US 20100312748A1

(19) **United States**

(12) **Patent Application Publication**

Dickerman et al.

(10) **Pub. No.: US 2010/0312748 A1**

(43) **Pub. Date: Dec. 9, 2010**

(54) **MULTIDIMENSIONAL CUBE FUNCTIONS**

(75) Inventors: **Howard J. Dickerman**, Bellevue, WA (US); **Brian W. Boon**, Redmond, WA (US); **Charles D. Ellis**, Seattle, WA (US); **Kaicheng Hu**, Bellevue, WA (US); **Robert G. Hawking**, Seattle, WA (US)

Correspondence Address:
MERCHANT & GOULD (MICROSOFT)
P.O. BOX 2903
MINNEAPOLIS, MN 55402-0903 (US)

(73) Assignee: **MICROSOFT CORPORATION**, Redmond, WA (US)

(21) Appl. No.: **12/852,235**

(22) Filed: **Aug. 6, 2010**

Related U.S. Application Data

(63) Continuation of application No. 11/251,598, filed on Oct. 14, 2005, now Pat. No. 7,805,433.

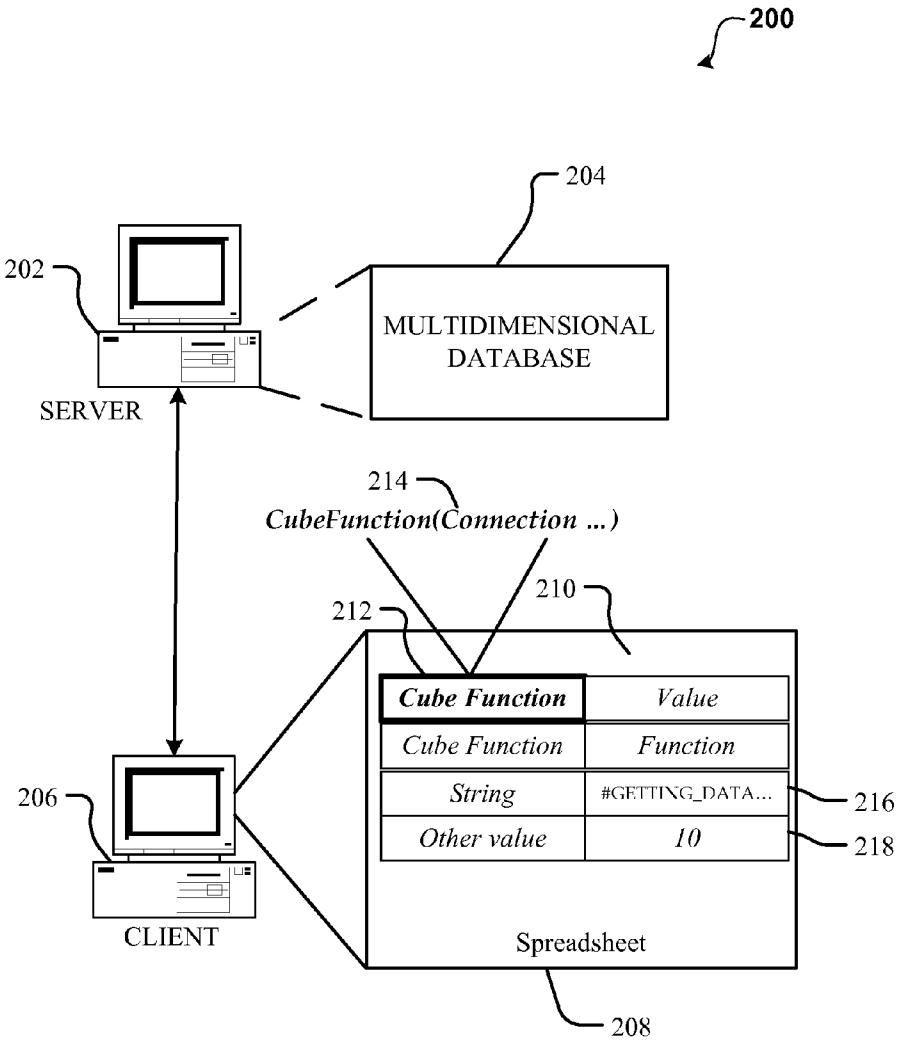
Publication Classification

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

(52) **U.S. Cl.** **707/602**; 707/E17.005

(57) **ABSTRACT**

Cube functions may be used to obtain data from a multidimensional database. The cube functions may be contained within one or more cells of a spreadsheet. These cube functions behave similarly to the standard functions that may be included within a spreadsheet. Exemplary cube functions include obtaining: a cube member, a cube value, a cube set, a ranked member, a KPI, a member property and a count relating to a set. The cube functions within the spreadsheet may access the cube data from one or more multidimensional databases. Using the cube formulas in individual cells allows the user to add/delete rows and/or columns from within the spreadsheet.



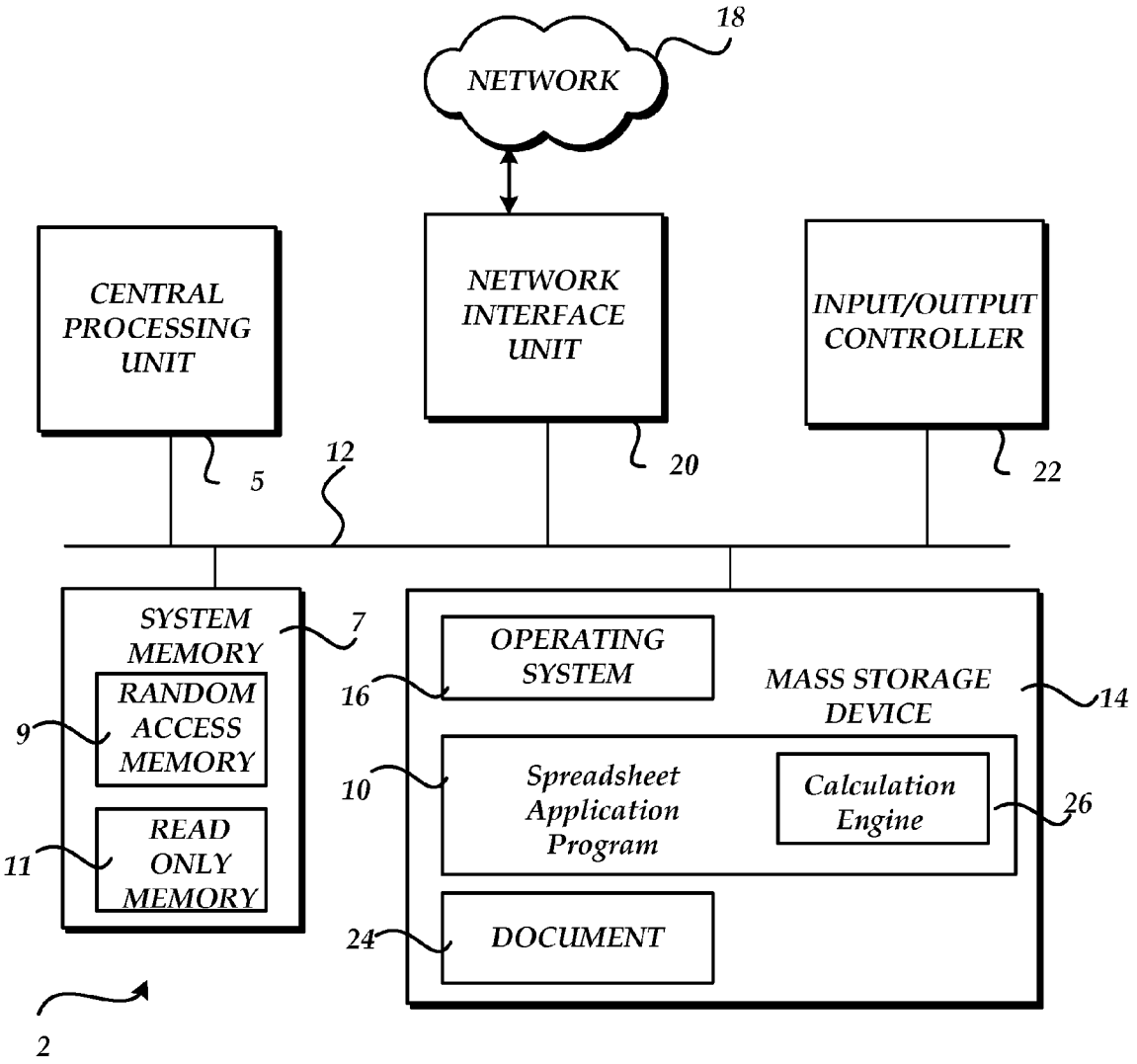

NETWORK
18
CENTRAL PROCESSING UNIT
NETWORK INTERFACE UNIT
INPUT/OUTPUT CONTROLLER
5
12
20
22
SYSTEM MEMORY
7
RANDOM ACCESS MEMORY
9
READ ONLY MEMORY
11
OPERATING SYSTEM
16
MASS STORAGE DEVICE
14
Spreadsheet Application Program
10
Calculation Engine
26
DOCUMENT
24
2

Fig. 1

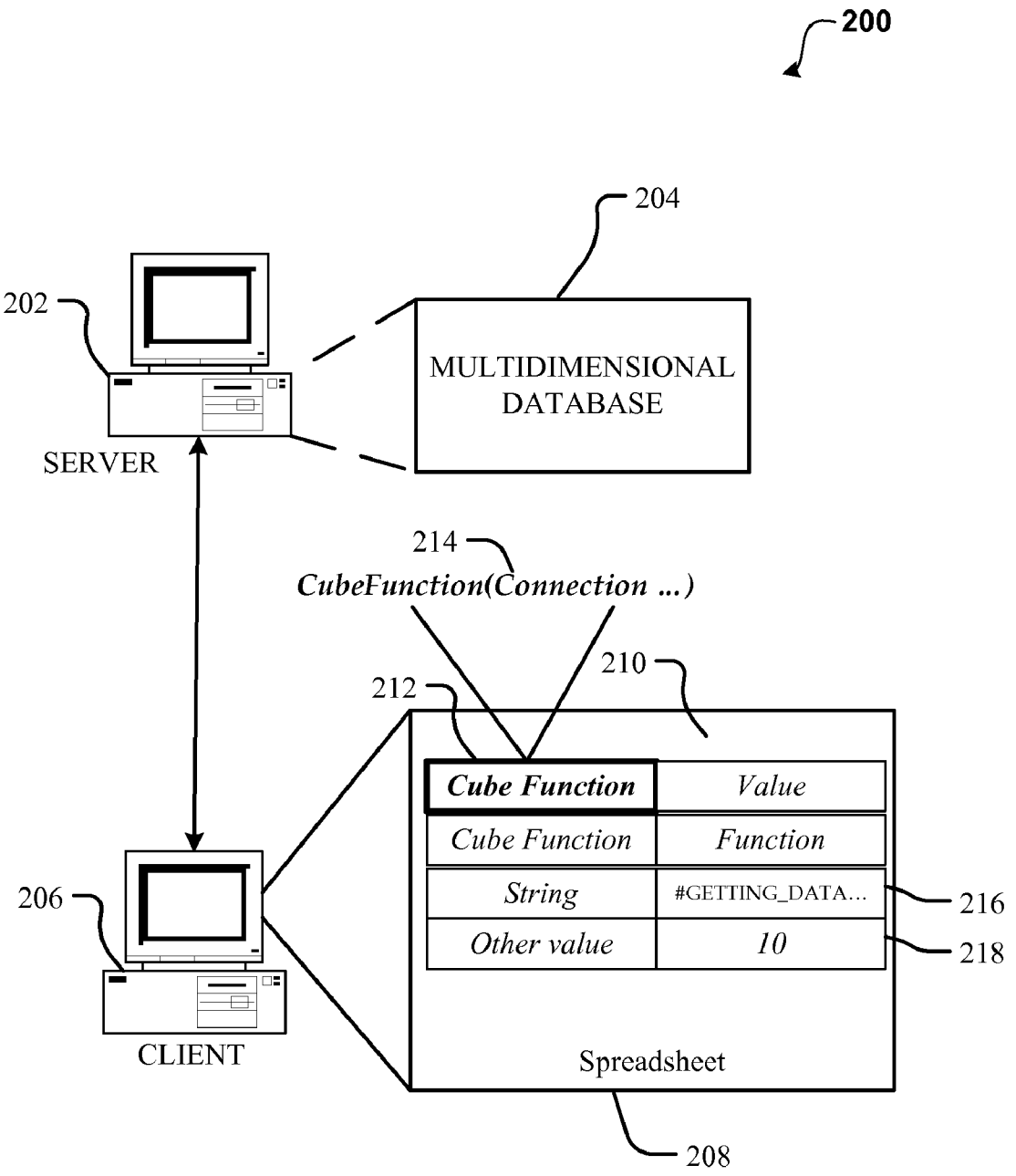

200
204
202
MULTIDIMENSIONAL DATABASE
SERVER
214
CubeFunction(Connection ...)
210
212
Cube Function
Value
Cube Function
Function
String
#GETTING_DATA...
Other value
10
216
218
206
CLIENT
Spreadsheet
208

Fig. 2

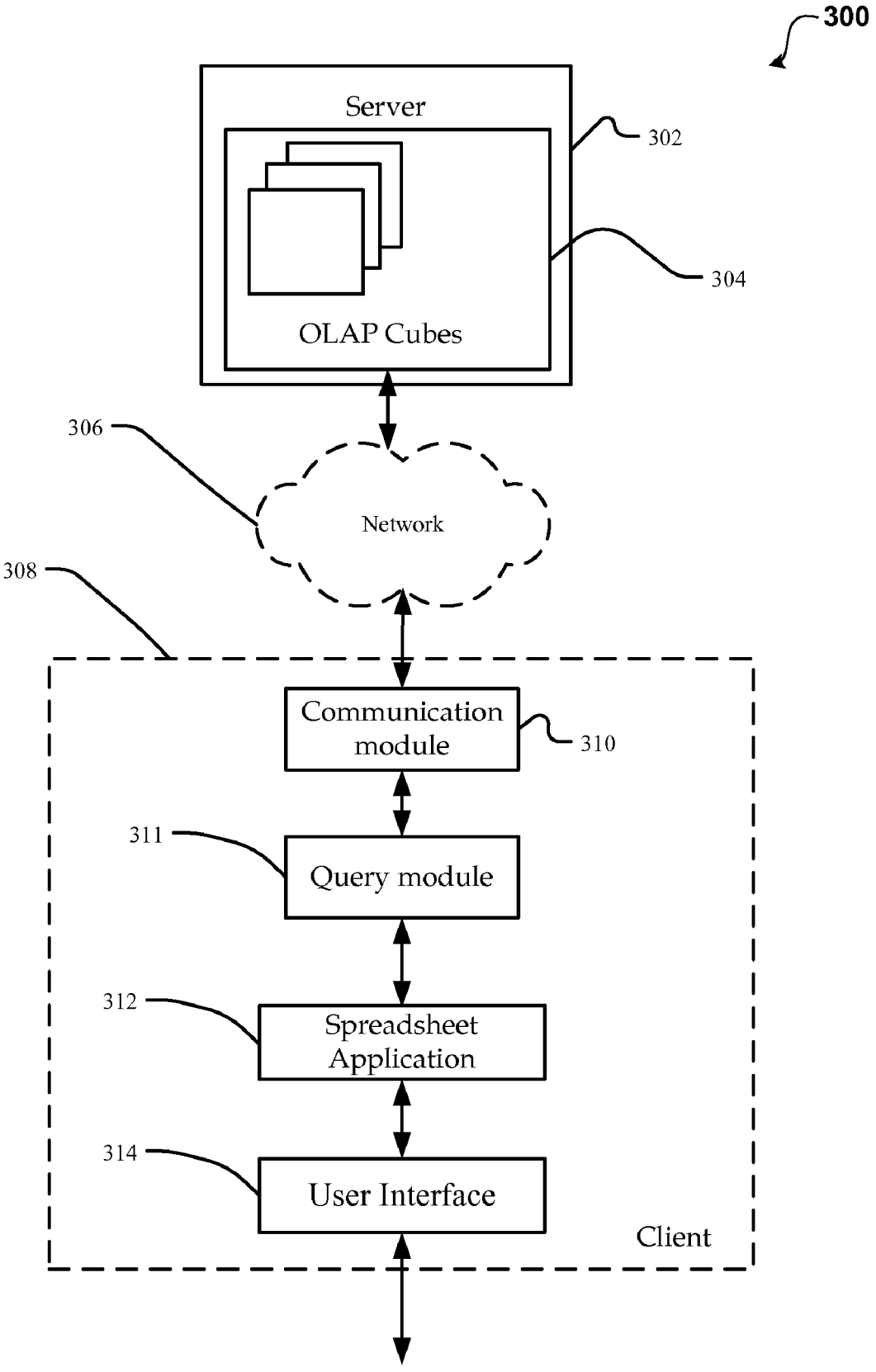

300
Server
302
304
OLAP Cubes
306
Network
308
Communication module
310
311
Query module
312
Spreadsheet Application
314
User Interface
Client

Fig. 3

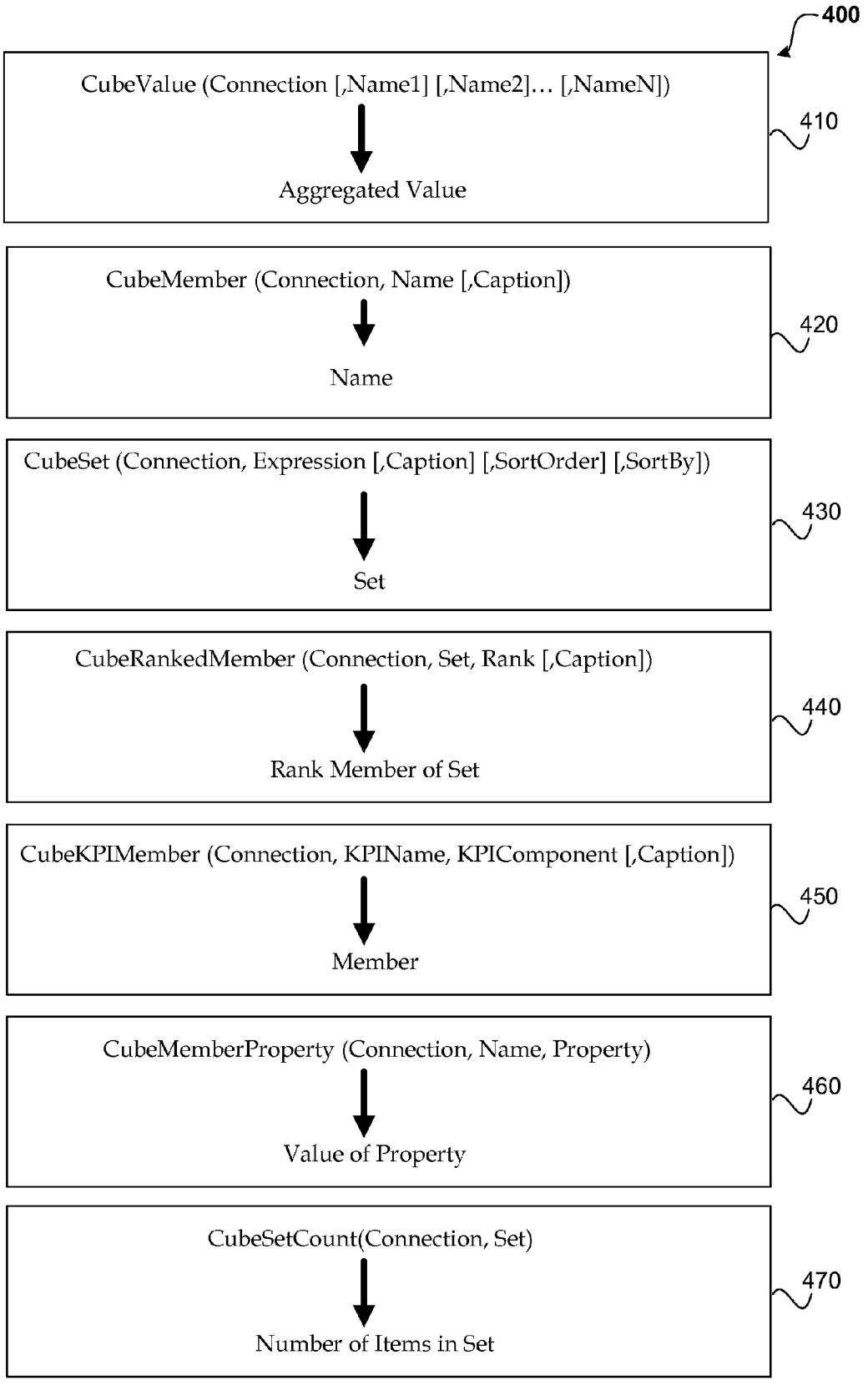

400
CubeValue (Connection [,Name1] [,Name2]… [,NameN])
Aggregated Value
410
CubeMember (Connection, Name [,Caption])
Name
420
CubeSet (Connection, Expression [,Caption] [,SortOrder] [,SortBy])
Set
430
CubeRankedMember (Connection, Set, Rank [,Caption])
Rank Member of Set
440
CubeKPIMember (Connection, KPIName, KPIComponent [,Caption])
Member
450
CubeMemberProperty (Connection, Name, Property)
Value of Property
460
CubeSetCount(Connection, Set)
Number of Items in Set
470

Fig. 4

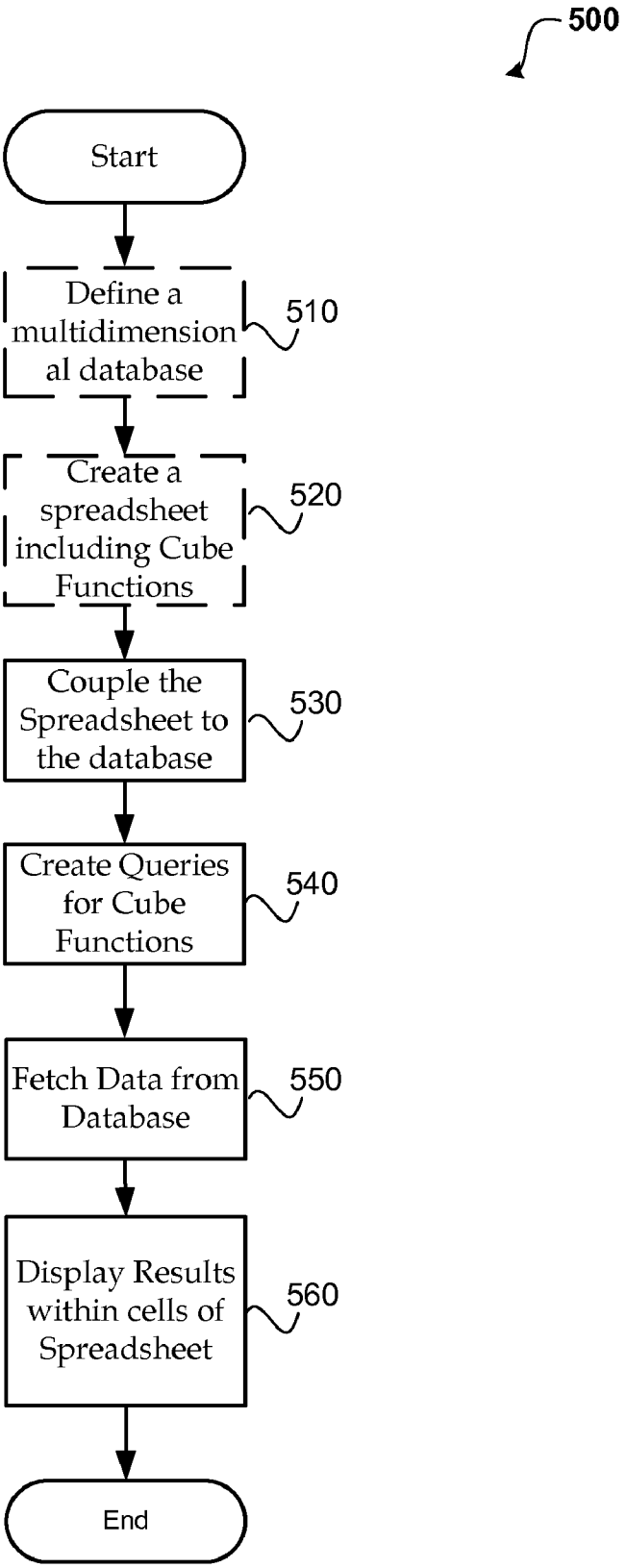

500
Start
Define a multidimensional database
510
Create a spreadsheet including Cube Functions
520
Couple the Spreadsheet to the database
530
Create Queries for Cube Functions
540
Fetch Data from Database
550
Display Results within cells of Spreadsheet
560
End

Fig. 5

600

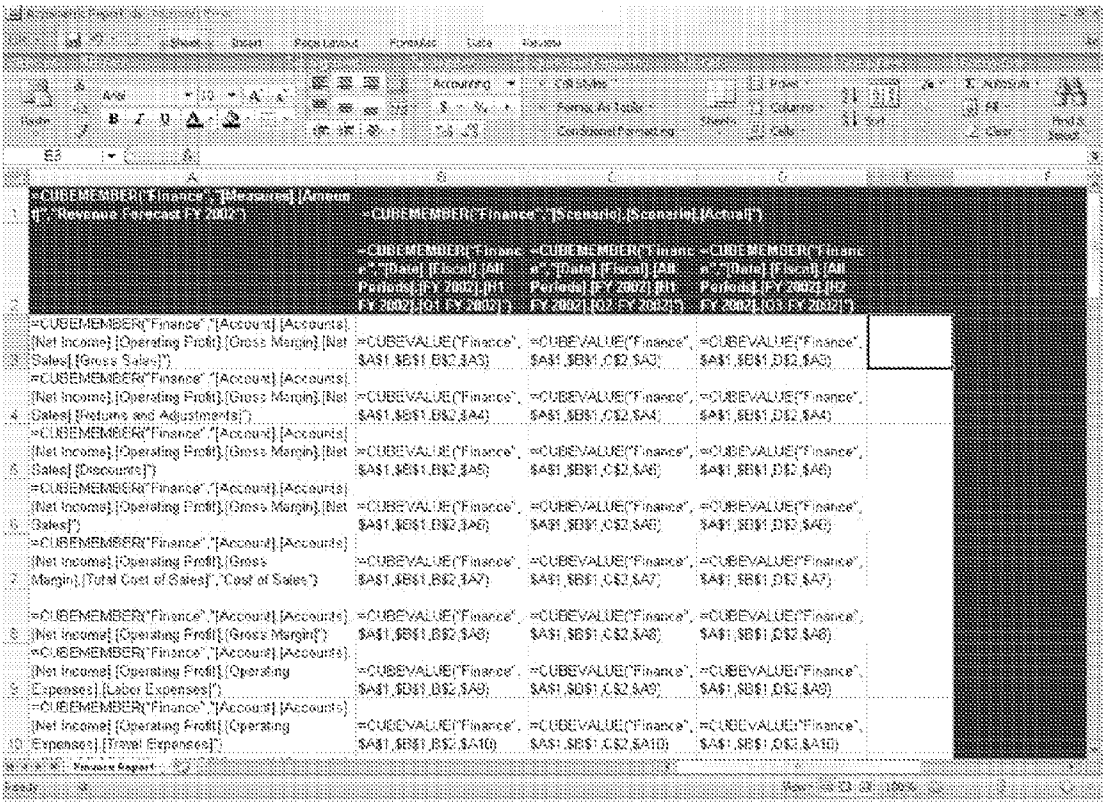

Revenue Report
Revenue Forecast FY 2002
=CUBEMEMBER("Finance","[Scenario].[Scenario].[Actual]")
=CUBEMEMBER("Finance","[Time].[Fiscal].[All Periods].[FY 2002].[Q1 FY 2002]")
=CUBEMEMBER("Finance","[Time].[Fiscal].[All Periods].[FY 2002].[Q2 FY 2002]")
=CUBEMEMBER("Finance","[Time].[Fiscal].[All Periods].[FY 2002].[Q3 FY 2002]")
=CUBEMEMBER("Finance","[Account].[Accounts].[Net Income].[Operating Profit].[Gross Margin].[Net Sales].[Gross Sales]")
=CUBEVALUE("Finance",A1,B1,B$2,$A3)
=CUBEVALUE("Finance",A1,B1,C$2,$A3)
=CUBEVALUE("Finance",A1,B1,D$2,$A3)
=CUBEMEMBER("Finance","[Account].[Accounts].[Net Income].[Operating Profit].[Gross Margin].[Net Sales].[Returns and Adjustments]")
=CUBEVALUE("Finance",A1,B1,B$2,$A4)
=CUBEVALUE("Finance",A1,B1,C$2,$A4)
=CUBEVALUE("Finance",A1,B1,D$2,$A4)
=CUBEMEMBER("Finance","[Account].[Accounts].[Net Income].[Operating Profit].[Gross Margin].[Net Sales].[Discounts]")
=CUBEVALUE("Finance",A1,B1,B$2,$A5)
=CUBEVALUE("Finance",A1,B1,C$2,$A5)
=CUBEVALUE("Finance",A1,B1,D$2,$A5)
=CUBEMEMBER("Finance","[Account].[Accounts].[Net Income].[Operating Profit].[Gross Margin].[Net Sales]")
=CUBEVALUE("Finance",A1,B1,B$2,$A6)
=CUBEVALUE("Finance",A1,B1,C$2,$A6)
=CUBEVALUE("Finance",A1,B1,D$2,$A6)
=CUBEMEMBER("Finance","[Account].[Accounts].[Net Income].[Operating Profit].[Gross Margin].[Total Cost of Sales]","Cost of Sales")
=CUBEVALUE("Finance",A1,B1,B$2,$A7)
=CUBEVALUE("Finance",A1,B1,C$2,$A7)
=CUBEVALUE("Finance",A1,B1,D$2,$A7)
=CUBEMEMBER("Finance","[Account].[Accounts].[Net Income].[Operating Profit].[Gross Margin]")
=CUBEVALUE("Finance",A1,B1,B$2,$A8)
=CUBEVALUE("Finance",A1,B1,C$2,$A8)
=CUBEVALUE("Finance",A1,B1,D$2,$A8)
=CUBEMEMBER("Finance","[Account].[Accounts].[Net Income].[Operating Profit].[Operating Expenses].[Labor Expenses]")
=CUBEVALUE("Finance",A1,B1,B$2,$A9)
=CUBEVALUE("Finance",A1,B1,C$2,$A9)
=CUBEVALUE("Finance",A1,B1,D$2,$A9)
=CUBEMEMBER("Finance","[Account].[Accounts].[Net Income].[Operating Profit].[Operating Expenses].[Travel Expenses]")
=CUBEVALUE("Finance",A1,B1,B$2,$A10)
=CUBEVALUE("Finance",A1,B1,C$2,$A10)
=CUBEVALUE("Finance",A1,B1,D$2,$A10)

Fig. 6

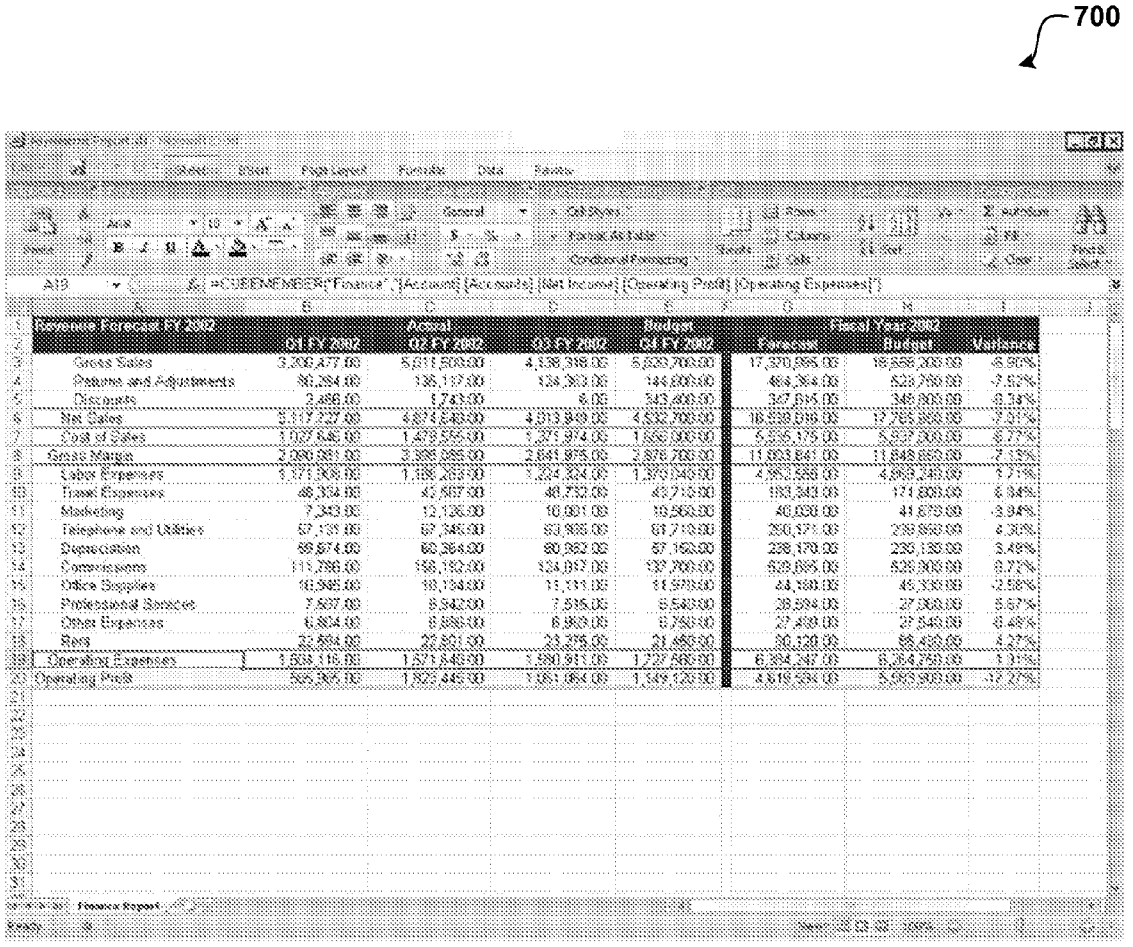


Fig. 7

MULTIDIMENSIONAL CUBE FUNCTIONS

BACKGROUND

[0001] Spreadsheet software applications are used by many different users for manipulating data. Typical spreadsheet applications simulate physical spreadsheets by capturing, displaying, and manipulating data arranged in rows and columns. In addition to using spreadsheet applications, many users also store and utilize enormous amounts of data stored in multidimensional databases. These multidimensional databases are also known as OLAP cubes. These OLAP cubes are architecturally different from relational databases or object oriented databases and the language used to query and describe elements within the OLAP cubes is the Multi-Dimensional eXpression (MDX) language. OLAP systems analyze data drawn from other databases, often large relational databases such as data warehouses, or other multidimensional databases. The purpose of such analysis is to aggregate and organize business information into a readily accessible, easy to use multidimensional structure. Placing this multidimensional data within a PivotTable report within a spreadsheet is limiting because PivotTable reports are restrictive in their layout.

SUMMARY

[0002] This Summary is provided to introduce a selection of concepts in a simplified form that are further described below in the Detailed Description. This Summary is not intended to identify key features or essential features of the claimed subject matter, nor is it intended to be used as an aid in determining the scope of the claimed subject matter.

[0003] Cube functions may be used to obtain data from a multidimensional database. The cube functions may be contained within one or more cells of a spreadsheet and/or may be accessed through an API. The cube functions behave similarly to the standard functions that may be included within cells of a spreadsheet. Exemplary cube functions include, obtaining: a cube member, a cube value, a cube set, a ranked member, a KPI, a member property and a count of items in a set. Data may be obtained from one or more multidimensional databases. The cube functions may be executed asynchronously such that a user may continue to interact with the spreadsheet while the requested data is being obtained from one or more of the multidimensional databases. The cube functions may take parameters that may include other spreadsheet cell references as well as other cube functions. The use of the cube functions enables the spreadsheet program to be used as a reporting tool for multidimensional (OLAP) data stores. Using the cube formulas in individual cells allows the user to add/delete rows and/or columns from within the spreadsheet.

BRIEF DESCRIPTION OF THE DRAWINGS

[0004] FIG. **1** illustrates an exemplary computing architecture for a computer;

[0005] FIG. **2** illustrates an overview of a spreadsheet system for accessing multidimensional data through the use of cube functions;

[0006] FIG. **3** illustrates a system for interfacing with OLAP cubes from cells of a spreadsheet application;

[0007] FIG. **4** illustrates cube functions that may be utilized to connect with a multidimensional database;

[0008] FIG. **5** displays an operational flow for using cube functions within a spreadsheet; and

[0009] FIGS. **6** and **7** show an exemplary spreadsheet that includes cube functions before evaluation and after evaluation, in accordance with aspects of the present invention.

DETAILED DESCRIPTION

[0010] Referring now to the drawings, in which like numerals represent like elements, various aspects of the present invention will be described. In particular, FIG. **1** and the corresponding discussion are intended to provide a brief, general description of a suitable computing environment in which embodiments of the invention may be implemented.

[0011] Generally, program modules include routines, programs, components, data structures, and other types of structures that perform particular tasks or implement particular abstract data types. Other computer system configurations may also be used, including hand-held devices, multiprocessor systems, microprocessor-based or programmable consumer electronics, minicomputers, mainframe computers, and the like. Distributed computing environments may also be used where tasks are performed by remote processing devices that are linked through a communications network. In a distributed computing environment, program modules may be located in both local and remote memory storage devices.

[0012] Throughout the specification and claims, the following terms take the meanings explicitly associated herein, unless the context clearly dictates otherwise. The term "MDX" refers to the MultiDimensional eXpressions language. The term "KPI" refers to a Key Performance Indicator. The term "MDX Name" is a name as defined by MDX. The MDX unique name of a member is generally in the form [Dimension].[Hierarchy].[Level].&[MemberKey] wherein all of these components are required. Non-unique names could have several other forms including [Member] or [Dimension].[Member] or [Dimension].&[98765]. The term "caption" refers to a non-unique friendly name to be displayed in the spreadsheet. The term "connection" refers to the name of a data connection that has been stored within a spreadsheet workbook. Connection names are strings that uniquely identify connections within the workbook in which they are used. The term "cube" refers to the multi-dimensional OLAP database from which data is retrieved. The term "member" is a value along one of the cube's dimensions. For example, a member of a Time dimension might be "June 2003". A member of a customers dimension might be "John Doe." The term "tuple" is the intersection of one or more members in a cube, with only one member from each dimension. The tuple represents the slice of the cube that includes the specified members. When a tuple contains only one member then that member and the tuple are identical to each other. The MDX Name for a tuple is of the form (<member1>, <member2> . . . <memberN>) where each <member> is replaced with the MDX name of that member. When the spreadsheet refers to a range of cells that contain members (or tuples) these ranges are interpreted as tuples. For example, if cells A10 thru A13 of a spreadsheet contain members, then the cube function=CubeValue ("MyConnection", $A10:$A13, D$2) has a tuple as its second argument. The term "set" is an ordered collection of one or more members (or tuples).

[0013] Briefly described, cube functions may be used to access data from a multidimensional database. According to embodiments, users can enter one or more cube functions into one or more of the spreadsheet cells, specify a multidimen-

sional database, query a selected database, and then return the data to be used within the cell(s) of the spreadsheet. According to one embodiment, the data may be an aggregated value, a dimension member, a KPI, or a member property. The use of these cube functions enables the spreadsheet program to be more easily used as a reporting tool for multidimensional OLAP data stores. Using the cube functions within the individual cells of the spreadsheet provides far greater layout and formatting flexibility than using other methods which enforce limitations on the layout of the spreadsheet.

[0014] Referring now to FIG. **1**, an exemplary computer architecture for a computer **2** utilized in various embodiments will be described. The computer architecture shown in FIG. **1** may be configured in many different ways. For example, the computer may be configured as a server, a personal computer, a mobile computer and the like. As shown, computer **2** includes a central processing unit **5** ("CPU"), a system memory **7**, including a random access memory **9** ("RAM") and a read-only memory ("ROM") **11**, and a system bus **12** that couples the memory to the CPU **5**. A basic input/output system containing the basic routines that help to transfer information between elements within the computer, such as during startup, is stored in the ROM **11**. The computer **2** further includes a mass storage device **14** for storing an operating system **16**, application programs, and other program modules, which will be described in greater detail below.

[0015] The mass storage device **14** is connected to the CPU **5** through a mass storage controller (not shown) connected to the bus **12**. The mass storage device **14** and its associated computer-readable media provide non-volatile storage for the computer **2**. Although the description of computer-readable media contained herein refers to a mass storage device, such as a hard disk or CD-ROM drive, the computer-readable media can be any available media that can be accessed by the computer **2**.

[0016] By way of example, and not limitation, computer-readable media may comprise computer storage media and communication media. Computer storage media includes 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, EPROM, EEPROM, flash memory or other solid state memory technology, CD-ROM, digital versatile disks ("DVD"), or other optical storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other medium which can be used to store the desired information and which can be accessed by the computer **2**.

[0017] According to various embodiments, the computer **2** operates in a networked environment using logical connections to remote computers through a network **18**, such as the Internet. The computer **2** may connect to the network **18** through a network interface unit **20** connected to the bus **12**. The network interface unit **20** may also be utilized to connect to other types of networks and remote computer systems.

[0018] The computer **2** may also include an input/output controller **22** for receiving and processing input from a number of devices, such as: a keyboard, mouse, electronic stylus and the like. Similarly, the input/output controller **22** may provide output to a display screen, a printer, or some other type of device (not shown).

[0019] As mentioned briefly above, a number of program modules and data files may be stored in the mass storage device **14** and RAM **9** of the computer **2**, including an operating system **16** suitable for controlling the operation of a networked computer, such as: the WINDOWS XP operating system from MICROSOFT CORPORATION; UNIX; LINUX and the like. The mass storage device **14** and RAM **9** may also store one or more program modules. In particular, the mass storage device **14** and the RAM **9** may store a spreadsheet application program **10**, such as the MICROSOFT® EXCEL spreadsheet application. According to one embodiment, the spreadsheet application **10** is operative to provide functionality for interacting with an OLAP data store through the use of cube functions that are entered into one or more cells of spreadsheet application **10**. The spreadsheet may generate one or more requests to fetch data from an OLAP cube identified by the cube function contained within the cell and populate a cell when there are cube functions contained within one or more cells of the spreadsheet.

[0020] The spreadsheet application **10** is configured to receive user input. For example, a user enters item data into a spreadsheet via a graphical user interface. The user input can be item data, item metadata, function information, cube function information, or other data. The user input may be direct input created by the user typing, pasting, or other deliberate action entering data into the spreadsheet or indirect input that may be generated by another program.

[0021] Calculation engine **26** performs operations relating to the cells within the spreadsheet. According to one embodiment, calculation engine **26** is a component within the spreadsheet application **10**. The calculation engine **26**, however, may be located externally from the spreadsheet application **10**. The operations performed by calculation engine **26** may be mathematical, such as summation, division, multiplication, etc., or may include other functions or features, such as interacting with a multidimensional data store. Calculation engine **26** may be configured to perform many different operations.

[0022] FIG. **2** illustrates an overview of a spreadsheet system **200** for accessing multidimensional data through the use of cube functions, in accordance with aspects of the invention. As illustrated, system **200** includes a server **202**, which includes and/or is coupled to a multidimensional database **204**, such as an OLAP data store. Server **202** is coupled to client **206** through a network connection. Client **206** includes a spreadsheet application **208**. Spreadsheet application **208** includes spreadsheet cells **210**. Typically, an active cell **212**, which is highlighted by the dark border, is the cell within the spreadsheet that is currently being acted upon. For example, the user is entering a cube function into the cell. Zero or more of the spreadsheet cells may contain a cube function which is directed at interacting with and retrieving data from the multidimensional database **204**. According to one embodiment, each cube function includes a Connection parameter **214** that identifies the multi-dimensional database to access. According to one embodiment, the user may select a UI element, such as a dropdown, to choose from a list of available connections to OLAP cubes. The selected value is then included within the cube function.

[0023] Communication between the spreadsheet application and the OLAP database **204** may be accomplished using MDX. Any other language, however, may be utilized that can communicate with an OLAP database. Furthermore, although the application is described herein as a spreadsheet, it will be appreciated that other applications, such as word processing applications that include spreadsheet cells, as well

as other applications utilizing cells, may utilize the cube functions described herein. According to another embodiment, the cube functions may be maintained separately from an application and may be accessed through an API.

[0024] Generally, a user may enter cube functions within one or more of the cells within spreadsheet **208** to fetch data from the OLAP database **204**. The queries to retrieve the data from the OLAP cube(s) that are sent may be MDX expressions. Each of the cube functions includes connection information that specifies the appropriate database.

[0025] The cells within the spreadsheet that contain cube functions appear to the user to behave as any other cell within the spreadsheet that contains a formula. Generally, formulas within the spreadsheet can include the following: cube functions as defined herein that directly query an OLAP cube; dependent cube functions that query an OLAP cube but that also require the results of a different query as one of their arguments; standard spreadsheet functions that have a dependency on the values returned by the cube functions; and standard spreadsheet functions that have no dependency.

[0026] If each of these cube functions is independently executed within the spreadsheet then there may be a large number of small queries against the OLAP server. This could result in diminished performance for the spreadsheet application. According to one embodiment, the queries are combined into fewer queries such that fewer queries are made to the OLAP server.

[0027] The spreadsheet cell calculations may be performed asynchronously. In other words, while data is being fetched from the OLAP server the calculations may continue within the other cells. Therefore, the calculations proceed for the cells that have no dependency on the result set, but are delayed for cells that are dependent. If the cell has no dependency on a query, the cell will get its value right away (**218**). If the cell has a dependency on a query, the cell is filled with a temporary error value of "#GETTING DATA . . . " (**216**) and the calculation proceeds to the next cell in the chain. This error shows the user that an action is being performed that relates to the cell.

[0028] When all of the cells have been evaluated, the spreadsheet triggers the aggregated query(s) needed to obtain data. This query is run asynchronously whenever possible. Asynchronous query processing may be desired so that the query won't block the spreadsheet applications UI thread and users can continue to work with the UI and can even abort the query when it's taking too long. As the values arrive for the cells that display the #GETTING_DATA . . . **216** error message, the error message is replaced with the fetched external data values and the calc is triggered for the cells that were dependent on the value that came in.

[0029] FIG. **3** illustrates a system **300** for interfacing with OLAP cubes from cells of a spreadsheet application, in accordance with aspects of the invention. System **300** in this embodiment includes a server **302** which correspondingly has one or more databases stored thereupon **304**. System **300** may optionally include a network **306** such as a LAN, WAN, the Internet or other network which server **302** may be coupled to.

[0030] System **300** includes client **308**. Client **308** includes a communication module **310** that is coupled to a spreadsheet application **312**. Furthermore, communication module **310** is coupled to the network **306**. Communication module **310** may also be directly coupled to server **302** and/or directly to OLAP cube(s) **304**.

[0031] When a user configures a new spreadsheet in spreadsheet application **312**, they may chose from a list of OLAP cubes **304** to which it may connect. Alternatively, the user may type in the location of an OLAP cube to be connected. This link may then be given a connection name, such that this name is used by a query module **311** to construct a query for the named OLAP cube. Query module **311** is configured to receive the cube function(s) from one or more cells within the spreadsheet application and then construct an MDX query that will be passed on to server **302** (via communication module **310**) to be interpreted. The appropriate cube **304** is then queried and, in response to the query, returns data from the database relating to the query to communication module **310**. Communication module **310** then passes the data to the spreadsheet application **312**, which in turn fills in the cell(s) with the data. When other cells within the spreadsheet depend upon the returned data, those cells may then be updated.

[0032] Communication module **310** may be located on client **308**, however it may also be included on server **302** or may be included in cube(s) **304**, among other locations. Communication module **310** is typically provided by cube(s) **304** such that the client **308** and spreadsheet application **312** may communicate with the cube(s) **304**. In one embodiment, communication module **310** may comprise a dynamic-link library (DLL) that is provided (and configured) by the particular linked cube.

[0033] While query module **311** is shown as being separate from spreadsheet application **312**, it may be included within the spreadsheet application **312**. The location of query module **311** may also be other than in the client **308**, such as within the server **302**, or at some remote location.

[0034] FIG. **4** illustrates cube functions that may be utilized to connect with a multidimensional database, in accordance with aspects of the invention.

[0035] As illustrated, FIG. **4** includes the following cube functions: cube value function **410**; cube member function **420**; cube set function **430**; cube ranked member **440**; cube KPI member **450**; cube member property **460**; and cube set count **470**.

[0036] CubeValue (Connection [,Name1] [,Name2] . . . [,NameN]) function **410** returns an aggregated value from the OLAP cube specified by the connection parameter. The "Name" parameter is an optional parameter that may appear any number of times within the CubeValue function call. For example, one cube value function call may contain only one name, while another cube value function call may contain many names, separated by commas. According to one embodiment, each name is the MDX name of a member or tuple (or tuple) within the cube. Alternatively, the name could be a Set as defined by a CubeSet function as described below. The provided names are used as slicers to define the portion of the cube for which an aggregated value is to be returned. According to one embodiment, if a Name is not supplied for any particular hierarchy, then the default member for that hierarchy is used. Similarly, when a Name is not supplied for any measure, then the default measure for the cube is used. The Name parameter may also be a cell reference to another cell in the spreadsheet that contains a unique MDX name in the cell's MDX property. Any cells that contain the CubeMember( ) function evaluate to a caption plus a unique MDX name in the cell's MDX property.

[0037] According to one embodiment, the tuple associated with any CubeValue is stored. This increases the efficiency to communicate with the OLAP server about the portion of the

cube that's associated with the desired value. The following are some illustrative examples of the CubeValue( ) function: CubeValue ("SALES", "[Measures].[Profit]", "[Time].[1999]", "[All Product].[Beverages]"); CubeValue (A1, "[Measures].[Profit]", D$12, $A23); and CubeValue ("SALES", B7, D$12, $A23).

[0038] The CubeMember (Connection, Name [,Caption]) function **420** returns a fully qualified unique MDX member name, or MDX tuple, as well as the user-friendly caption for that member. According to one embodiment, when a tuple is returned, the user-friendly caption is the caption of the last member in the tuple. Evaluating this function causes the cell to display the caption, while the cell's MDX property is set to the unique MDX member name or MDX expression for the tuple. The CubeMember function returns an error (#N/A) when the specified Name is not found within the specified cube. The Name parameter is a string containing the MDX member name of a member within the cube. The CubeMember function validates that the provided name exists within the cube and then returns the unique member name associated with this member as the MDX property along with that member's caption as the function's value. This argument may also represent a tuple. When the argument is a tuple, the user specifies an array of members using the spreadsheet's array syntax. The name argument may also be a reference to a cell that contains a member, or a cell that contains a string which evaluates to a unique MDX name, or a range of cells which will evaluate to a tuple. The Caption parameter is an optional string to be used as the caption in place of any caption that we obtain from the cube.

[0039] The following are some illustrative examples of the CubeMember( ) function: CubeMember ("conn", "[Customers].[USA]"); and CubeMember("Finance",[Account].[Accounts].[NetIncome].[Operating Profit].[Gross Margin].[Net Sales].[Gross Sales]).

[0040] The CubeSet (Connection, Expression [,Caption] [,SortOrder] [,SortBy]) function **430** defines a set that is to be built for the spreadsheet by the OLAP server. The CubeSet function sends the Expression to the OLAP server defined by the specified Connection. The server then creates a set by evaluating the Expression and returns that set to the spreadsheet program. The spreadsheet then uses the Caption as the function's display value, while also setting the function's MDX (value) metadata to the MDX Expression that was sent to the OLAP server.

[0041] The Expression parameter is a string containing any arbitrary MDX expression that can be evaluated by the OLAP server which results in a set of members or a set of tuples. An expression can also be a cell reference to a range that contains one or more cube members or tuples or sets. The Expression argument defines the items that are to be included in the set.

[0042] The Caption parameter is the optional string to be used as the caption that describes the set being created. When the caption is not provided, then the caption is set to a null string.

[0043] The SortOrder parameter is an optional integer argument whose value, according to one embodiment, is one of the following:

0 = SORT_NONE	Leaves the Set in it's existing order
1 = SORT_ASCENDING	Sorts Set in ascending order by SortBy field
2 = SORT_DESCENDING	Sorts Set in descending order by SortBy field
3 = SORT_ALPHA_ASCENDING	Sorts Set in Alpha ascending order
4 = SORT_ALPHA_DESCENDING	Sorts Set in Alpha descending order
5 = SORT_NATURAL_ASCENDING	Sorts Set in Natural ascending order
6 = SORT_NATURAL_DESCENDING	Sorts Set in Natural descending order

[0044] The SortBy parameter is an optional expression string that defines the values on which sorting is to be done. For example, if a user would want to obtain the city with the highest sales, then the Expression parameter defines a set of cities, and the SortBy parameter is the sales measure. If the user wants to obtain the city with the highest population, then the Expression parameter defines a set of cities, and the SortBy parameter is the population measure. The CubeSet function results in a set containing the items defined by the Expression parameter in the specified SortOrder by SortBy.

[0045] The following are some illustrative examples of the CubeSet( ) function: CubeSet("Finance", "Order([Product].[Product].[Product Category].Members, [Measures].[Unit Sales], ASC)", "Products"); and CubeSet("Sales", "[Product].[All Products].Children", "Products", 1, "[Measures].[Sales Amount]").

[0046] The CubeRankedMember (Connection, Set, Rank [,Caption]) function **440** returns the Nth (Rank) cube member (or tuple) in a Set, with the Caption as the display value, and a unique MDX name as the cell's MDX property. The Set is an expression that evaluates to a set within an OLAP cube. In order to obtain this set, the CubeSet( ) function may be used. This means that Set may be a reference to a cell that contains a CubeSet( ) function. In some OLAP servers, such as MICROSOFT CORPORATION'S Analysis Services a set may be defined by an expression within curly braces: such as "{[Item1].children}". The CubeRankedMember function returns one of the elements within the Set. The Rank parameter is an integer that determines the element within the set to return. When the Rank parameter is set to one then the CubeRanked Member function returns the first value. When the Rank parameter equals two then the second value is returned, and so on. For example, calling this function five different times with the Rank parameter set to 1 thru 5 is one way to obtain the "Top 5" items from an OLAP cube. The Caption parameter is an optional string to be used as the caption in place of the member caption as obtained from the OLAP cube.

[0047] The following are some illustrative examples of the CubeRankedMember( ) function: CubeRankedMember (D4, 1, "Top Month"); and CubeRankedMember (CubeSet ("SALES", "Summer", "[1999].[June]", "[1999].[July]", "[1999].[August]"), 1, "Top Month").

[0048] The CubeKPIMember (Connection, KPIName, KPIComponent [,Caption]) function **450** returns the KPI specified by the KPI name and the KPI Component. According to one embodiment, this KPI includes a fully qualified unique MDX expression as well as the user-friendly caption for that KPI. Evaluating CubeKPIMember function causes the cell's value within the spreadsheet to be set to the caption, while the cell's MDX property is set to the unique MDX expression for the requested KPI. The CubeKPIMember

function returns an empty string caption and has a NULL MDX property when the specified KPIName is not found within the specified OLAP cube. The KPIName is the name of the KPI in the cube that is to be queried. The KPIComponent is one of the following values: 1—KPIValue; 2—KPIGoal; 3—KPIStatus; 4—KPITrend; 5—KPIWeight; and 6—KPICurrentTimeMember. These are the components (properties) that make up a KPI. This Caption is an optional string to be used as the caption in place of any caption that is obtained from the cube. To obtain the value of a KPI member with additional slicers as appropriate, the user may use the CubeValue function, and references this KPI just as if it were a member of any hierarchy in the cube.

[0049] The following are some illustrative examples of the CubeKPIMember( ) function: CubeKPIMember ("Sales", "MySalesKPI", 1); and CubeKPIMember ("Sales", "MySalesKPI", **2**, "Sales KPI Goal").

[0050] The CubeMemberProperty (Connection, Name, Property) function **460** returns the value of a member property in the cube. The Name is the MDX unique name of a member within the cube. The CubeMemberProperty function validates that the name exists within the cube and then returns the specified property for this member. The Property is the MDX name of the property to be returned or a reference to a cell that contains the name of the property.

[0051] The following are some illustrative examples of the CubeMemberProperty( ) function: CubeMemberProperty ("Sales", "[Time].[Fiscal].[1999]", A3); and CubeMemberProperty ("Sales", "[Store].[MyFavoriteStore]", "[Store].[Store Name].[Store Sqft]").

[0052] The CubeSetCount(Connection, Set) function **470** returns the number of items in the Set. The Set is an expression that evaluates to a set as defined by the CubeSet function. In many cases, the cell will contain a CubeSet function. The following are some illustrative examples of the CubeSetCount( ) function: CubeSetCount (A3) and CubeSetCount (CubeSet (connection1, expression1, caption)).

[0053] When the Cube function is CubeSet or CubeSetCount, the MDX property is the MDX expression that is sent to the server for evaluation. It is the same as the Expression argument for the CubeSet function. For example: Cell C3 contains: =CubeSet("conn", "[Customers].children"); Cell C4 contains: =CubeSetCount(C3); Range ("C3").MDX="[Customers].children" and Range ("C4").MDX="[Customers]. children."

[0054] When the Cube function is CubeMember, CubeRankedMember, CubeValue, CubeKPIMember or CubeMemberProperty, then the MDX property provides a tuple as returned from the server representing the unique name of the member or tuple defined by the function. For example: Cell C3 contains: =CubeMember ("conn", "[Customers].[USA]"); Cell C4 contains: =CubeValue("conn", "[Bicycles]", "[WA]"); Range ("C3").MDX=(<unique member name for USA>) as returned by server; and Range ("C4").MDX=(<tuple containing unique member names for Bicycles and WA>).

[0055] When reading the discussion of the routines presented herein, it should be appreciated that the logical operations of various embodiments are implemented (1) as a sequence of computer implemented acts or program modules running on a computing system and/or (2) as interconnected machine logic circuits or circuit modules within the computing system. The implementation is a matter of choice dependent on the performance requirements of the computing system implementing the invention. Accordingly, the logical operations illustrated and making up the embodiments of the described herein are referred to variously as operations, structural devices, acts or modules. These operations, structural devices, acts and modules may be implemented in software, in firmware, in special purpose digital logic, and any combination thereof.

[0056] FIG. **5** displays an operational flow **500** for using cube functions within a spreadsheet, in accordance with aspects of the invention. After a start block, the process flows to define operation **510**, which defines a multidimensional database. A database is defined or created when the particular coordinates are defined and saved, and the data fields populated. Typically, the multidimensional databases are OLAP databases that have already been created and populated. In this case, the selected databases are coupled to the spreadsheet at operation **530**.

[0057] Moving to operation **520**, a spreadsheet is created that includes cube functions. Generally, a user may create a spreadsheet from scratch in which all of the cells within the spreadsheet are defined, but, typically, a user may start with a spreadsheet that has at least been partially created. According to one embodiment of the invention, a cell may include zero or more of the following cube functions: a cube member function; a cube value function; a cube set function; a cube ranked member function; a cubed KPI member function; a cube member property function; and a cube set count function as described more fully herein.

[0058] Flowing to operation **530**, the spreadsheet is coupled to the database. When the spreadsheet is created and particular cells are defined within the spreadsheet to include cube functions, different databases may be linked to the information in that cell and/or spreadsheet. In this way, each cell containing a cube function may fetch data from the appropriate OLAP data store.

[0059] Transitioning to operation **540**, queries are created for the cube functions within the cells of the spreadsheet. According to one embodiment, the queries are in the form of MDX queries.

[0060] Moving to operation **550**, the queries are used to fetch the data relating to the cube functions from one or more multidimensional data stores. The cells within the spreadsheet may then be populated with the fetched data.

[0061] At operation **560**, the results of the queries and any calculations that were performed may be displayed to the user. The process then moves to an end block and returns to processing other actions.

[0062] FIGS. **6** and **7** show an exemplary spreadsheet that includes cube functions before evaluation and after evaluation, in accordance with aspects of the invention. FIG. **6** shows the cube function text within the cells. FIG. **7** shows the result of calculating the cube functions.

[0063] As illustrated, each cell within spreadsheet **600** and **700** includes a cube function. Including a cube function within each cell of the spreadsheet is merely for illustrative purposes. The spreadsheet may have zero or more cube functions contained within the cells. For example, a spreadsheet may include only a single cube function to obtain a single value from an OLAP cube.

[0064] Referring to a few of the cells within spreadsheet **600** it can be seen that cell A3 includes the cube member function that connects to the finance database and includes an MDX expression "[Account].[Accounts].[Net Income].[Operating Profit].[Gross Margin].[Net Sales].[Gross Sales]." As

can be seen referring to spreadsheet **700** in FIG. **7**, cell A3 is displayed as "Gross Sales." Cell B3 includes a cube value function that obtains the values from the A1, B1, B$2, and $A3 that results in an aggregated value of 3,200,477.00 as displayed in cell B3 of spreadsheet **700**. As illustrated each cube function has connection information to the "Finance" OLAP cube. As discussed above, however, each spreadsheet may have zero or more cube connections. For example, one cell could include a cube function to obtain data from a first OLAP cube and another cell could include a cube function to obtain data from a second store.

[0065] Using cube functions within the spreadsheet allows the users to insert and/or delete rows and columns without interfering with the data's ability to be refreshed.

[0066] The above specification, examples and data provide a complete description of the manufacture and use of the composition of the invention. Since many embodiments of the invention can be made without departing from the spirit and scope of the invention, the invention resides in the claims hereinafter appended.

1-26. (canceled)

27. A computer-readable storage medium having computer-executable instructions for interacting with an OLAP cube, comprising:

parsing a cube function having parameters; wherein the cube function is included within a cell of a spreadsheet and wherein the cube function is directed at obtaining data from an OLAP cube; wherein the cube function is from cube functions comprising: a cube member function that identifies an item in the OLAP cube and returns a name for the member; a cube value function that identifies an aggregated value from the OLAP cube; a cube set function that defines a set that is to be built for the spreadsheet by an OLAP server storing the OLAP cube; and a member property function that identifies a value of a member property in the OLAP cube and a count relating to a set within the OLAP cube;

determining an OLAP cube from which to obtain data from one of the parameters of the cube function; and

fetching the data from the determined OLAP cube in response to the parameters of the cube function.

28. The computer-readable storage medium of claim **27**, wherein determining the OLAP cube from which to obtain data comprises locating a connection name that determines the OLAP cube to obtain data from.

29. The computer-readable storage medium of claim **27**, wherein parsing the cube function comprises parsing the parameters to locate one or more names, wherein the names identify at least one of a member and a tuple within the OLAP cube; and wherein each of the located names is used a slicer to define a portion of the OLAP cube for which an aggregated value is to be fetched from the OLAP cube.

30. The computer-readable storage medium of claim **27**, wherein parsing the cube function comprises parsing the parameters to locate a name that identifies a member of the OLAP cube and using the name to fetch at least one of a unique MDX member name, a MDX tuple, and a caption for the member of the OLAP cube.

31. The computer-readable storage medium of claim **27**, wherein parsing the cube function comprises parsing the parameters to locate an expression that identifies a set that is at least one of a set of members within the OLAP cube and a set of tuples within the OLAP cube; and then using the query to fetch the set from the OLAP cube; and when a sort parameter exists within the parameters; using the sort parameter to sort the set.

32. The computer-readable storage medium of claim **27**, wherein parsing the cube function comprises parsing the parameters to locate a set and a rank; wherein the rank is used to fetch at least one of the elements from the set.

33. The computer-readable storage medium of claim **27**, wherein parsing the cube function comprises parsing the parameters to locate a KPI name and a KPI component; wherein the KPI name and the KPI component are used to fetch a member from the OLAP cube.

34. The computer-readable storage medium of claim **27**, wherein parsing the cube function comprises parsing the parameters to locate a set and wherein a number of the members within the set is determined.

35. The computer-readable storage medium of claim **27**, wherein parsing the cube function comprises parsing the parameters to locate a name and a member; wherein the name and member are used to obtain a value of a member property in the OLAP cube.

36. A system for interacting with a multidimensional database from a spreadsheet, comprising:

a processor and a computer-readable storage medium; and

a spreadsheet application that is coupled to a network and is configured to perform steps, comprising:

parsing a cube function having parameters that is located within a cell of the spreadsheet to generate an MDX query; wherein the MDX query locates data within a multidimensional database; wherein the cube function is from cube functions comprising a cube member function that identifies an item in the multidimensional database and returns a name for the member; a cube value function that identifies an aggregated value from the multidimensional database; a cube set function that defines a set that is to be built for the spreadsheet by an OLAP server storing the multidimensional database; and a member property function that identifies a value of a member property in the multidimensional database and a count relating to a set within the multidimensional database;

determining the multidimensional database to query based on at least one of the parameters of the cube function;

querying a server identified by the determined multidimensional database using the MDX query;

receiving data from the server that is returned in response to the MDX query; and

updating the cell and any other dependent cells in response to the received data.

37. The system of claim **36**, wherein the MDX query requests at least one of the following from the multidimensional database: a ranked member; a KPI member; a member property of an item within the multidimensional database and a count relating to a set within the multidimensional database.

38. The system of claim **37**, wherein different cells of the spreadsheet request data from different multidimensional databases.

39. The system of claim **37**, wherein parsing the cube function comprises parsing the parameters to locate names, wherein the names identify a member and a tuple within the OLAP cube; and wherein each of the located names is used a slicer to define a portion of the OLAP cube for which an aggregated value is to be fetched from the OLAP cube.

40. A method for interacting with an OLAP cube, comprising:

parsing a cube function having parameters; wherein the cube function is included within a cell of a spreadsheet and wherein the cube function is directed at obtaining data from an OLAP cube; wherein the spreadsheet includes cells that do not include data from a multidimensional database;

determining an OLAP cube from which to obtain the data by accessing one of the parameters of the cube function; and

fetching the data from the determined OLAP cube in response to the parameters of the cube function.

41. The method of claim **40**, wherein parsing the cube function comprises parsing the parameters to locate names that identify at least one of a member and a tuple within the OLAP cube; and wherein each of the located names is used a slicer to define a portion of the OLAP cube for which an aggregated value is to be fetched from the OLAP cube.

42. The method of claim **40**, wherein parsing the cube function comprises parsing the parameters to locate a name that is used to fetch at least one of a unique MDX member name, a MDX tuple, and a caption for the member of the OLAP cube.

43. The method of claim **40**, wherein parsing the cube function comprises parsing the parameters to locate an expression that identifies a set that is at least one of a set of members within the OLAP cube and a set of tuples within the OLAP cube; and then using the query to fetch the set from the OLAP cube; and when a sort parameter exists within the parameters; using the sort parameter to sort the set.

44. The method of claim **40**, wherein parsing the cube function comprises parsing the parameters to locate a set and a rank; wherein the rank is used to fetch at least one of the elements from the set.

45. The method of claim **40**, wherein parsing the cube function comprises parsing the parameters to locate a set and wherein a number of the members within the set is determined.

46. The method of claim **40**, wherein parsing the cube function comprises parsing the parameters to locate a name and a member; wherein the name and member are used to obtain a value of a member property in the OLAP cube.

* * * * *