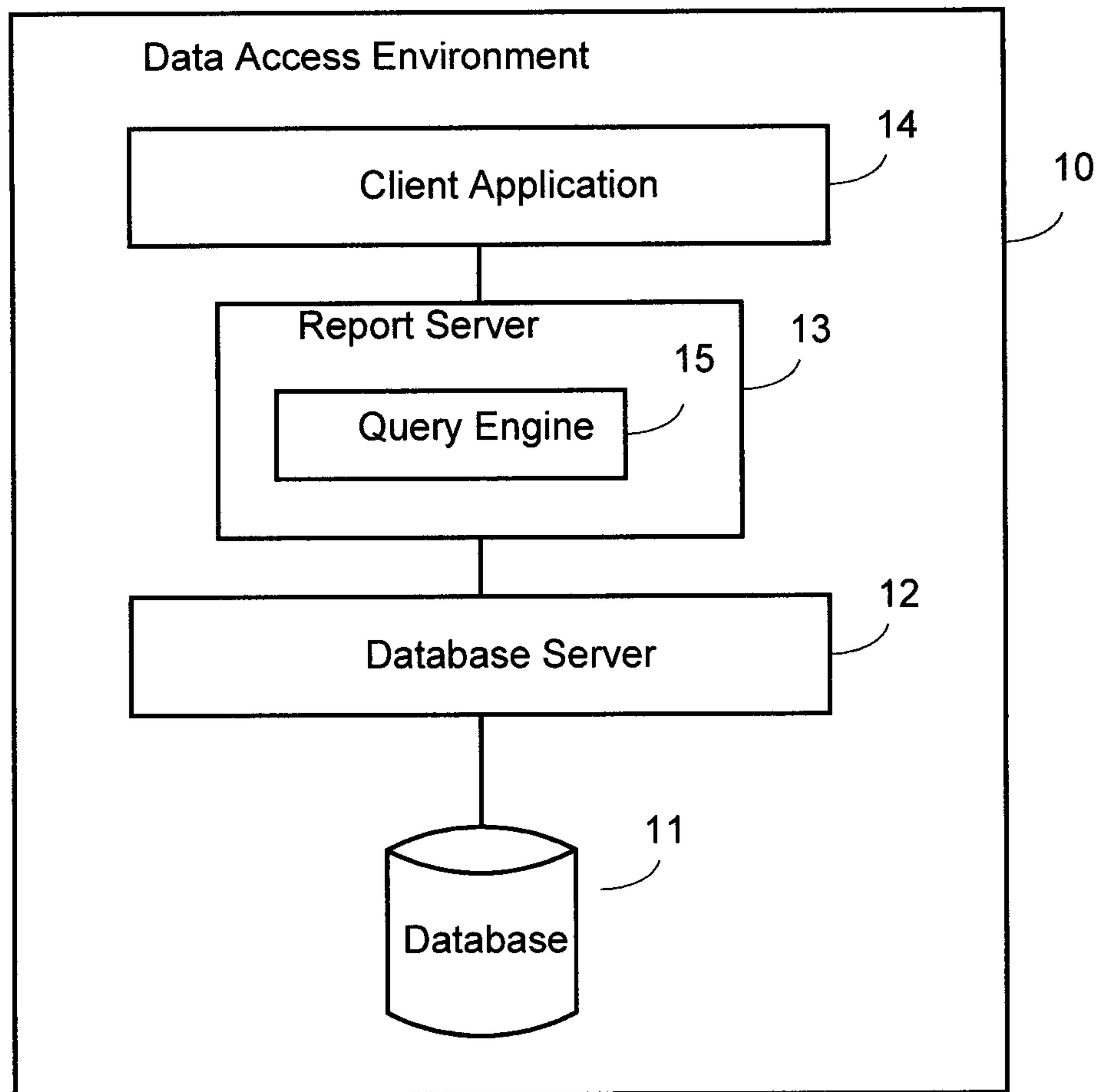


Figure 1

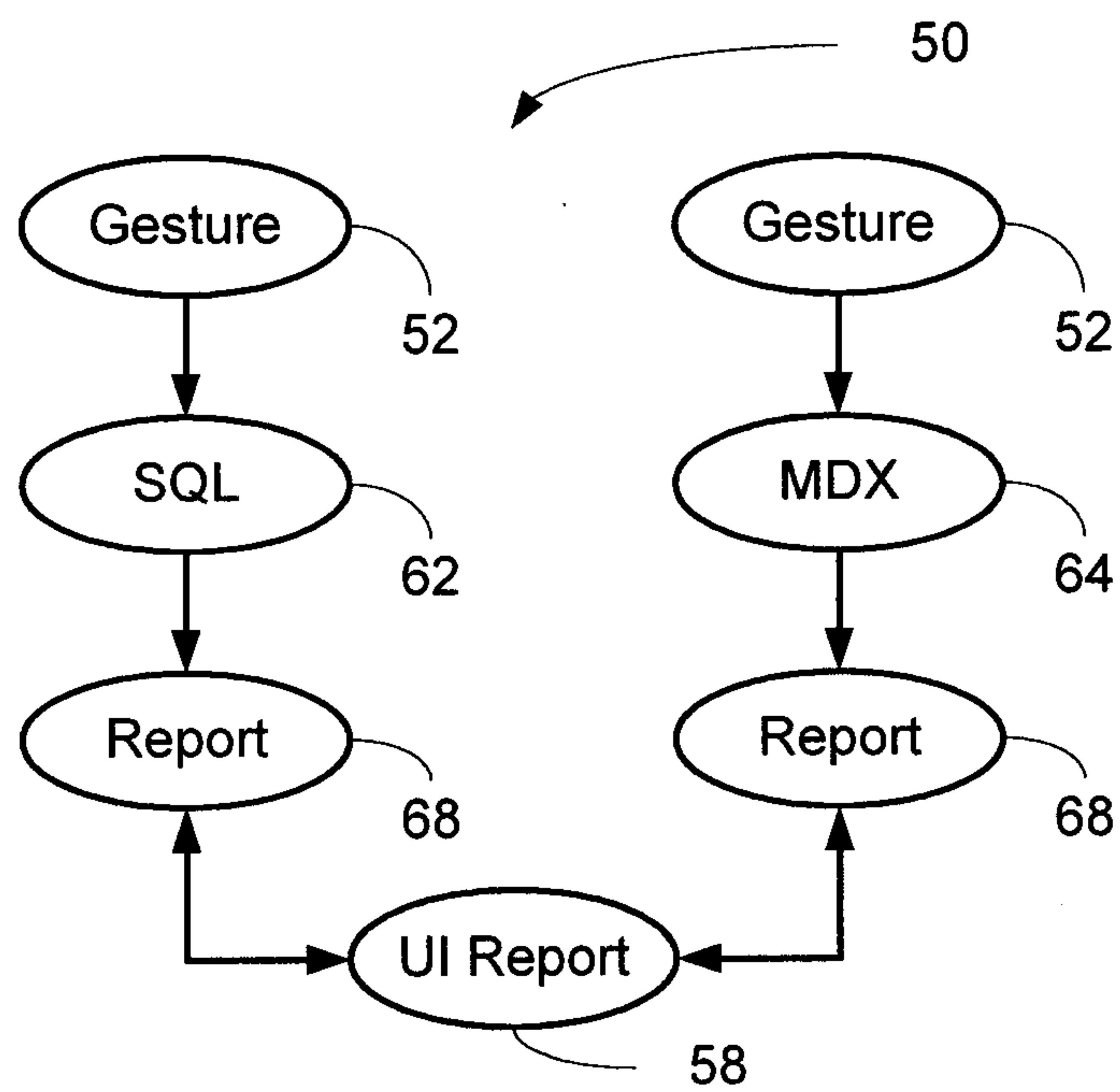


Figure 2

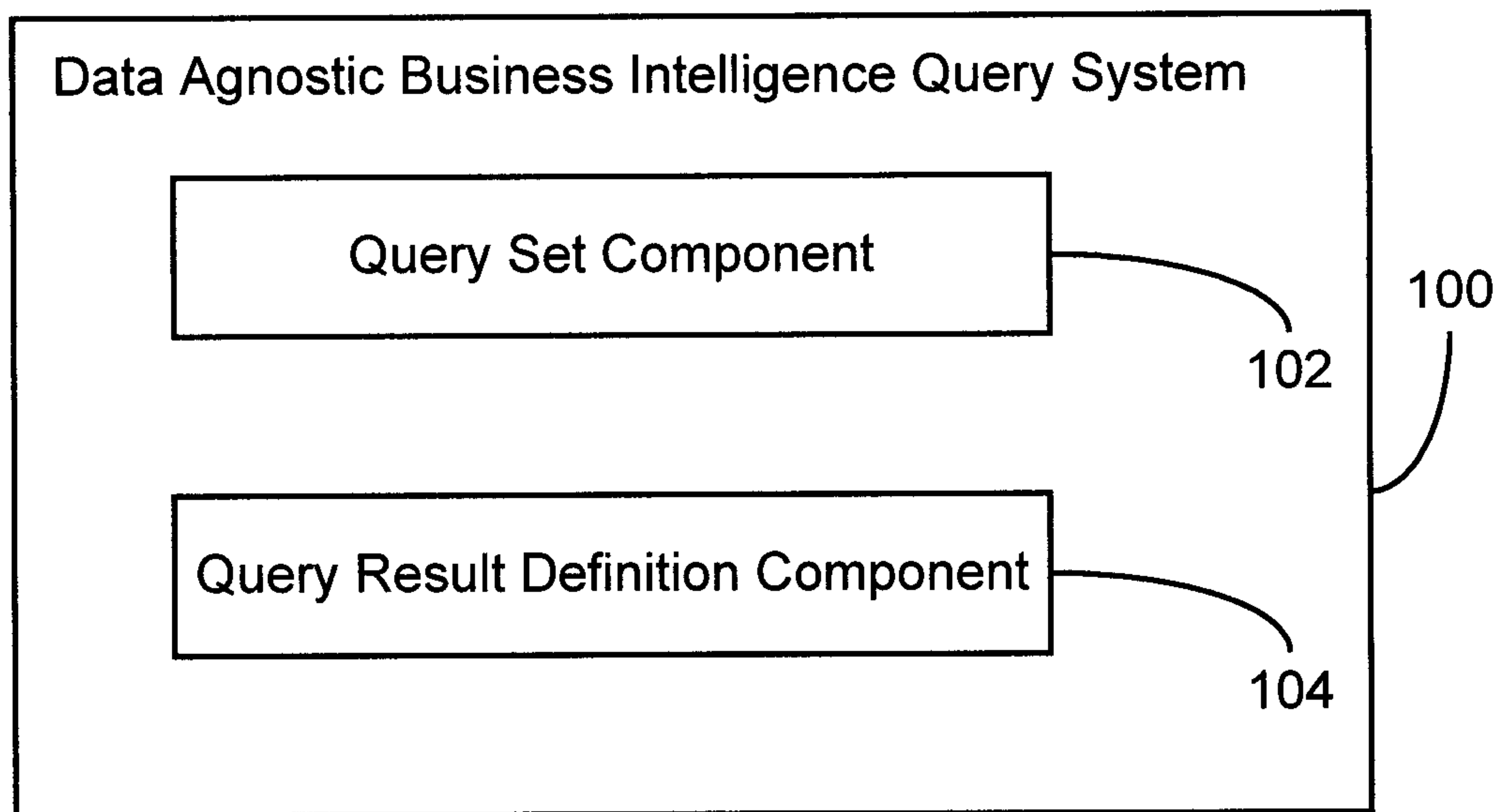


Figure 3

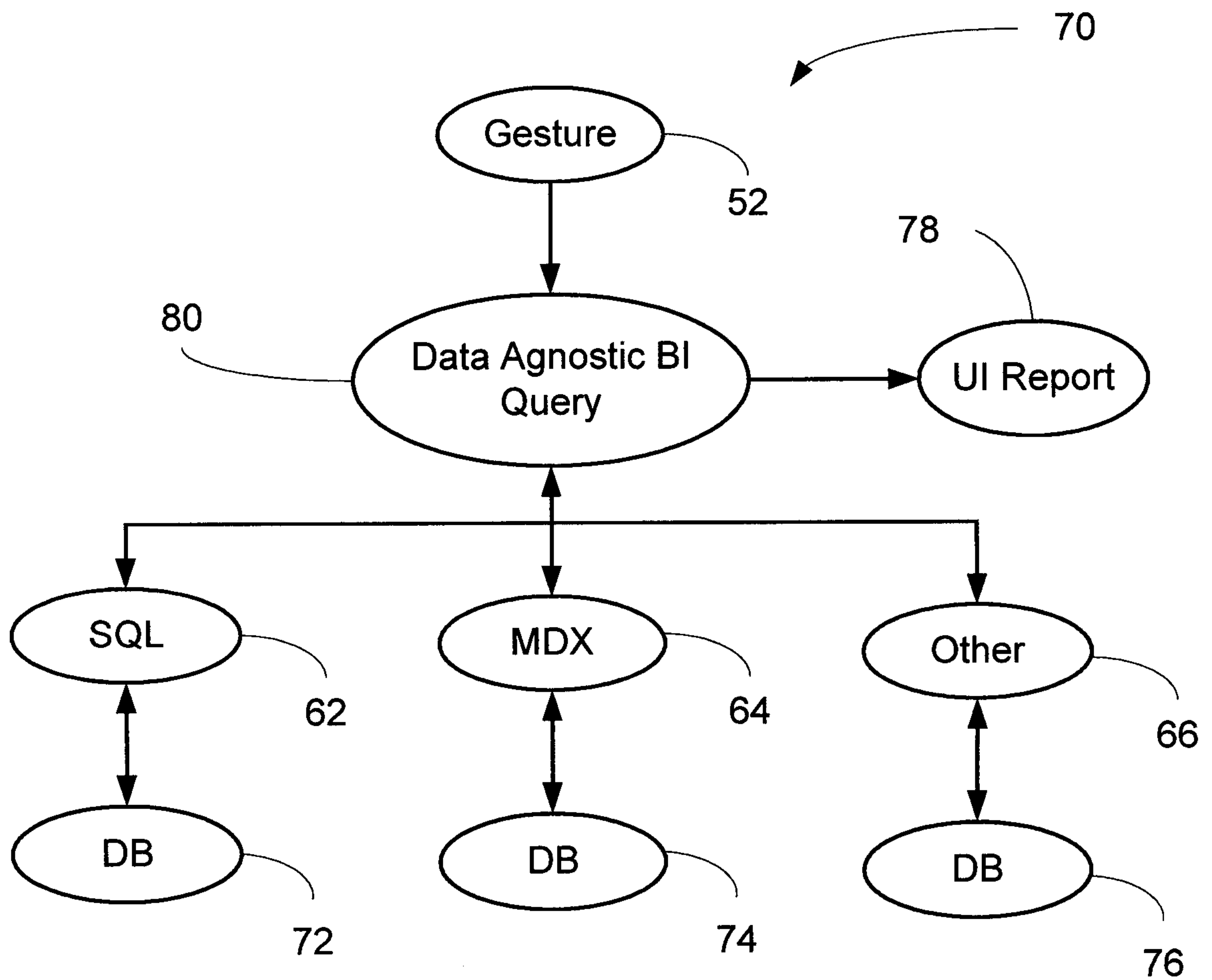


Figure 4

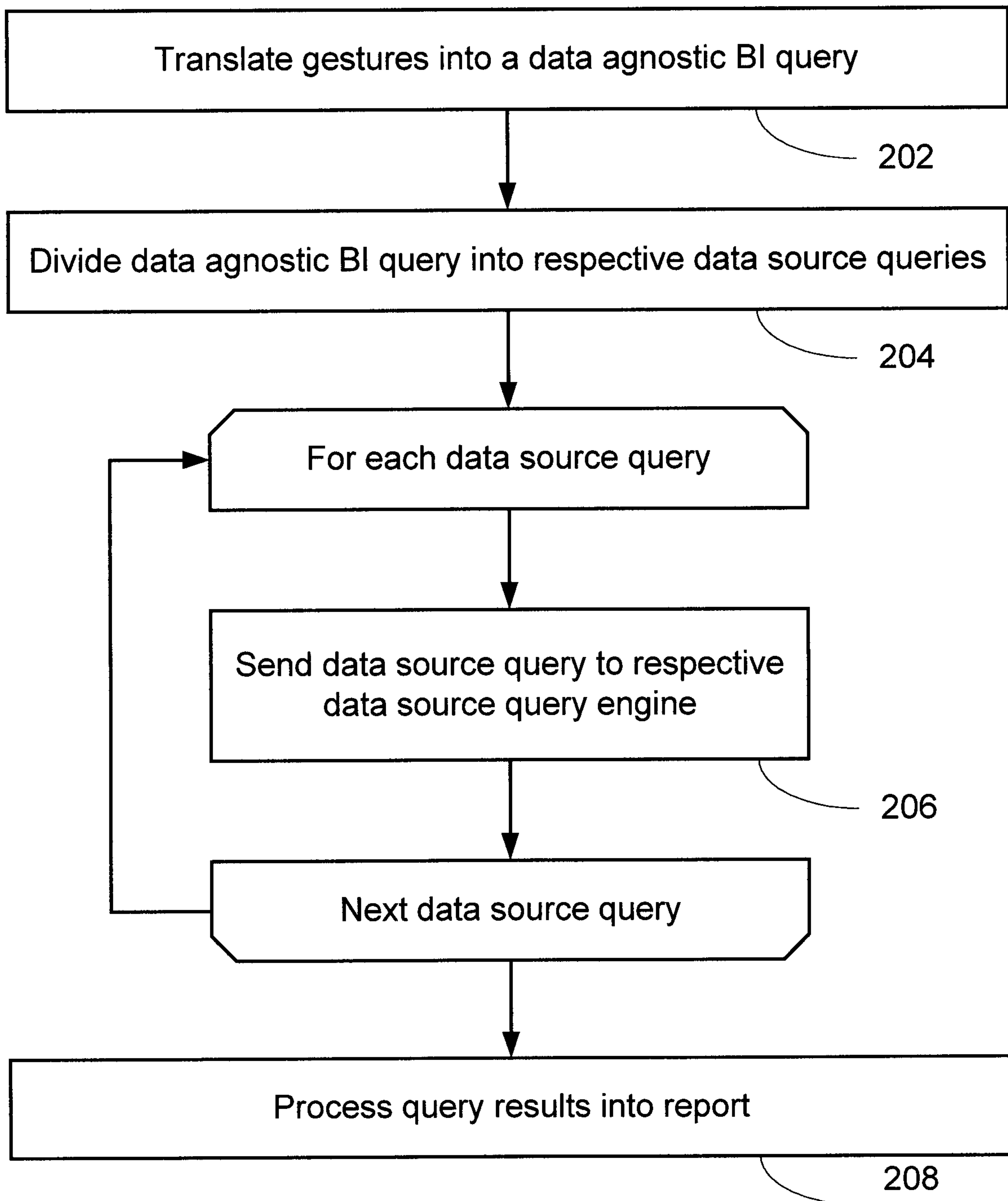


Figure 5

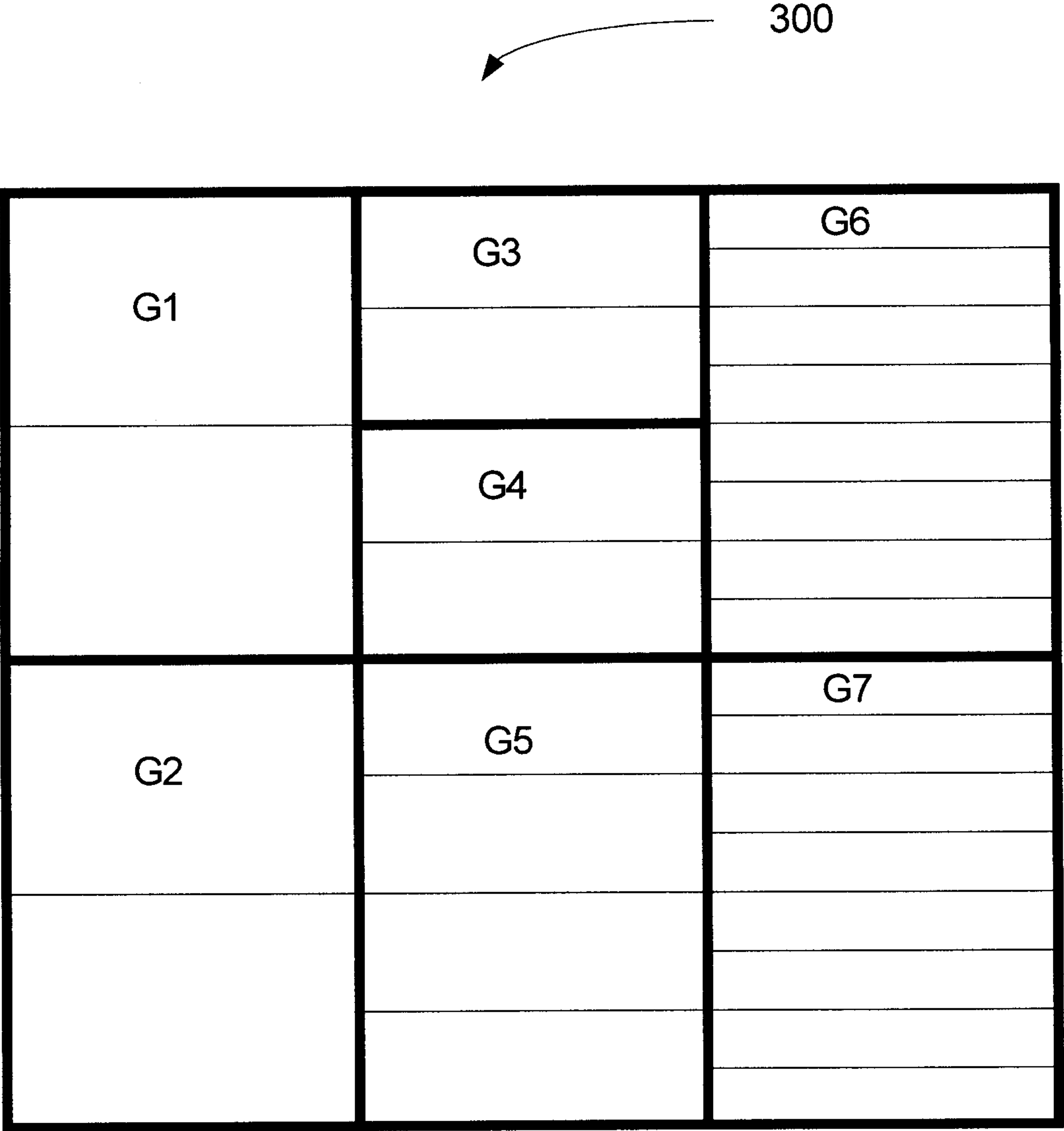


Figure 6

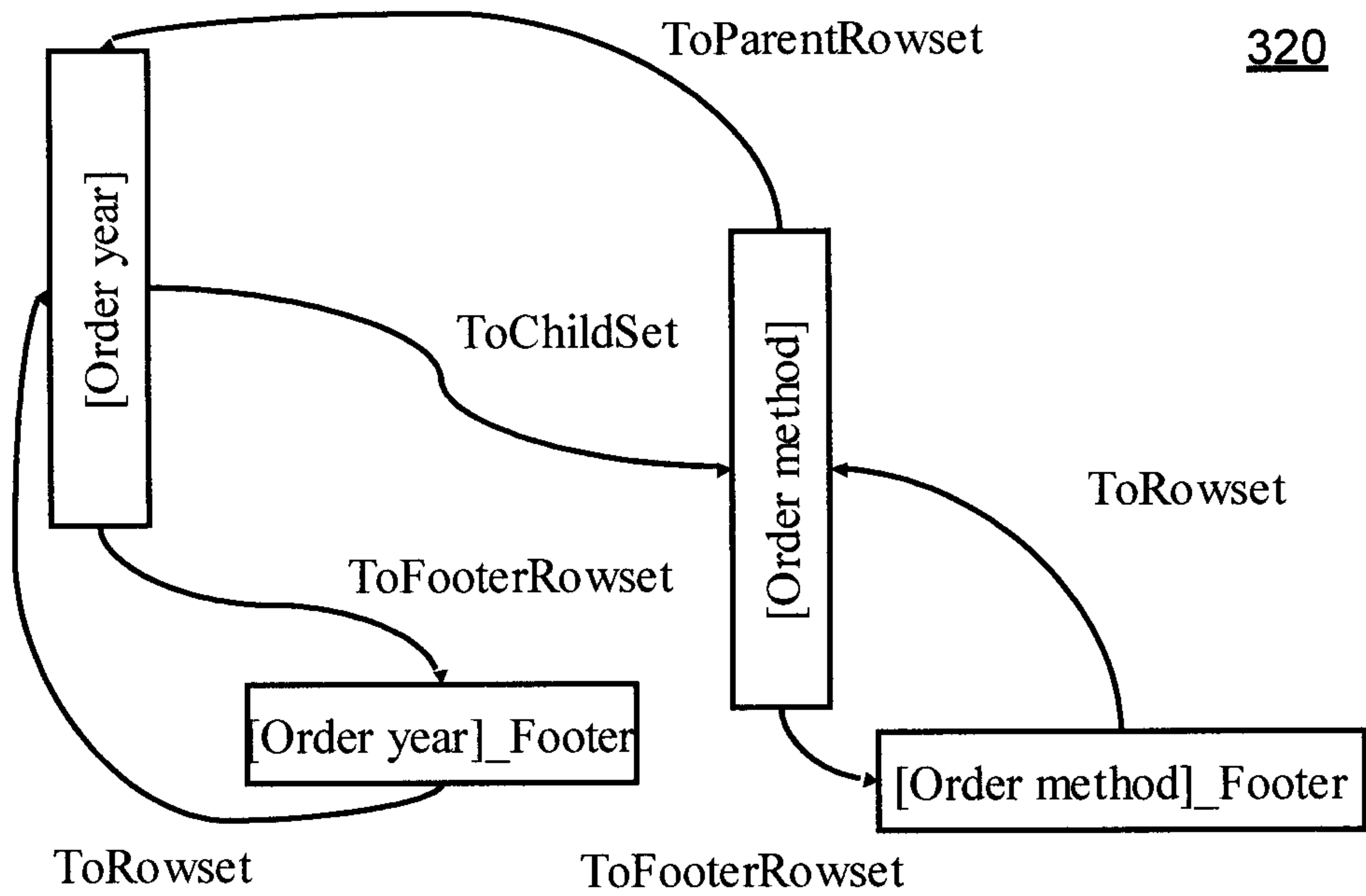


Figure 7

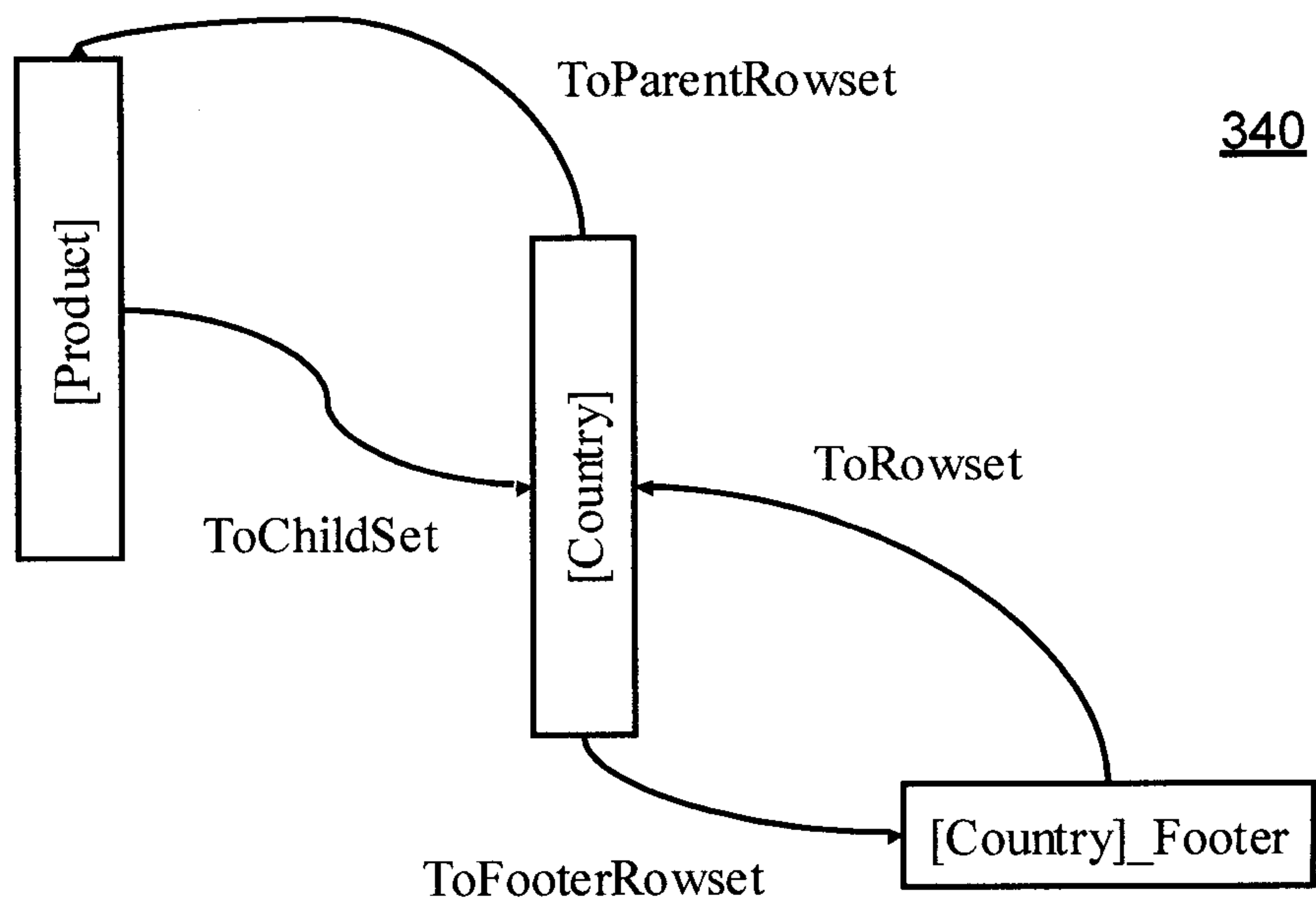


Figure 8

