(11) Application No. AU 2004201343 B2

(19) AUSTRALIAN PATENT OFFICE

(54) Title

A rendering independent persistence of information

(51)[6] International Patent Classification(s)

G06F 12/00 (2006.01) 15/00

G06F 15/00 (2006.01) 20060101ALI2006010

G06F 17/30 (2006.01) 1BMKR G06F

G06F 12/00 17/30

20060101AFI2005122 20060101ALI2005100

0BMJP G06F 8BMEP

(21) Application No: 2004201343 (22) Application Date: 2004.03.30

(30) Priority Data

(31) Number	(32) Date	(33) Country
10/404746	2003.03.31	US

(43) Publication Date : 2004.10.14

(43) Publication Journal Date : 2004.10.14

(71) Applicant(s)

Microsoft Corporation

(72) Inventor(s)

Bergstrom, Yee Man; Wang, Fang; Carlson, Jason

(74) Agent/Attorney

Davies Collison Cave, 1 Nicholson Street, Melbourne, VIC, 3000

(56) Related Art

US 2002/0143824

US 2002/0143521

EP 1122652

US 7053958

ABSTRACT OF THE DISCLOSURE

A system and methods providing a rendering independent persistence of information is provided. In an illustrative implementation, data is provided having some predefined structure. In operation, the data is processed such that the data and it's associated structure are represented in a data format that is persistent. In operation, the data is parsed according to one or more constraints and translated into the persistent data format.

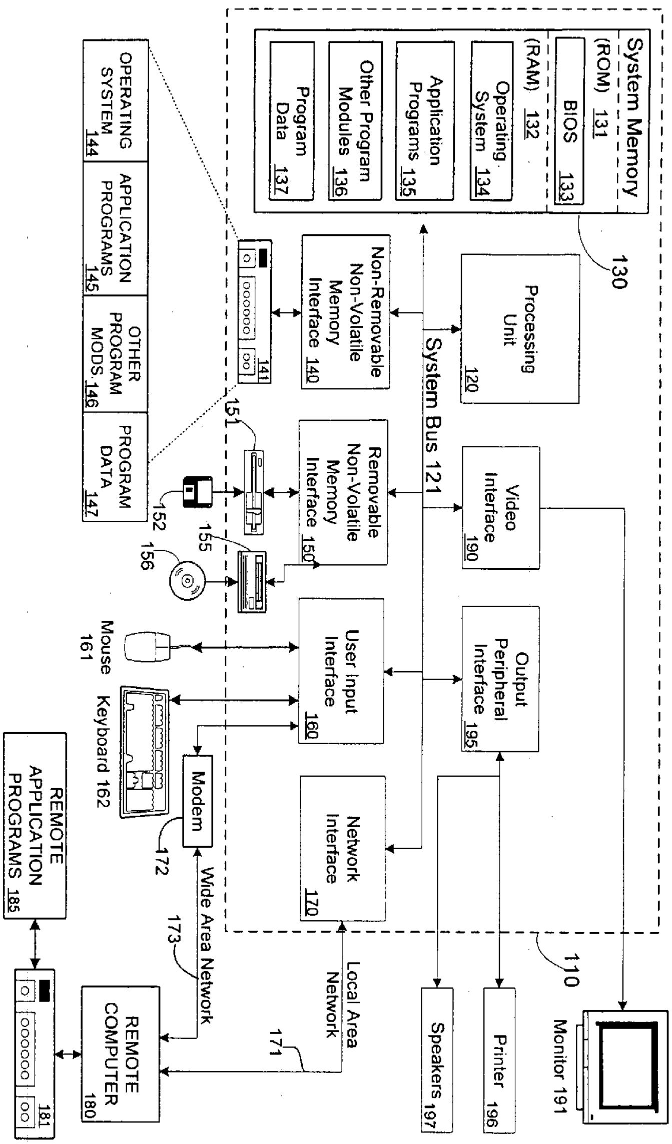
100
System Memory
(ROM) 131
BIOS 133
(RAM) 132
Operating System 134
Application Programs 135
Other Program Modules 136
Program Data 137
130
Processing Unit 120
System Bus 121
Video Interface 190
Output Peripheral Interface 195
Network Interface 170
Non-Removable Non-Volatile Memory Interface 140
Removable Non-Volatile Memory Interface 150
User Input Interface 160
141
151
152
155
156
OPERATING SYSTEM 144
APPLICATION PROGRAMS 145
OTHER PROGRAM MODS. 146
PROGRAM DATA 147
Mouse 161
Keyboard 162
Modem
172
Wide Area Network
173
Local Area Network
171
110
Monitor 191
Printer 196
Speakers 197
REMOTE COMPUTER 180
181
REMOTE APPLICATION PROGRAMS 185

Figure 1

AUSTRALIA

PATENTS ACT 1990

COMPLETE SPECIFICATION

NAME OF APPLICANT(S)::

Microsoft Corporation

ADDRESS FOR SERVICE:

DAVIES COLLISON CAVE
Patent Attorneys
1 Nicholson Street, Melbourne, 3000, Australia

INVENTION TITLE:

A rendering independent persistence of information

The following statement is a full description of this invention, including the best method of performing it known to me/us:-

5102

2004201343 27 Aug 2009

TECHNICAL FIELD

Embodiments relate generally to the field of data rendering, and, more particularly, to novel expression of a data in a rendering independent persistent data format that is more easily usable by a variety of cooperating computing applications and computing environments.

BACKGROUND

Data storage and format is ubiquitous with data management and data processing. In the context of computing applications, data is the lifeline. From simple content based data to complex embedded instruction sets, data acts as the input to most computing applications and is the resultant output of these computing applications. Not surprisingly, computing application designers and developers have developed staggering amounts of computing applications to create, manage, store, and process data that touches each of us in almost every aspect of our lives. From simple word processing applications to complicated encryption techniques for use in communicating sensitive data, data processing computing applications have become integrated within daily routines and practices. The development of the myriad of data operative computing applications has rendered an expected byproduct - simply a number of varying and disparate data formats and types.

With disparate data formats, it becomes increasingly difficult to share data between cooperating computing applications and computing environments that have their own native data formats and definitions. Addressing these concerns, computing application developers have developed and implemented various data filters and translators that allow them to accept non-native data formats. However, the integration and implementation of such data conversion mechanisms comes at an expense, namely, increased processing requirements and the loss of data integrity. Moreover, data conversion may not be available to each and every computing application seeking to process non-native data. As such, sharing desired data among cooperating computing applications is rendered difficult at best.

Data may be characterized by a rendering extension which is representative of the underlying format and/or layout. The rendering extension prompts cooperating applications of the data format and/or layout and, if appropriate, will trigger the conversion of data by the cooperating computing application to a native format native to the requesting cooperating computing application. The rendering extensions, generally, also provide an indication of which computing application or computing environment generated a piece or set of data. For example if a particular word processor computing application generated a data (e.g. a document), the rendering extension may be of the ".doc" variety. Comparatively, if a spreadsheet computing application generated some data (e.g. a spreadsheet, graph, etc.), then the rendering extension may be ".xls".

Currently, computing applications generally generate data (e.g. reports) having a single rendering extension (e.g. .html, .doc, .xls, .xml) definition that is usually native to the computing application that is generating the data. As such, cooperating applications, when processing reports, first are required to perform a translation of the foreign rendering extension to a native rendering extension. This translation step, in some instances, can introduce errors, that is, data layout/formatting error and, more importantly, data error. Moreover, the data, in such form, has limited utility to cooperating applications as the generated report is not easily queryable. In most instances, participating users will use the computing applications to generate new data having new report definitions instead of trying to reuse an already generated report.

Another drawback of existing practices is an inability to perform a time driven analysis of already generated data. As described, a computing application may operate on one or more cooperating data stores. These data stores have various tables having various field definitions. Over time the value of the fields will change to reflect one or more change in the organization and/or enterprise operating the data store. For example, a car dealership may employ a computing application cooperating with a data store to record sales. The sales values would change as more cars are sold. In the same example, the computing application may operate to generate a report to show the total sales by each of the sales staff of the car dealership. Once again with increasing sales, the report values change. Current data (e.g. report) generating computing applications operate such to gather the necessary data according to a definition and generate a data work product according to the report definition. However, the data work product

acts a snapshot of the values of the data fields found in the cooperating data store at the time of the data work product generation.

Moreover, current computing applications would expose the generated data work product as data construct that is probably not schematized, not stored in a non-persistent data format, and thus is not easily queryable. As such, these applications would not be able to support a time-dimensioned query on the historical data work products to provide a time-driven analysis of one or more of the data values. By storing data in a non-persistent rendering dependent format, current applications are incapable of performing a time driven analysis that may be used to determine trends.

Rendering independent persistent data formats have many applications outside of the report generation and management context. For example, a rendering independent persistent data format may be incorporated to communicate a variety of data, such as web content, across disparate computing application having their own native rendering requirements and standards.

From the foregoing it is desired to address or ameliorate one or more shortcomings or disadvantages of previous approaches or to provide at least a useful alternative and preferably a system and method that provides data in a rendering independent persistent format for use in a variety of processing that is not realized by current practices.

SUMMARY

Some embodiments relate to a method implemented at least in part by a computing device to provide a rendering independent persistence of information comprising the steps of:

providing a plurality of data sources, each said data source having a predefined data structure and associated with a time, each said predefined data structure comprising a corresponding schema definition;

translating each of the plurality of data sources according to the corresponding schema definition that yields corresponding rendering independent persisted data in a rendering independent persisted data format, wherein the translating step comprises the step of performing a binary translation of said each of the plurality of data sources; and

aggregating the rendering independent persisted data from each of the plurality of

2004201343 27 Aug 2009

data sources according to a time-based parameter to generate a new schematized queryable data source having the rendering independent persistent data format and the new schematized queryable data source comprising the aggregated rendering independent persisted data from each of the plurality of data sources according to the time-based parameter to allow a time driven query to be performed on the new schematized queryable data source.

Some embodiments relate to a computer readable storage medium having a tangible component with computer readable instructions stored thereon to instruct a computer to perform the following method:

providing a plurality of data sources, each said data source having a predefined data structure and associated with a time, each said predefined data structure comprising a corresponding schema definition;

translating each of the plurality of data sources according to the corresponding schema definition that yields corresponding rendering independent persisted data in a rendering independent persisted data format, wherein the translating step comprises the step of performing a binary translation of said each of the plurality of data sources; and

aggregating the rendering independent persisted data from each of the plurality of data sources according to a time-based parameter to generate a new schematized queryable data source having the rendering independent persistent data format and the new schematized queryable data source comprising the aggregated rendering independent persisted data from each of the plurality of data sources according to the time-based parameter to allow a time driven query to be performed on the new schematized queryable data source.

Some embodiments relate to a system providing a rendering independent persistence of information comprising:

at least one computing processor;

memory communicatively coupled with said at least one computing process, said memory comprising instructions executable by said at least one computing processor, said instructions comprising:

instructions for providing a plurality of data sources, each said data source having data, wherein the data has an associated data structure and is associated with a time, each

2004201343 27 Aug 2009

said data structure comprising a schema definition;

instructions for providing a processing module, the processing module operating on each of the plurality of the data sources to generate corresponding rendering persisted data in a rendering independent persisted data format, wherein the rendering independent persisted data format operates in various rendering specific environments, wherein the processing module executes according to the corresponding schema definition a binary translation of the data into the rendering independent persisted data format; and

instructions for aggregating the rendering independent persistence of information from each of the plurality of data sources according to a time-based parameter to generate a new schematized queryable data source having the rendering independent persistent data format and the new schematized queryable data source comprising the aggregated rendering independent persisted data from each of the plurality of data sources according to the time-based parameter to allow a time driven query to be performed on the new schematized queryable data source.

Some embodiments relate to, in a computing environment, a method providing a rendering independent persisted data format comprising:

receiving a plurality of data sources, each said data source associated with a time and comprising a schema definition;

parsing each of the data sources according to at least one predefined parsing rule;

translating the parsed data according to the schema definition into rendering independent persisted data having a rendering independent persisted data format, wherein the translating step comprises binary translating the parsed data; and

aggregating the rendering independent persisted data from each of the plurality of data sources according to a time-based parameter to generate a new schematized queryable data source having the rendering independent persistent data format and the new schematized queryable data source comprising the aggregated rendering independent persisted data from each of the plurality of data sources according to the time-based parameter to allow a time driven query to be performed on the new schematized queryable data source.

Some embodiments relate to a method comprising:

receiving a plurality of data sources, each said data source associated with a time and comprising a schema definition;

parsing each of the data sources according to at least one predefined parsing rule;

translating the parsed data according to the schema definition into rendering independent persisted data having a rendering independent persisted data format, wherein the translating step comprises binary translating the parsed data to an extension specific format of the requesting environment; and

aggregating the rendering independent persisted data from each of the plurality of data sources according to a time-based parameter to generate a new schematized queryable data source having the rendering independent persistent data format and the queryable schematized data source comprising the aggregated rendering independent persisted data from each of the plurality of data sources according to the time-based parameter to allow a time driven query to be performed on the new schematized queryable data source.

Embodiments relate to systems and methods to represent data as a rendering independent persistence of information. In an illustrative implementation, data having a predefined structure is provided. The data is processed such that a representation of the data is created wherein the representation includes information about the data and the data's structure. In operation, the data is parsed and converted to a predefined format that is persistent.

In a contemplated implementation, a generated data set is provided by the computing application in a rendering independent persistent data format. The rendering independent persistent data format, *inter alia*, allows for the application to perform time lapsed and time driven queries on the report exposed as schematized queryable data source, and more importantly, allows the report to be perceived as any other data source by other cooperating computing applications.

Other features and aspects of the herein described systems and methods are described in more detail below.

BRIEF DESCRIPTION OF THE DRAWINGS

Figures 1 and 2 are schematic illustrations of exemplary computing environments suitable for the present invention, with Figure 2 depicting an exemplary networked computing environment;

Figure 3 is a block diagram of showing an exemplary implementation of a report as a data source in accordance with the herein described system and methods;

Figure 4 is a block diagram depicting the flow of report processing in accordance with the herein described systems and methods;

Figure 5 is a block diagram depicting the flow of report utilization in accordance with the herein described system and methods;

Figure 6 is a detailed block diagram of exemplary components to process schematized data constructs in accordance with the herein described systems and methods;

Figure 7 is flow diagram of the processing performed to expose a report as a schematized queryable data source in accordance with the herein described systems and methods;

Figure 8A is a flow diagram of the processing performed to expose the data source as a rendering independent persistence of information;

Figure 8B is a flow diagram of the processing performed when translating the rendering independent persistence of information as a report exposed as a schematized queryable data source; and

Figure 9 is a flow diagram of the processing performed when processing data exposed as a schematized queryable data source to generate a desired snapshot.

2004201343 27 Aug 2009

DETAILED DESCRIPTION

Overview

The persistence of data is tantamount to effective data warehousing and data processing. Currently, computing applications and computing environments operate on data to generate one or more data work products that can be described by one or more rendering extensions. The rendering extension, generally, may be used to characterize the format and definition of the data found in data work products. Typically, a computing application will operate on a set of data in a data processing mode to generate data work products having a particular rendering extension. For example, a word processing computing application may operate on text type data to generate formatted documents. Such formatted documents may then be stored as data having the ".doc" rendering type.

A problem arises however when cooperating computing applications and computing environments vie for data that has a non-native data format or definition (i.e. stored and exposed having different and varying rendering extensions). Typically, the computing applications, if equipped, will call upon one or more data conversion operations to translate the desired data work product into a native rendering extension. However, such conversion can be processing intensive and can introduce significant error into the final data work product.

Described embodiments aim to ameliorate the shortcomings of existing practices by providing a system and methods that expose data having a rendering independent persistent data format. Specifically, an exemplary computing application is provided that operates on data according to a predetermined definition. The data definition contains, *inter alia,* information relevant to the desired data to be provided, data layout information, and data formatting information. The exemplary computing application retrieves the desired data and data definition from the cooperating data store. Once gathered, the exemplary computing application defines a schema for the data and stores the data in an intermediate data format. The intermediate data format is a rendering independent persistent data format. As such, the retrieved data having an associated schema is exposed as a data source with all of the benefits a data source provides to cooperating applications. Moreover, being stored as a rendering independent persistent data format, current reusability (i.e. usability between disparate computing applications and computing environments having varying data rendering extension requirements) issues are resolved,

2004201343 27 Aug 2009

as well as, with the present system and methods, time driven queries are more easily executable against a set of generated reports.

It is appreciated that although the herein described systems and methods are described in the context of the generation of one or more data work products, that the rendering independent persistence of information may be utilized in various ways that go beyond the scope of the provided examples.

A. Exemplary Computing Environment

Figure 1 illustrates an example of a suitable computing system environment 100 in which embodiments may be implemented. The computing system environment 100 is only one example of a suitable computing environment and is not intended to suggest any limitation as to the scope of use or functionality of the invention. Neither should the computing environment 100 be interpreted as having any dependency or requirement relating to any one or combination of components illustrated in the exemplary operating environment 100.

Embodiments are operational with numerous other general purpose or special purpose computing system environments or configurations. Examples of well known computing systems, environments, and/or configurations that may be suitable for use with the invention include, but are not limited to, personal computers, server computers, hand-held or laptop devices, multiprocessor systems, microprocessor-based systems, set top boxes, programmable consumer electronics, network PCs, minicomputers, mainframe computers, distributed computing environments that include any of the above systems or devices, and the like.

Embodiments may be described in the general context of computer-executable instructions, such as program modules, being executed by a computer. Generally, program modules include routines, programs, objects, components, data structures, etc. that perform particular tasks or implement particular abstract data types. Embodiments may also be practiced in distributed computing environments where tasks are performed by remote processing devices that are linked through a communications network or other data transmission medium. In a distributed computing environment, program modules and other data may be located in both local and remote computer storage media including memory storage devices.

With reference to Figure 1, an exemplary system for implementing embodiments

2004201343 27 Aug 2009

includes a general purpose computing device in the form of a computer 110. Components of computer 110 may include, but are not limited to, a processing unit 120, a system memory 130, and a system bus 121 that couples various system components including the system memory to the processing unit 120. The system bus 121 may be any of several types of bus structures including a memory bus or memory controller, a peripheral bus, and a local bus using any of a variety of bus architectures. By way of example, and not limitation, such architectures include Industry Standard Architecture (ISA) bus, Micro Channel Architecture (MCA) bus, Enhanced ISA (EISA) bus, Video Electronics Standards Association (VESA) local bus, and Peripheral Component Interconnect (PCI) bus (also known as Mezzanine bus).

Computer 110 typically includes a variety of computer readable media. Computer readable media can be any available media that can be accessed by computer 110 and includes both volatile and non-volatile media, removable and non-removable media. By way of example, and not limitation, computer readable media may comprise computer storage media and communication media. Computer storage media includes both volatile and non-volatile, removable and 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. Computer storage media includes, but is not limited to, RAM, ROM, EEPROM, flash memory or other memory technology, CD-ROM, digital versatile disks (DVD) or other optical disk storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other medium which can be used to store the desired information and which can be accessed by computer 110. Communication media typically embodies computer readable instructions, data structures, program modules or other data in a modulated data signal such as a carrier wave or other transport mechanism and includes any information delivery media. The term "modulated data signal" means a signal that has one or more of its characteristics set or changed in such a manner as to encode information in the signal. By way of example, and not limitation, communication media includes wired media such as a wired network or direct-wired connection, and wireless media such as acoustic, RF, infrared and other wireless media. Combinations of any of the above should also be included within the scope of computer readable media.

The system memory 130 includes computer storage media in the form of volatile and/or non-volatile memory such as ROM 131 and RAM 132. A basic input/output system 133 (BIOS), containing the basic routines that help to transfer information between elements within computer 110, such as during start-up, is typically stored in ROM 131. RAM 132 typically contains data and/or program modules that are immediately accessible to and/or presently being operated on by processing unit 120. By way of example, and not limitation, Figure 1 illustrates operating system 134, application programs 135, other program modules 136, and program data 137.

The computer 110 may also include other removable/non-removable, volatile/non-volatile computer storage media. By way of example only, Figure 1 illustrates a hard disk drive 140 that reads from or writes to non-removable, non-volatile magnetic media, a magnetic disk drive 151 that reads from or writes to a removable, non-volatile magnetic disk 152, and an optical disk drive 155 that reads from or writes to a removable, non-volatile optical disk 156, such as a CD-ROM or other optical media. Other removable/non-removable, volatile/non-volatile computer storage media that can be used in the exemplary operating environment include, but are not limited to, magnetic tape cassettes, flash memory cards, digital versatile disks, digital video tape, solid state RAM, solid state ROM, and the like. The hard disk drive 141 is typically connected to the system bus 121 through a non-removable memory interface such as interface 140, and magnetic disk drive 151 and optical disk drive 155 are typically connected to the system bus 121 by a removable memory interface, such as interface 150.

The drives and their associated computer storage media, discussed above and illustrated in Figure 1, provide storage of computer readable instructions, data structures, program modules and other data for the computer 110. In Figure 1, for example, hard disk drive 141 is illustrated as storing operating system 144, application programs 145, other program modules 146, and program data 147. Note that these components can either be the same as or different from operating system 134, application programs 135, other program modules 136, and program data 137. Operating system 144, application programs 145, other program modules 146, and program data 147 are given different numbers here to illustrate that, at a minimum, they are different copies. A user may enter commands and information into the computer 110 through input devices such as a keyboard 162 and pointing device 161, commonly referred to as a mouse, trackball or touch pad. Other input devices (not shown) may include a microphone, joystick,

game pad, satellite dish, scanner, or the like. These and other input devices are often connected to the processing unit 120 through a user input interface 160 that is coupled to the system bus, but may be connected by other interface and bus structures, such as a parallel port, game port or a universal serial bus (USB). A monitor 191 or other type of display device is also connected to the system bus 121 via an interface, such as a video interface 190. In addition to the monitor, computers may also include other peripheral output devices such as speakers 197 and printer 196, which may be connected through an output peripheral interface 190.

The computer 110 may operate in a networked environment using logical connections to one or more remote computers, such as a remote computer 180. The remote computer 180 may be a personal computer, a server, a router, a network PC, a peer device or other common network node, and typically includes many or all of the elements described above relative to the computer 110, although only a memory storage device 181 has been illustrated in Figure 1. The logical connections depicted include a local area network (LAN) 171 and a wide area network (WAN) 173, but may also include other networks. Such networking environments are commonplace in offices, enterprise-wide computer networks, intranets and the Internet.

When used in a LAN networking environment, the computer 110 is connected to the LAN 171 through a network interface or adapter 170. When used in a WAN networking environment, the computer 110 typically includes a modem 172 or other means for establishing communications over the WAN 173, such as the Internet. The modem 172, which may be internal or external, may be connected to the system bus 121 via the user input interface 160, or other appropriate mechanism. In a networked environment, program modules depicted relative to the computer 110, or portions thereof, may be stored in the remote memory storage device. By way of example, and not limitation, Figure 1 illustrates remote application programs 185 as residing on memory device 181. It will be appreciated that the network connections shown are exemplary and other means of establishing a communications link between the computers may be used.

B. Exemplary Networked Computing Environment

Computer 20a, described above, can be deployed as part of a computer network. In general, the above description for computers applies to both server computers and client

computers deployed in a network environment. Figure 1A illustrates an exemplary network environment, with a server in communication with client computers via a network, in which the present invention may be employed. As shown in Figure 1A, a number of servers 10a, 10b, etc., are interconnected via a communications network 160 (which may be a LAN, WAN, intranet, the Internet, or other computer network) with a number of client computers 20a, 20b, 20c, or computing devices, such as, mobile phone 15, land-line telephone 16, and personal digital assistant 17. In a network environment in which the communications network 160 is the Internet, for example, the servers 10 can be Web servers with which the clients 20 communicate via any of a number of known protocols, such as, hypertext transfer protocol (HTTP) or wireless application protocol (WAP). Each client computer 20 can be equipped with browser 180a to gain access to the servers 10. Similarly, personal digital assistant 17 can be equipped with browser 180b and mobile phone 15 can be equipped with browser 180c to display and receive various data.

In operation, a user (not shown) may interact with a computing application running on a client computing devices to expose a report as a schematized queryable data source. The reports may be stored on server computers and communicated to cooperating users through client computing devices over communications network 160. A user may generate, manage, and interact with such reports by interfacing with computing applications on client computing devices. These transactions may be communicated by client computing devices to server computers for processing and storage. Server computers may host computing applications to expose reports as queryable schematized data sources.

Thus, the present invention can be utilized in a computer network environment having client computing devices for accessing and interacting with the network and a server computer for interacting with client computers. However, the systems and methods described herein can be implemented with a variety of network-based architectures, and thus should not be limited to the example shown. The herein described systems and methods will now be described in more detail with reference to a presently illustrative implementation.

C. Reports As Data Sources

Figure 3 shows a block diagram of an exemplary illustrative architecture of an exemplary report generation and management system that exposes a report . A shown, the exemplary architecture 300 comprises report server 320. Report server 320 further comprises report processing engine 325, analysis services engine 330, and report intermediate format 335. The report intermediate format 335 further comprises data 345 and schema 340. Lastly, exemplary architecture 400 comprises report viewer/browser 305 and OLEDB/ADO 310. In operation, report server 320 generates schema 340 that operates on report data 345. The report server 320 generates report intermediate format which is a schematized queryable data source having a rendering independent persisted data format. The intermediate data format can then be used by report server 320 to display the generated report via 305. In this context, the intermediate format 335 is processed by the report processing engine 325 of the report server 320 to display the report on report viewer/browser 305. The intermediate format can also be used by report server 320 to communicate the generated report to cooperating environments via analysis services engine 330 using OLEDB module 310.

By having the report in an intermediate format that is schematized, the report looks like and acts like a data source to cooperating environments and cooperating computing applications.

D. Exposing Data Work Products as Schematized Data Sources

Figure 4 shows the exemplary data flow between exemplary components of a data work product (e.g. report) generation and management system that expose reports as schematized queryable data sources. As shown, report generation and management system 400 comprises various components for use in exposing reports as schematized queryable data sources. Specifically, report generation and management system 400 comprises report processing module 420. As is shown, report processing module 420 cooperates with report definition 415 and accepts data from external data sources 405 and 410 to generate reports that are stored in an intermediate format 425.

In operation, a generated report may be requested by one or more cooperating environments. In this context, the report, stored in an intermediate data format, is communicated

to an event processing module 430 which coordinates the communication of generated reports, in whole or in part, to requesting environments. Event processing module 430 determines the rendering of the requesting environment and provides the report in whole or in part to the requesting environment in a rendering extension native to the requesting environment. For example, if html rendering is required html rendering extensions 435 are employed. Comparatively, if XML rendering is required, XML rendering extension 440 is employed. And so on, such that other rendering types would be represented by other rendering extensions 445.

Figure 5 shows an exemplary high level deployment of an exemplary report generation and management system 500. As shown report generation and management system 500 comprises report processing module 525. Report processing module 525 comprises data extensions 525 and rendering extensions 530. Furthermore, report processing module cooperates with report definition 510 and data source 505. In operation, a request for a report is provided to the report processing module 525. The report processing module 525 obtains the proper report definition from report definitions filed 510 for application to data found in data source 505. The data is then processed by report processing module 525 using data extension 535 to identify the data elements from data source 505. Report processing module 525 then processes the data according to the appropriate report definition to generate a report that is exposed as a schematized queryable data source. The report is then stored by the report processing module 525 in an intermediate data format 520 for future use.

In addition to generating reports, report generation and management system 500 is capable of communicated generated reports to cooperating environments in spite of the rendering required by the required by the requesting environments. For example, report generation and management system 500 may be employed to communicate a generated report to a cooperating environment. In this context, the generated report stored in an intermediate format 520 is retrieved by report processing module 525 and processed using rendering extensions 530 to generate a report in a rendering format acceptable by the requesting environment.

Figure 6 shows a more detailed exemplary deployment of an exemplary report generation and management system capable of exposing a report as a schematized queryable data source having a rendering independent persistent data format. As shown report generation and management system 600 comprises a report server 605. Report server 605 in turn comprises

mapping module 610, report processing module 615 and query processing optimization/execution engine 625. Report server 605 cooperates with various cooperating components including but not limited to report user interface (UI) 630, report definitions 650, a second report server 635, Analysis Services (AS) data provider 640, and pivot control component 645.

When generating a report, report server 605 cooperates with report definitions 650 to obtain the appropriate report definition for the desired report. The report definition is then processed by report processing module 615 of report server 605. Report server 605, using the appropriate report definition gathers the appropriate data and generates a schematized queryable data source representative of the desired report. The report can then be displayed on the report UI 630 by report server 605. In this contemplated operation, report processing module 615 of report server 605 cooperates with mapping module 610 to map the desired report for viewing and display on report UI 630.

In operation, report generation and management system 600 may support a number of operations and functions. For example a report be run on a set of already generated reports. As shown in Figure 6 by the arrow a request for a report on a report is provided to a report server 635. The report server 635 processes the report on a report request and cooperates with AS data provider module to fulfill the request. In turn, AS data provider 640 cooperates with query processing optimization/execution engine 625. This engine cooperates with report processing module 615 to obtain/generate the necessary data to satisfy the report on a report request. Similarly, a request for an OWC on a report may be provided to pivot control module 645 which in turn cooperates with AS data provider 640 to fulfill the OWC on the report request.

Another use of report generation and management system 600 is to allow participating users to view reports on report UI 630 originating from remote cooperating environments. In this context, the report UI 630 cooperates with report processing module 615. Report processing module 615 cooperates with mapping module 610. Mapping module 610 operates to translate data from one data format to another. As such, it may be used to translate data in a rendering independent data format to a rendering dependent data format for rendering in requesting environment. The mapping module 610 then cooperates with the query processing optimization/execution engine 625 for communication to one or more cooperating modules.

It is appreciated that report server 605 in the contemplated implementations can comprise any of computing hardware, computing software, and the combination of computing hardware and computing software.

Figure 7 shows a flowchart of exemplary processing performed to expose a report as a schematized queryable data source and the subsequent processing performed in utilizing such exposed report. As shown processing begins at block 700 and proceeds to block 705 where the report definition is obtained. At block 710 a schema is created for the report according the obtained report definition. The report data is then obtained at block 715 and the created schema, created at block 710, is applied to the report data at block 720. The schematized report is then stored in an intermediate format at block 725 (according the to processing described in Figures 8A and 8B). From there the processing proceeds to block 730 where a check is performed to determine if a report has been requested (e.g. requested by the native environment or by a cooperating environment). In the context of the examples provided, an environment comprises ay of a computing environment and a partial computing environment. If a report has been requested, processing proceeds to block 735 where data extensions are applied to the report and data extensions provided at block 740. In a contemplated illustrative implementation, the data extensions are applied to the data of the report to assist to identify the definition of the data fields. The rendering extensions, as described above, are used to translate the report for use in the rendering format of the environment using the report. The report is then rendered at block 745 for display in the environment requesting the report. Processing then terminates at block 750. If, however, at block 730 a report is not requested, processing proceeds to block 750 and terminates.

E. Rendering Independent Persistence of Information

Figure 8A shows a flowchart of exemplary processing performed to provide an exposed report in a rendering independent persistent data format. As shown, processing begins at block 800 and proceeds to block 805 where the report schema of the exposed schematized queryable report is obtained. Processing then proceeds to block 810 where the rendering independent persistent data format is identified. In an illustrative implementation, the rendering independent persistent data format comprises a binary data format. In context to block 810, the processing of

the herein described methods in accordance with the provided example contemplate the identification of binary representation of the schema. Further to the provided example, processing proceeds to block 815 where the identified binary schema is applied to the desired report to translate the schematized report into a binary data format. Included in the translation step is the execution of a data parsing process that is defined by at least one parsing rule which acts to separate the report data for processing. Processing then proceeds to block 820 where the generated binary representation is provided as an intermediate data format in which the exposed report may exist. Processing then terminates at block 825.

It is appreciated that although the exemplary implementation contemplates the use of a binary representation of the rendering independent persistent data format that the inventive concepts disclosed herein extend beyond the provided illustrative implementations to include but not limited to hexadecimal representation, assembly language representations, and high level programming language representations.

Figure 8B shows a flowchart of the exemplary processing performed when processing requests by cooperating environments to retrieve an exposed report, in whole or in part. for use in the requesting environment. As shown, processing begins at block 830 and proceeds to block 835 where the desired report (in whole or in part) is requested and is provided in its intermediate format (e.g. rendering independent persistent data format). From there, processing proceeds to block 840 where the intermediate format, a rendering independent format, to a rendering dependent format, namely, the rendering format of the requesting environment. In an illustrative implementation, the translation step contemplates the binary translation of the report schema to the extension specific format of the requesting environment. At block 845, the schematized report and accompanying data are extracted for presentation in the rendering extension of the requesting environment. The final report is then provided at block 850. Processing then terminates at block 850.

Figure 9 shows a flow chart of exemplary processing performed when performing a time-based query is performed against a set of generated reports. As shown, processing begins at block 900 and proceeds to block 905 where the parameters of the desired snapshot view is provided. From there processing proceeds to block 910 where the schematized queryable reports are processed as data sources. At block 915, the data is then aggregated from the set of

2004201343 27 Aug 2009

generated reports according to the provided parameters (e.g. collect all sales values from year 1 to year 2 in the Midwest geographic region across all sales reports from year 1 to year 2). The aggregated data is then collected and processed at block 920 to provide a new schematized queryable report having the desired snapshot data. Processing the proceeds to block 925 where it terminates.

The processing described in Figure 9A employs one or more features of the herein described systems and methods. Specifically, the herein described systems and methods contemplate a mechanism that exposes a report as a schematized queryable data source having a rendering independent persistent data format. The reports being schematized, queryable, and persistent, a time-based query (e.g. trend snapshot) is easily processed on a set of such reports as data is collected over a time value identified from the report's schema. The values are reliable as the report is stored in a persistent data format.

F. Conclusion

As mentioned above, while exemplary embodiments of the present invention have been described in connection with various computing devices and network architectures, the underlying concepts may be applied to any computing device or system in which it is desirable to traverse and/or perform other functions in connection with data. Thus, the processes and systems described above may be applied to a variety of applications and devices. While exemplary data structures, programming languages, names and examples are chosen herein as representative of various choices, these are not intended to be limiting.

The various techniques described herein may be implemented in connection with hardware or software or, where appropriate, with a combination of both. Thus, the methods and apparatus of the described embodiments, or certain aspects or portions thereof, may take the form of program code (i.e., instructions) embodied in tangible media, such as floppy diskettes, CD-ROMs, hard drives, or any other machine-readable storage medium, wherein, when the program code is loaded into and executed by a machine, such as a computer, the machine becomes an apparatus for practicing the invention. In the case of program code execution on programmable computers, the computing device will generally include a processor, a storage medium readable by the processor (including volatile and non-volatile memory and/or storage elements), at least one input device, and at

2004201343 27 Aug 2009

least one output device. One or more programs that may utilize the debugging interface aspects of the present invention, e.g., through the use of a data processing API or the like, are preferably implemented in a high level procedural or object-oriented programming language to communicate with a computer system. However, the program(s) can be implemented in assembly or machine language, if desired. In any case, the language may be a compiled or interpreted language, and combined with hardware implementations.

The described methods and apparatus may also be practiced via communications embodied in the form of program code that is transmitted over some transmission medium, such as over electrical wiring or cabling, through fiber optics, or via any other form of transmission, wherein, when the program code is received and loaded into and executed by a machine, such as an EPROM, a gate array, a programmable logic device (PLD), a client computer, a video recorder or the like, or a receiving machine having the debugging capabilities as described in exemplary embodiments above becomes an apparatus for practicing such embodiments. When implemented on a general-purpose processor, the program code combines with the processor to provide a unique apparatus that operates to invoke the functionality of the described embodiments. Additionally, any storage techniques used in connection with the embodiments may invariably be a combination of hardware and software.

While some embodiments have been described, it is to be understood that other similar embodiments may be used or modifications and additions may be made to the described embodiments for performing the same functions. For example, one skilled in the art will recognize that embodiments as described in the present application may apply to any computing device or environment, whether wired or wireless, and may be applied to any number of such computing devices connected via a communications network, and interacting across the network. Furthermore, it should be emphasized that a variety of computer platforms, including handheld device operating systems and other application specific operating systems are contemplated, especially as the number of wireless networked devices continues to proliferate. Still further, embodiments may be implemented in or across a plurality of processing chips or devices, and storage may similarly be effected across a plurality of devices. Therefore,

the present invention should not be limited to any single embodiment, but rather should be construed in breadth and scope in accordance with the appended claims.

Throughout this specification and the claims which follow, unless the context requires otherwise, the word "comprise", and variations such as "comprises" and "comprising", will be understood to imply the inclusion of a stated integer or step or group of integers or steps but not the exclusion of any other integer or step or group of integers or steps.

The reference to any prior art in this specification is not, and should not be taken as, an acknowledgement or any form of suggestion that that prior art forms part of the common general knowledge in Australia.

2004201343 27 Aug 2009

THE CLAIMS DEFINING THE INVENTION ARE AS FOLLOWS:

1. A method implemented at least in part by a computing device to provide a rendering independent persistence of information comprising the steps of:

providing a plurality of data sources, each said data source having a predefined data structure and associated with a time, each said predefined data structure comprising a corresponding schema definition;

translating each of the plurality of data sources according to the corresponding schema definition that yields corresponding rendering independent persisted data in a rendering independent persisted data format, wherein the translating step comprises the step of performing a binary translation of said each of the plurality of data sources; and

aggregating the rendering independent persisted data from each of the plurality of data sources according to a time-based parameter to generate a new schematized queryable data source having the rendering independent persistent data format and the new schematized queryable data source comprising the aggregated rendering independent persisted data from each of the plurality of data sources according to the time-based parameter to allow a time driven query to be performed on the new schematized queryable data source.

2. The method as recited in claim 1, further comprising operating on the rendering independent persistent data format to produce rendering specific data.

3. The method as recited in claim 1 or claim 2, further comprising parsing the data according to some predefined parsing step.

4. A computer readable storage medium having a tangible component with computer readable instructions stored thereon to instruct a computer to perform the following method:

providing a plurality of data sources, each said data source having a predefined data structure and associated with a time, each said predefined data structure comprising a corresponding schema definition;

2004201343 27 Aug 2009

translating each of the plurality of data sources according to the corresponding schema definition that yields corresponding rendering independent persisted data in a rendering independent persisted data format, wherein the translating step comprises the step of performing a binary translation of said each of the plurality of data sources; and

aggregating the rendering independent persisted data from each of the plurality of data sources according to a time-based parameter to generate a new schematized queryable data source having the rendering independent persistent data format and the new schematized queryable data source comprising the aggregated rendering independent persisted data from each of the plurality of data sources according to the time-based parameter to allow a time driven query to be performed on the new schematized queryable data source.

5. A system providing a rendering independent persistence of information comprising:

at least one computing processor;

memory communicatively coupled with said at least one computing process, said memory comprising instructions executable by said at least one computing processor, said instructions comprising:

instructions for providing a plurality of data sources, each said data source having data, wherein the data has an associated data structure and is associated with a time, each said data structure comprising a schema definition;

instructions for providing a processing module, the processing module operating on each of the plurality of the data sources to generate corresponding rendering persisted data in a rendering independent persisted data format, wherein the rendering independent persisted data format operates in various rendering specific environments, wherein the processing module executes according to the corresponding schema definition a binary translation of the data into the rendering independent persisted data format; and

instructions for aggregating the rendering independent persistence of information from each of the plurality of data sources according to a time-based parameter to generate a new schematized queryable data source having the rendering independent persistent data format and the new schematized queryable data source comprising the aggregated rendering independent persisted data from each of the plurality of data sources according

to the time-based parameter to allow a time driven query to be performed on the new schematized queryable data source.

6. The system as recited in claim 5, wherein the system comprises a computing environment.

7. The system as recited in claim 5 or claim 6, wherein the data comprises report data.

8. The system as recited in claim 7, wherein the report data comprises a schematized queryable data source.

9. The system as recited in claim 8, wherein the report data is aggregated according to at least one report definition.

10. In a computing environment, a method providing a rendering independent persisted data format comprising:

receiving a plurality of data sources, each said data source associated with a time and comprising a schema definition;

parsing each of the data sources according to at least one predefined parsing rule;

translating the parsed data according to the schema definition into rendering independent persisted data having a rendering independent persisted data format, wherein the translating step comprises binary translating the parsed data; and

aggregating the rendering independent persisted data from each of the plurality of data sources according to a time-based parameter to generate a new schematized queryable data source having the rendering independent persistent data format and the new schematized queryable data source comprising the aggregated rendering independent persisted data from each of the plurality of data sources according to the time-based parameter to allow a time driven query to be performed on the new schematized queryable data source.

11. A method comprising:

receiving a plurality of data sources, each said data source associated with a time and comprising a schema definition;

parsing each of the data sources according to at least one predefined parsing rule;

translating the parsed data according to the schema definition into rendering independent persisted data having a rendering independent persisted data format, wherein the translating step comprises binary translating the parsed data to an extension specific format of the requesting environment; and

aggregating the rendering independent persisted data from each of the plurality of data sources according to a time-based parameter to generate a new schematized queryable data source having the rendering independent persistent data format and the queryable schematized data source comprising the aggregated rendering independent persisted data from each of the plurality of data sources according to the time-based parameter to allow a time driven query to be performed on the new schematized queryable data source.

12. Computer readable storage storing instructions which, when executed by at least one processor, cause the at least one processor to perform the method of any one of claims 1 to 3, 10 and 11.

13. A system comprising means for performing the method of any one of claims 1 to 3, 10 and 11.

14. A method substantially as hereinbefore described with reference to the drawings.

15. A system substantially as hereinbefore described with reference to the drawings.

16. A computer readable medium substantially as hereinbefore described with reference to the drawings.

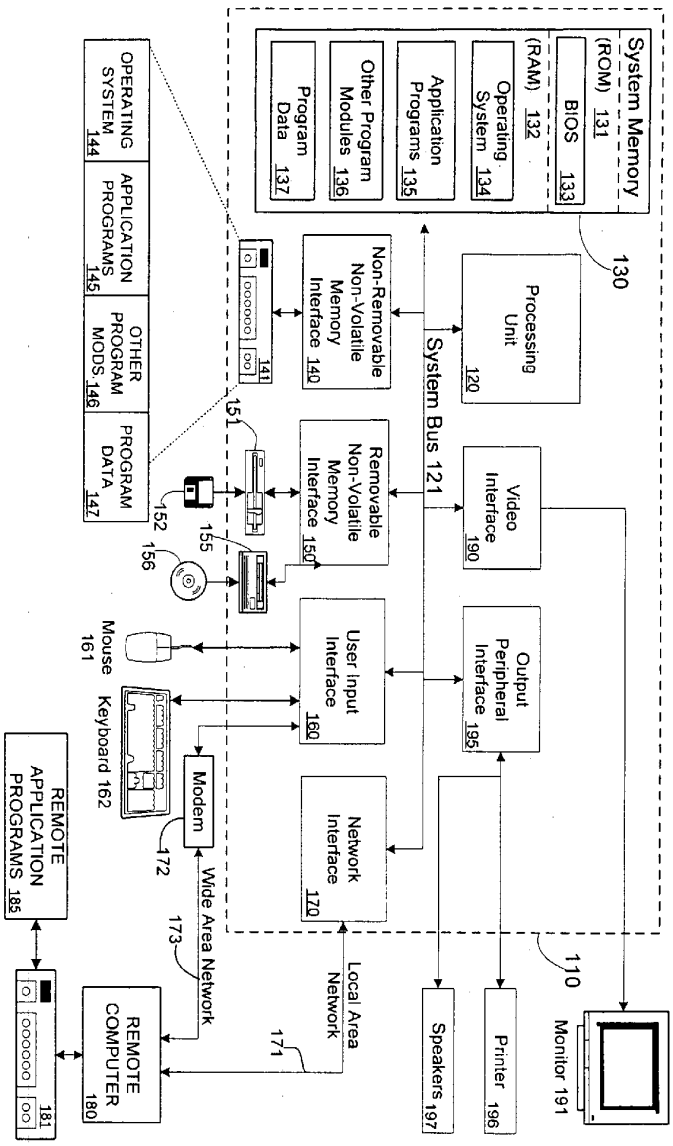
100
System Memory
(ROM) 131
BIOS 133
(RAM) 132
Operating System 134
Application Programs 135
Other Program Modules 136
Program Data 137
130
Processing Unit 120
System Bus 121
Non-Removable Non-Volatile Memory Interface 140
Removable Non-Volatile Memory Interface 150
Video Interface 190
User Input Interface 160
Output Peripheral Interface 195
Network Interface 170
Modem
172
Wide Area Network
173
Local Area Network
171
Monitor 191
Printer 196
Speakers 197
110
REMOTE COMPUTER 180
REMOTE APPLICATION PROGRAMS 185
181
Mouse 161
Keyboard 162
OPERATING SYSTEM 144
APPLICATION PROGRAMS 145
OTHER PROGRAM MODS 146
PROGRAM DATA 147
141
151
152
155
156

Figure 1

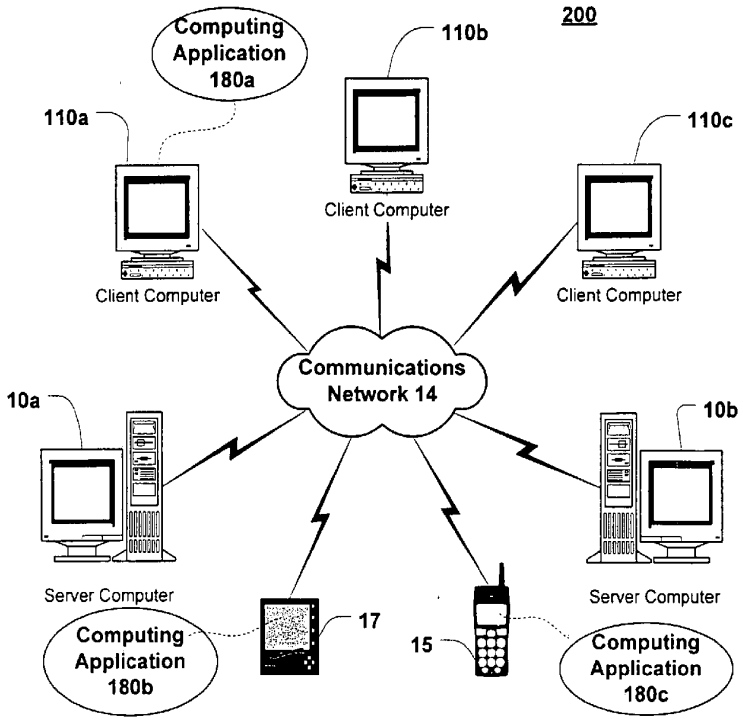
200
Computing Application 180a
110b
110a
110c
Client Computer
Client Computer
Client Computer
Communications Network 14
10a
10b
Server Computer
Server Computer
Computing Application 180b
17
15
Computing Application 180c

Figure 2

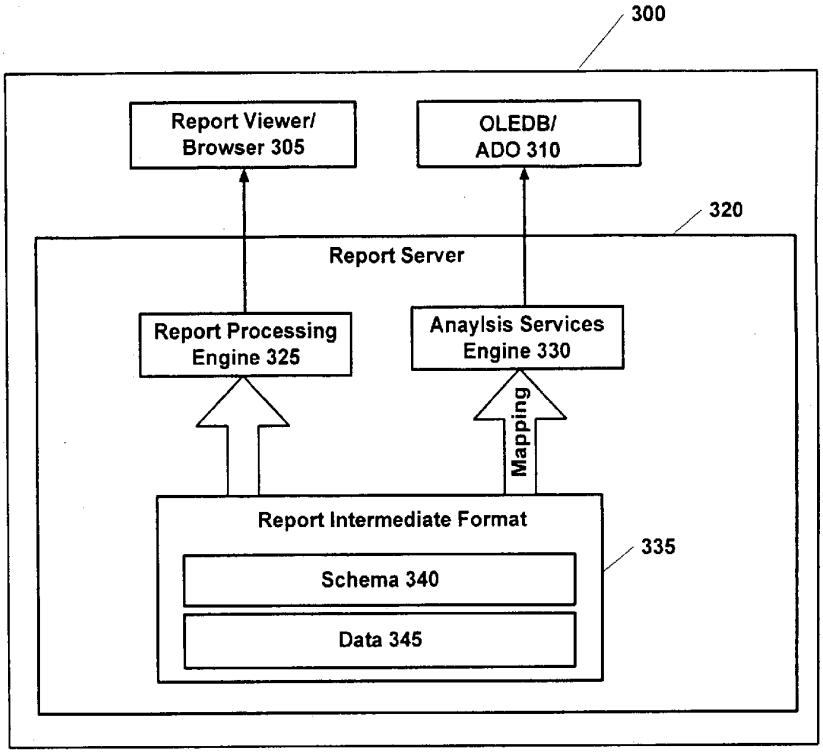
300
Report Viewer/
Browser 305
OLEDB/
ADO 310
320
Report Server
Report Processing
Engine 325
Anaylsis Services
Engine 330
Mapping
Report Intermediate Format
335
Schema 340
Data 345

Figure 3

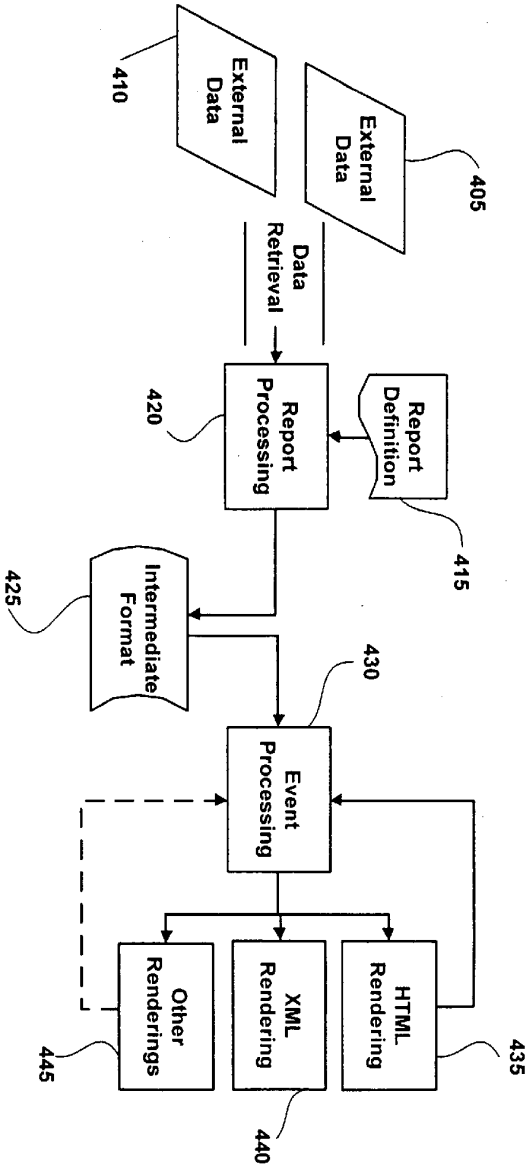
400
External Data
410
External Data
405
Data Retrieval
Report Processing
420
Report Definition
415
Intermediate Format
425
430
Event Processing
Other Renderings
445
XML Rendering
440
HTML Rendering
435

Figure 4

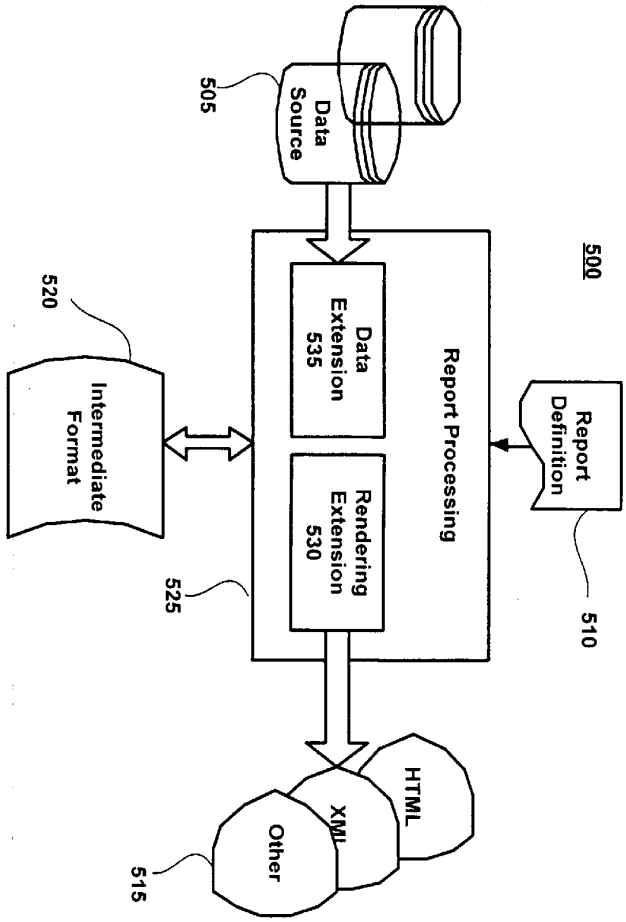
505
Data Source
500
520
Intermediate Format
Data Extension 535
Report Processing
Report Definition
Rendering Extension 530
525
510
HTML
XML
Other
515

Figure 5

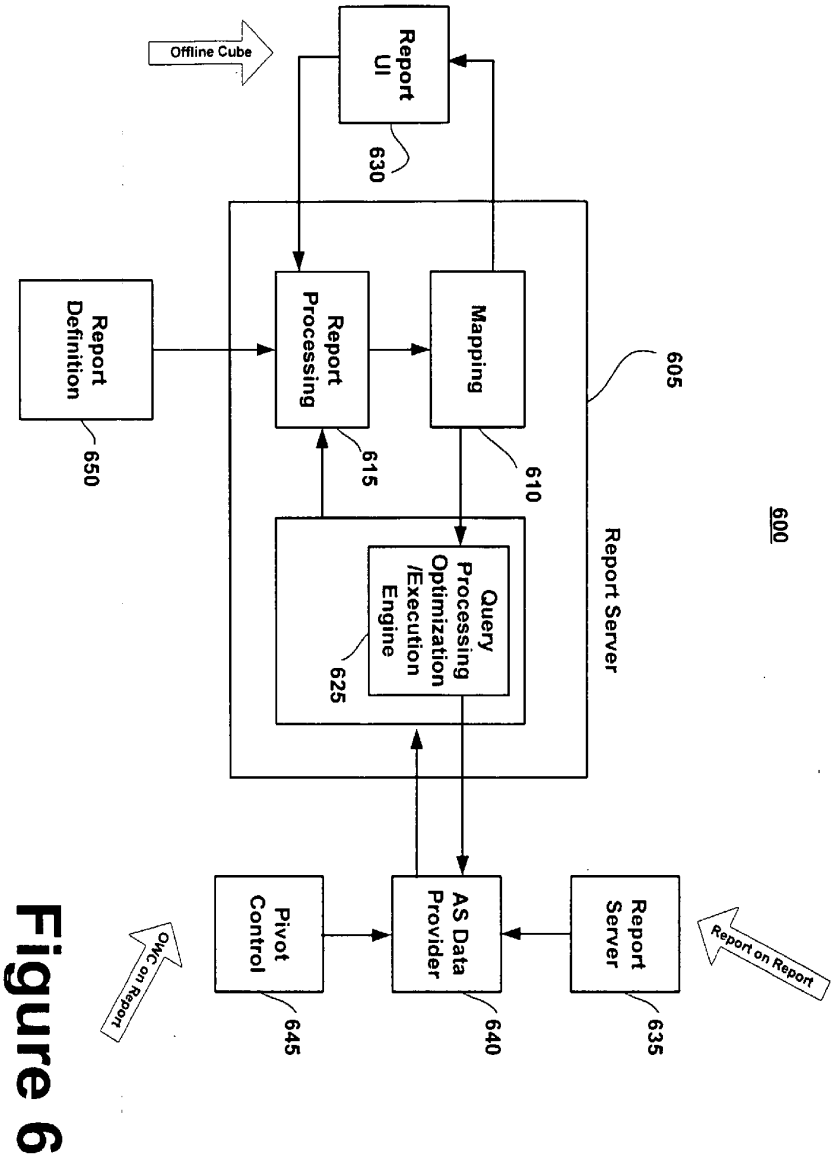
Offline Cube
Report UI
630
Report Definition
650
Report Processing
615
Mapping
610
605
Report Server
600
Query Processing Optimization /Execution Engine
625
Pivot Control
645
AS Data Provider
640
Report Server
635
OWC on Report
Report on Report

Figure 6

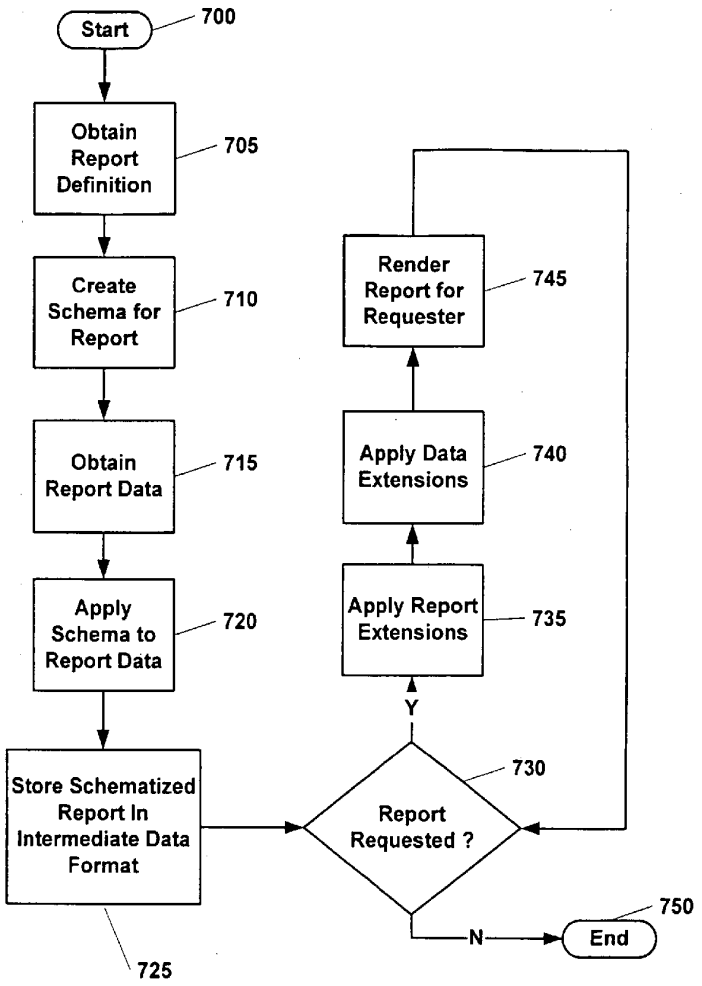
Start
700
Obtain Report Definition
705
Create Schema for Report
710
Obtain Report Data
715
Apply Schema to Report Data
720
Store Schematized Report In Intermediate Data Format
725
Report Requested ?
730
Y
N
Apply Report Extensions
735
Apply Data Extensions
740
Render Report for Requester
745
End
750

Figure 7

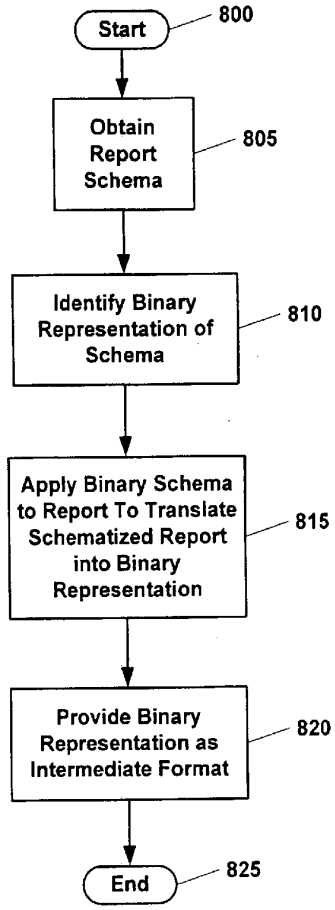
Start
800
Obtain Report Schema
805
Identify Binary Representation of Schema
810
Apply Binary Schema to Report To Translate Schematized Report into Binary Representation
815
Provide Binary Representation as Intermediate Format
820
End
825

Figure 8A

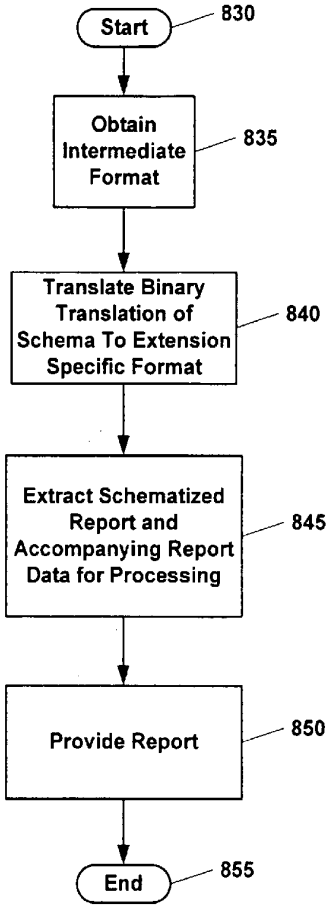
Start
830
Obtain Intermediate Format
835
Translate Binary Translation of Schema To Extension Specific Format
840
Extract Schematized Report and Accompanying Report Data for Processing
845
Provide Report
850
End
855

Figure 8B

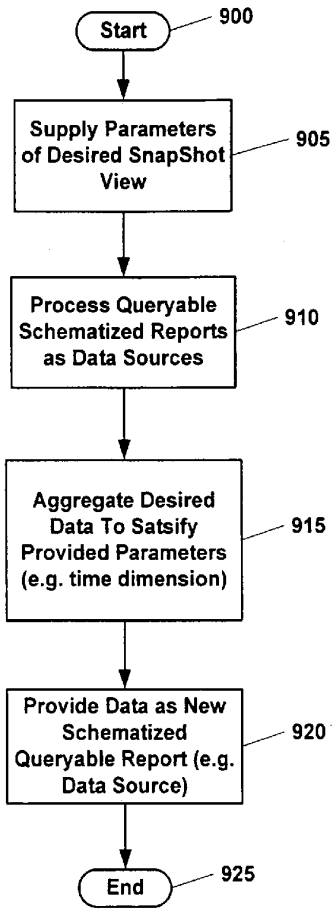
Start
900
Supply Parameters of Desired SnapShot View
905
Process Queryable Schematized Reports as Data Sources
910
Aggregate Desired Data To Satsify Provided Parameters (e.g. time dimension)
915
Provide Data as New Schematized Queryable Report (e.g. Data Source)
920
End
925

Figure 9