Title: A SYSTEM, METHOD AND ARTICLE OF MANUFACTURE FOR ORGANIZING MULTIPLE DIMENSIONS OF TAX DATA FOR THE PURPOSE OF GENERATING TAX-RELATED FACT DETAILS

Abstract: A system, method, and article of manufacture are provided for organizing multiple dimensions of tax-related data for generating fact details to facilitate payment of taxes. First, a plurality of dimensions of tax-related data are provided along with a plurality of attributes for each of the dimensions. Next, a plurality of entries are received which are associated with the attributes of the dimensions. A plurality of fact details are then generated by aggregating the entries of a predetermined set of the attributes of the dimensions. Thereafter, the fact details are outputted.
A SYSTEM, METHOD AND ARTICLE OF MANUFACTURE FOR ORGANIZING
MULTIPLE DIMENSIONS OF TAX DATA FOR THE PURPOSE OF GENERATING
TAX-RELATED FACT DETAILS

FIELD OF INVENTION

The present invention relates to tax-related software and more particularly to managing and
organizing tax-related details in a database for effective input and retrieval.

BACKGROUND OF INVENTION

Numerous systems have been developed to automate financial record keeping for the purpose of
tracking expense related items, income, assets, and/or liabilities, for budgetary or tax purposes.
Nevertheless, although attractive in theory, most conventional systems are impractical for
effective use. In most cases, one cannot afford to hire a bookkeeper to gather the necessary data,
and even though a number of commercial accounting programs are available which can be run
by an individual with no special training on a personal home computer, most individuals do not
have the time or desire to run such programs, which inevitably require the individual to manually
enter the necessary data into the computer each time a transaction, i.e., the writing of a check or
deposit slip, is carried out, or to scan prerecorded forms, which must be specially prepared and
compared with the original checks or deposit slips to ensure accuracy.

Numerous entities have finances which would benefit from automation, especially for budget
planning or tax purposes, but the automation is useless if it requires generation of additional
forms, manual verification and/or manual data entry. The simplest investments, such as a rental
property or the purchase of ordinary common stocks, can generate an overwhelming amount of
transaction data. Nevertheless, even the most user-friendly home accounting and tax preparation programs tend to end up on the shelf when the user discovers the effort needed to enter transaction data every time a cancelled check is returned or a deposit is made. Even where the bank makes available electronic or optically scannable records which can be entered in the home accounting program, the time necessary to verify the data often negates the time saved by automated entry. Despite the best of intentions, the cancelled checks and other documents concerning the transaction thus often end up in a "shoe box" until they are needed, for example, for taxes, or for an unforeseen crisis such as a divorce or bankruptcy, by which time the individual's records can at best be only partially reconstructed and at great difficulty. While such a possibility should motivate individuals to keep better records, especially in view of the currently available technology for that purpose, it simply does not.
SUMMARY OF INVENTION

A system, method, and article of manufacture are provided for organizing multiple dimensions of tax-related data for generating fact details to facilitate payment of taxes. First, a plurality of dimensions of tax-related data are provided along with a plurality of attributes for each of the dimensions. Next, a plurality of entries are received which are associated with the attributes of the dimensions. A plurality of fact details are then generated by aggregating the entries of a predetermined set of the attributes of the dimensions. Thereafter, the fact details are outputted.

In one aspect of the present invention, a description may be provided for each of the attributes, dimensions, and fact details. Such description may facilitate the inputting of the entries or serve to explain the generated fact details.

In another aspect of the present invention, the dimensions may include asset, date, entity, jurisdiction, tax classification, and/or transaction dimensions. Further, the fact details may include an adjusted book fact, a tax adjustment fact, a schedule M fact, a tax return fact, an apportionment fact, a line item fact, a transaction fact, an invoice fact, an asset fact, a payroll fact, a taxes paid fact, and/or a projected financial fact.
BRIEF DESCRIPTION OF DRAWINGS

The invention will be better understood when consideration is given to the following detailed
description thereof. Such description makes reference to the annexed drawings wherein:

Figure 1 is a schematic diagram of a hardware implementation of one embodiment of the present
invention;

Figure 1a is a flowchart depicting the various steps of one embodiment of the present invention.

Figure 2 is an illustration showing various dimensions of tax-related data which together
comprise a plurality of fact groups in accordance with one embodiment of the present invention;

Figure 3 is an illustration showing various examples of attributes associated with the asset
dimension in accordance with one embodiment of the present invention;

Figure 4 is an illustration showing various examples of attributes associated with the date
dimension in accordance with one embodiment of the present invention;

Figure 5 is an illustration showing various examples of attributes associated with the entity
dimension in accordance with one embodiment of the present invention;

Figure 6 is an illustration showing various examples of attributes associated with the jurisdiction
dimension in accordance with one embodiment of the present invention;

Figure 7 is an illustration showing various examples of attributes associated with the tax
classification dimension in accordance with one embodiment of the present invention;

Figure 8 is an illustration showing various examples of attributes associated with the transaction
dimension in accordance with one embodiment of the present invention;

Figure 9 is an illustration showing a set of the attributes associated with the adjusted book fact in
accordance with one embodiment of the present invention;
Figure 10 is an illustration showing a set of the attributes associated with the tax adjustment fact in accordance with one embodiment of the present invention;

Figure 11 is an illustration showing a set of the attributes associated with the schedule M fact in accordance with one embodiment of the present invention;

Figure 12 is an illustration showing a set of the attributes associated with the tax return fact in accordance with one embodiment of the present invention;

Figure 13 is an illustration showing a set of the attributes associated with the apportionment fact in accordance with one embodiment of the present invention;

Figure 14 is an illustration showing a set of the attributes associated with the line item fact in accordance with one embodiment of the present invention;

Figure 15 is an illustration showing a set of the attributes associated with the transaction amount fact in accordance with one embodiment of the present invention;

Figure 16 is an illustration showing a set of the attributes associated with the invoice fact in accordance with one embodiment of the present invention;

Figure 17 is an illustration showing a set of the attributes associated with the asset fact in accordance with one embodiment of the present invention;

Figure 18 is an illustration showing a set of the attributes associated with the payroll amount fact in accordance with one embodiment of the present invention;

Figure 19 is an illustration showing a set of the attributes associated with the taxes paid fact in accordance with one embodiment of the present invention; and

Figure 20 is an illustration showing a set of the attributes associated with the projected financials fact in accordance with one embodiment of the present invention.
DISCLOSURE OF INVENTION

The present invention is provided for organizing multiple dimensions of tax-related data for generating fact details to facilitate payment of taxes. In order to accomplish this, one embodiment of the present invention is practiced in the context of a personal computer such as an IBM compatible personal computer, Apple Macintosh computer or UNIX based workstation.

A representative hardware environment is depicted in Figure 1, which illustrates a typical hardware configuration of a workstation in accordance with a preferred embodiment having a central processing unit 110, such as a microprocessor, and a number of other units interconnected via a system bus 112. The workstation shown in Figure 1 includes a Random Access Memory (RAM) 114, Read Only Memory (ROM) 116, an I/O adapter 118 for connecting peripheral devices such as disk storage units 120 to the bus 112, a user interface adapter 122 for connecting a keyboard 124, a mouse 126, a speaker 128, a microphone 132, and/or other user interface devices such as a touch screen (not shown) to the bus 112, communication adapter 134 for connecting the workstation to a communication network (e.g., a data processing network) and a display adapter 136 for connecting the bus 112 to a display device 138.

The workstation typically has resident thereon an operating system such as the Microsoft Windows NT or Windows/95 Operating System (OS), the IBM OS/2 operating system, the MAC OS, or UNIX operating system. Those skilled in the art will appreciate that the present invention may also be implemented on platforms and operating systems other than those mentioned.

Operating on the hardware is software. One embodiment of software is written using JAVA, C, and the C++ language and utilizes object oriented programming methodology. Object oriented programming (OOP) has become increasingly used to develop complex applications. As OOP moves toward the mainstream of software design and development, various software solutions require adaptation to make use of the benefits of OOP. A need exists for these principles of OOP to be applied to a messaging interface of an electronic messaging system such that a set of OOP classes and objects for the messaging interface can be provided.
OOP is a process of developing computer software using objects, including the steps of analyzing the problem, designing the system, and constructing the program. An object is a software package that contains both data and a collection of related structures and procedures. Since it contains both data and a collection of structures and procedures, it can be visualized as a self-sufficient component that does not require other additional structures, procedures or data to perform its specific task. OOP, therefore, views a computer program as a collection of largely autonomous components, called objects, each of which is responsible for a specific task. This concept of packaging data, structures, and procedures together in one component or module is called encapsulation.

In general, OOP components are reusable software modules which present an interface that conforms to an object model and which are accessed at run-time through a component integration architecture. A component integration architecture is a set of architecture mechanisms which allow software modules in different process spaces to utilize each others capabilities or functions. This is generally done by assuming a common component object model on which to build the architecture. It is worthwhile to differentiate between an object and a class of objects at this point. An object is a single instance of the class of objects, which is often just called a class. A class of objects can be viewed as a blueprint, from which many objects can be formed.

OOP allows the programmer to create an object that is a part of another object. For example, the object representing a piston engine is said to have a composition-relationship with the object representing a piston. In reality, a piston engine comprises a piston, valves and many other components; the fact that a piston is an element of a piston engine can be logically and semantically represented in OOP by two objects.

OOP also allows creation of an object that “depends from” another object. If there are two objects, one representing a piston engine and the other representing a piston engine wherein the piston is made of ceramic, then the relationship between the two objects is not that of composition. A ceramic piston engine does not make up a piston engine. Rather it is merely one kind of piston engine that has one more limitation than the piston engine; its piston is made of ceramic. In this case, the object representing the ceramic piston engine is called a derived object, and it inherits all of the aspects of the object representing the piston engine and adds further
limitation or detail to it. The object representing the ceramic piston engine “depends from” the object representing the piston engine. The relationship between these objects is called inheritance.

When the object or class representing the ceramic piston engine inherits all of the aspects of the objects representing the piston engine, it inherits the thermal characteristics of a standard piston defined in the piston engine class. However, the ceramic piston engine object overrides these specific thermal characteristics, which are typically different from those associated with a metal piston. It skips over the original and uses new functions related to ceramic pistons. Different kinds of piston engines have different characteristics, but may have the same underlying functions associated with it (e.g., how many pistons in the engine, ignition sequences, lubrication, etc.). To access each of these functions in any piston engine object, a programmer would call the same functions with the same names, but each type of piston engine may have different/overriding implementations of functions behind the same name. This ability to hide different implementations of a function behind the same name is called polymorphism and it greatly simplifies communication among objects.

With the concepts of composition-relationship, encapsulation, inheritance and polymorphism, an object can represent just about anything in the real world. In fact, our logical perception of the reality is the only limit on determining the kinds of things that can become objects in object-oriented software. Some typical categories are as follows:

- Objects can represent physical objects, such as automobiles in a traffic-flow simulation, electrical components in a circuit-design program, countries in an economics model, or aircraft in an air-traffic-control system.
- Objects can represent elements of the computer-user environment such as windows, menus or graphics objects.
- An object can represent an inventory, such as a personnel file or a table of the latitudes and longitudes of cities.
- An object can represent user-defined data types such as time, angles, and complex numbers, or points on the plane.

With this enormous capability of an object to represent just about any logically separable matters, OOP allows the software developer to design and implement a computer program that is
a model of some aspects of reality, whether that reality is a physical entity, a process, a system, or a composition of matter. Since the object can represent anything, the software developer can create an object which can be used as a component in a larger software project in the future.

If 90% of a new OOP software program consists of proven, existing components made from preexisting reusable objects, then only the remaining 10% of the new software project has to be written and tested from scratch. Since 90% already came from an inventory of extensively tested reusable objects, the potential domain from which an error could originate is 10% of the program. As a result, OOP enables software developers to build objects out of other, previously built objects.

This process closely resembles complex machinery being built out of assemblies and sub-assemblies. OOP technology, therefore, makes software engineering more like hardware engineering in that software is built from existing components, which are available to the developer as objects. All this adds up to an improved quality of the software as well as an increased speed of its development.

Programming languages are beginning to fully support the OOP principles, such as encapsulation, inheritance, polymorphism, and composition-relationship. With the advent of the C++ language, many commercial software developers have embraced OOP. C++ is an OOP language that offers a fast, machine-executable code. Furthermore, C++ is suitable for both commercial-application and systems-programming projects. For now, C++ appears to be the most popular choice among many OOP programmers, but there is a host of other OOP languages, such as Smalltalk, Common Lisp Object System (CLOS), and Eiffel. Additionally, OOP capabilities are being added to more traditional popular computer programming languages such as Pascal.

The benefits of object classes can be summarized, as follows:

- Objects and their corresponding classes break down complex programming problems into many smaller, simpler problems.
- Encapsulation enforces data abstraction through the organization of data into small, independent objects that can communicate with each other. Encapsulation protects the
data in an object from accidental damage, but allows other objects to interact with that
data by calling the object's member functions and structures.

- Subclassing and inheritance make it possible to extend and modify objects through
deriving new kinds of objects from the standard classes available in the system. Thus,
new capabilities are created without having to start from scratch.

- Polymorphism and multiple inheritance make it possible for different programmers to
mix and match characteristics of many different classes and create specialized objects
that can still work with related objects in predictable ways.

- Class hierarchies and containment hierarchies provide a flexible mechanism for modeling
real-world objects and the relationships among them.

- Libraries of reusable classes are useful in many situations, but they also have some
limitations. For example:

  - Complexity. In a complex system, the class hierarchies for related classes can become
    extremely confusing, with many dozens or even hundreds of classes.

  - Flow of control. A program written with the aid of class libraries is still responsible for
    the flow of control (i.e., it must control the interactions among all the objects created
    from a particular library). The programmer has to decide which functions to call at what
times for which kinds of objects.

  - Duplication of effort. Although class libraries allow programmers to use and reuse many
    small pieces of code, each programmer puts those pieces together in a different way.
    Two different programmers can use the same set of class libraries to write two programs
    that do exactly the same thing but whose internal structure (i.e., design) may be quite
different, depending on hundreds of small decisions each programmer makes along the
way. Inevitably, similar pieces of code end up doing similar things in slightly different
ways and do not work as well together as they should.

Class libraries are very flexible. As programs grow more complex, more programmers are
forced to reinvent basic solutions to basic problems over and over again. A relatively new
extension of the class library concept is to have a framework of class libraries. This framework
is more complex and consists of significant collections of collaborating classes that capture both
the small scale patterns and major mechanisms that implement the common requirements and
design in a specific application domain. They were first developed to free application
programmers from the chores involved in displaying menus, windows, dialog boxes, and other standard user interface elements for personal computers.

Frameworks also represent a change in the way programmers think about the interaction between the code they write and code written by others. In the early days of procedural programming, the programmer called libraries provided by the operating system to perform certain tasks, but basically the program executed down the page from start to finish, and the programmer was solely responsible for the flow of control. This was appropriate for printing out paychecks, calculating a mathematical table, or solving other problems with a program that executed in just one way.

The development of graphical user interfaces began to turn this procedural programming arrangement inside out. These interfaces allow the user, rather than program logic, to drive the program and decide when certain actions should be performed. Today, most personal computer software accomplishes this by means of an event loop which monitors the mouse, keyboard, and other sources of external events and calls the appropriate parts of the programmer’s code according to actions that the user performs. The programmer no longer determines the order in which events occur. Instead, a program is divided into separate pieces that are called at unpredictable times and in an unpredictable order. By relinquishing control in this way to users, the developer creates a program that is much easier to use. Nevertheless, individual pieces of the program written by the developer still call libraries provided by the operating system to accomplish certain tasks, and the programmer must still determine the flow of control within each piece after it’s called by the event loop. Application code still “sits on top of” the system.

Even event loop programs require programmers to write a lot of code that should not need to be written separately for every application. The concept of an application framework carries the event loop concept further. Instead of dealing with all the nuts and bolts of constructing basic menus, windows, and dialog boxes and then making these things all work together, programmers using application frameworks start with working application code and basic user interface elements in place. Subsequently, they build from there by replacing some of the generic capabilities of the framework with the specific capabilities of the intended application.
Application frameworks reduce the total amount of code that a programmer has to write from scratch. However, because the framework is really a generic application that displays windows, supports copy and paste, and so on, the programmer can also relinquish control to a greater degree than event loop programs permit. The framework code takes care of almost all event handling and flow of control, and the programmer’s code is called only when the framework needs it (e.g., to create or manipulate a proprietary data structure).

A programmer writing a framework program not only relinquishes control to the user (as is also true for event loop programs), but also relinquishes the detailed flow of control within the program to the framework. This approach allows the creation of more complex systems that work together in interesting ways, as opposed to isolated programs, having custom code, being created over and over again for similar problems.

Thus, as is explained above, a framework basically is a collection of cooperating classes that make up a reusable design solution for a given problem domain. It typically includes objects that provide default behavior (e.g., for menus and windows), and programmers use it by inheriting some of that default behavior and overriding other behavior so that the framework calls application code at the appropriate times.

There are three main differences between frameworks and class libraries:

- Behavior versus protocol. Class libraries are essentially collections of behaviors that you can call when you want those individual behaviors in your program. A framework, on the other hand, provides not only behavior but also the protocol or set of rules that govern the ways in which behaviors can be combined, including rules for what a programmer is supposed to provide versus what the framework provides.

- Call versus override. With a class library, the code the programmer instantiates objects and calls their member functions. It’s possible to instantiate and call objects in the same way with a framework (i.e., to treat the framework as a class library), but to take full advantage of a framework’s reusable design, a programmer typically writes code that overrides and is called by the framework. The framework manages the flow of control among its objects. Writing a program involves dividing responsibilities among the various pieces of software that are called by the framework rather than specifying how the different pieces should work together.
• Implementation versus design. With class libraries, programmers reuse only implementations, whereas with frameworks, they reuse design. A framework embodies the way a family of related programs or pieces of software work. It represents a generic design solution that can be adapted to a variety of specific problems in a given domain. For example, a single framework can embody the way a user interface works, even though two different user interfaces created with the same framework might solve quite different interface problems.

Thus, through the development of frameworks for solutions to various problems and programming tasks, significant reductions in the design and development effort for software can be achieved. A preferred embodiment of the invention utilizes HyperText Markup Language (HTML) to implement documents on the Internet together with a general-purpose secure communication protocol for a transport medium between the client and the Newco. HTTP or other protocols could be readily substituted for HTML without undue experimentation. Information on these products is available in T. Berners-Lee, D. Connoly, "RFC 1866: Hypertext Markup Language - 2.0" (Nov. 1995); and R. Fielding, H. Frystyk, T. Berners-Lee, J. Gettys and J.C. Mogul, "Hypertext Transfer Protocol -- HTTP/1.1: HTTP Working Group Internet Draft" (May 2, 1996). HTML is a simple data format used to create hypertext documents that are portable from one platform to another. HTML documents are SGML documents with generic semantics that are appropriate for representing information from a wide range of domains. HTML has been in use by the World-Wide Web global information initiative since 1990. HTML is an application of ISO Standard 8879; 1986 Information Processing Text and Office Systems; Standard Generalized Markup Language (SGML).

To date, Web development tools have been limited in their ability to create dynamic Web applications which span from client to server and interoperate with existing computing resources. Until recently, HTML has been the dominant technology used in development of Web-based solutions. However, HTML has proven to be inadequate in the following areas:

• Poor performance;
• Restricted user interface capabilities;
• Can only produce static Web pages;
• Lack of interoperability with existing applications and data; and
• Inability to scale.
Sun Microsystem's Java language solves many of the client-side problems by:

- Improving performance on the client side;
- Enabling the creation of dynamic, real-time Web applications; and
- Providing the ability to create a wide variety of user interface components.

With Java, developers can create robust User Interface (UI) components. Custom "widgets" (e.g., real-time stock tickers, animated icons, etc.) can be created, and client-side performance is improved. Unlike HTML, Java supports the notion of client-side validation, offloading appropriate processing onto the client for improved performance. Dynamic, real-time Web pages can be created. Using the above-mentioned custom UI components, dynamic Web pages can also be created.

Sun's Java language has emerged as an industry-recognized language for "programming the Internet." Sun defines Java as: "a simple, object-oriented, distributed, interpreted, robust, secure, architecture-neutral, portable, high-performance, multithreaded, dynamic, buzzword-compliant, general-purpose programming language. Java supports programming for the Internet in the form of platform-independent Java applets." Java applets are small, specialized applications that comply with Sun's Java Application Programming Interface (API) allowing developers to add "interactive content" to Web documents (e.g., simple animations, page adornments, basic games, etc.). Applets execute within a Java-compatible browser (e.g., Netscape Navigator) by copying code from the server to client. From a language standpoint, Java's core feature set is based on C++. Sun's Java literature states that Java is basically, "C++ with extensions from Objective C for more dynamic method resolution."

Another technology that provides similar function to JAVA is provided by Microsoft and ActiveX Technologies, to give developers and Web designers wherewithal to build dynamic content for the Internet and personal computers. ActiveX includes tools for developing animation, 3-D virtual reality, video and other multimedia content. The tools use Internet standards, work on multiple platforms, and are being supported by over 100 companies. The group's building blocks are called ActiveX Controls, small, fast components that enable developers to embed parts of software in hypertext markup language (HTML) pages. ActiveX Controls work with a variety of programming languages including Microsoft Visual C++,
Borland Delphi, Microsoft Visual Basic programming system and, in the future, Microsoft's development tool for Java, code named "Jakarta." ActiveX Technologies also includes ActiveX Server Framework, allowing developers to create server applications. One of ordinary skill in the art readily recognizes that ActiveX could be substituted for JAVA without undue experimentation to practice the invention.

Figure 1A is a flowchart illustrating the various operations of the present invention that may be carried out by the hardware and software tools set forth hereinabove. In the most general terms, the process begins in operation 150. Next, a plurality of dimensions of tax-related data to be submitted in a tax return are provided along with a plurality of attributes for each of the dimensions. See operation 152 and 154, respectively. The dimensions include information pertinent to the calculation of tax liability. Such information includes tax classification and entity type. This information may be further classified in the various attributes of the dimensions.

With continuing reference to Figure 1A, and in particular operation 156, a plurality of entries are shown to be received which are associated with the attributes of the dimensions. A plurality of fact details are then generated by aggregating the entries of a predetermined set of the attributes of some or all of the dimensions, as indicated in operation 158. Thereafter, in operation 160 the fact details are outputted for various reasons such as submission with the tax return after which the process is terminated in operation 162.

It should be noted that the dimensions, attributes, and entries may be inputted by any input device such as the keyboard 124, the mouse 126, the microphone 132, the touch screen (not shown), or anything else that is capable of conveying such information. As an option, the dimensions attributes need only be entered initially once, or simply provided.

Further, the generation of the fact details may be carried out via the CPU 110 which in turn may be governed by a computer program stored on a computer readable medium, i.e. the RAM 114, ROM 116, the disk storage units 120, and/or anything else capable of storing the computer program. In the alternative, dedicated hardware such as an application specific integrated circuit (ASIC) may be employed to accomplish the same.
In order to generate the fact details, a computer program, or any other type of logic, is adapted to use different sets of the attributes of some or all of the dimensions for calculation of each of the various fact details. Upon the appropriate selection of attributes that are applicable to the calculation of the fact detail at hand, the fact detail may be calculated using equations and rules set forth by the desired tax preparation practice. In the alternative, such calculation of the fact details may be carried out manually, thereby using the present invention primarily for the collection, organization, and convenient access to the tax-related data.

The outputting of the fact details may be effected by way of the display 138, the speaker 128, a printer (not shown) or any other output mechanism capable of delivering the results to the user. It should be understood that the foregoing components need not be resident on a single computer, but also may be a component of either a networked client and/or a server.

In one embodiment of the present invention, a description may be provided for each of the attributes, dimensions, and fact details. Such description may facilitate the inputting of the entries or serve to explain the generated fact details.

In use, the system may be capable of supporting data requirements for federal and state income tax filing; supporting the analysis of foreign sale corporation activity on a transaction-by-transaction basis, storing and retrieving transaction data to support federal income tax information document requests, analyzing sales and using tax refund opportunities, comparing specific valuation bases for property tax reporting, supplying financial data on a jurisdictional basis to support state and local credit and incentives analysis, and storing and retrieving data related to income projection and schedule M reversals to support deferred planning. The present invention may thus include an API which can handle any tax-related matter including those listed above, and may further be capable of allowing convenient retrieval and input of tax-related fact details through an automated organization method which will soon be set forth.

**Dimensions & Attributes**

In one embodiment of the present invention, the dimensions may include asset 200, date 202, entity 204, jurisdiction 206, tax classification 208, and/or transaction 210 dimensions. Note
Figure 2 as well as the following table, which lists and describes each of the dimensions found in Figure 2 and used for generation of fact details.

<table>
<thead>
<tr>
<th>Dimension Name</th>
<th>Dimension Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asset</td>
<td>Contains all of the attributes associated with an asset.</td>
</tr>
<tr>
<td>Date</td>
<td>Contains attributes about the time when an activity occurred.</td>
</tr>
<tr>
<td>Entity</td>
<td>Contains all the attributes associated with a legal entity.</td>
</tr>
<tr>
<td>Jurisdiction</td>
<td>Contains all the attributes of the taxing jurisdiction.</td>
</tr>
<tr>
<td>Tax Classification</td>
<td>Represents the categories by which information is reported on various returns.</td>
</tr>
<tr>
<td>Transaction</td>
<td>Contains all the attributes of all source systems from which information was imported.</td>
</tr>
</tbody>
</table>

Various attributes of the asset dimension 200 of the tax-related data are shown in Figure 3. The following table lists each attribute of the asset dimension 200 shown in Figure 3 and its exemplary description. Also included are sample entries that are used for, for example, calculating fact details and tax cost or identifying a particular asset.

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Attribute Description</th>
<th>Sample Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asset ID</td>
<td>Represents a unique identifier assigned to individual assets in the property subsystem.</td>
<td>1234</td>
</tr>
<tr>
<td>Asset Category</td>
<td>Represents the category of assets in the property subsystem.</td>
<td>Vehicles</td>
</tr>
<tr>
<td>Asset Description</td>
<td>Represents more detail on individual asset in the property subsystem.</td>
<td>1997 Chevy Truck</td>
</tr>
<tr>
<td>Asset Status</td>
<td>Identifies the condition of each asset.</td>
<td>New, used, obsolete</td>
</tr>
</tbody>
</table>
Various attributes of the date dimension **202** of the tax-related data are shown in Figure 4 and further characterized as follows (*shaded areas show future attributes):

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Attribute Description</th>
<th>Sample Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date</td>
<td>The specific day that a transaction</td>
<td>2/28/1998</td>
</tr>
<tr>
<td>Attribute Name</td>
<td>Attribute Description</td>
<td>Sample Values</td>
</tr>
<tr>
<td>------------------</td>
<td>---------------------------------------------------------------------------------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>Calendar Week</td>
<td>The calendar week.</td>
<td>Week 1 1999</td>
</tr>
<tr>
<td>Calendar Month</td>
<td>The calendar month.</td>
<td>January, 1998</td>
</tr>
<tr>
<td>Calendar Quarter</td>
<td>The calendar quarter.</td>
<td>1998Q1</td>
</tr>
<tr>
<td>Calendar Year</td>
<td>The calendar year.</td>
<td>1998</td>
</tr>
<tr>
<td>Fiscal Week</td>
<td>The week that represents the corporate calendar. Note that the F in the data value indicates that this is a fiscal time period.</td>
<td>F Week 1 1999, F Week 46 1999</td>
</tr>
<tr>
<td>Fiscal Month</td>
<td>The fiscal period comprised of 4 or 5 weeks. Note that the F in the data value indicates that this is a fiscal time period.</td>
<td>F January, 1998</td>
</tr>
<tr>
<td>Fiscal Quarter</td>
<td>The grouping of 3 fiscal months.</td>
<td>F 1998Q1</td>
</tr>
<tr>
<td>Fiscal Year</td>
<td>The grouping of 52 fiscal weeks / 12 fiscal months that comprise the financial year.</td>
<td>F 1998</td>
</tr>
<tr>
<td>Filing Date</td>
<td>The actual date the return was filed.</td>
<td>5/15/1998</td>
</tr>
</tbody>
</table>

Various attributes of the entity dimension 204 of the tax-related data are shown in Figure 5 and further characterized as follows:

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Attribute Description</th>
<th>Sample Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost Center</td>
<td>Indicates a specific cost center that has a budget. This is the lowest level of the corporate organizational structure.</td>
<td>Sales</td>
</tr>
<tr>
<td>Consolidated Group</td>
<td>Represents the companies, which are grouped together for reporting</td>
<td>North America</td>
</tr>
<tr>
<td>Attribute Name</td>
<td>Attribute Description</td>
<td>Sample Values</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------------------------------------------------------------------------------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>Company</td>
<td>Represents the different companies as separate legal entities.</td>
<td>XYZ, Inc.</td>
</tr>
<tr>
<td>Company code</td>
<td>Represents the company's unique internal numeric identifier.</td>
<td>ABC12345</td>
</tr>
<tr>
<td>Taxpayer ID</td>
<td>Represents the legal entity's unique 9 digit tax identification #.</td>
<td>74-1751406</td>
</tr>
<tr>
<td>Date of incorporation</td>
<td>Represents the legal entity's date of incorporation.</td>
<td>January 1, 1998</td>
</tr>
<tr>
<td>Place of incorporation</td>
<td>Represents where legal entity was incorporated.</td>
<td>Texas</td>
</tr>
<tr>
<td>Division</td>
<td>Major operating areas of the company.</td>
<td>Commercial Services</td>
</tr>
<tr>
<td>Region</td>
<td>Business units that are grouped geographically for field organization and home office by functional area.</td>
<td>Midwest</td>
</tr>
<tr>
<td>Business Unit</td>
<td>A grouping of cost centers.</td>
<td>Greater Chicago</td>
</tr>
</tbody>
</table>

Various attributes of the jurisdiction dimension **206** of the tax-related data are shown in Figure 6 and further characterized as follows:

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Attribute Description</th>
<th>Sample Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Represents the name of the taxing authority.</td>
<td>City of Houston</td>
</tr>
<tr>
<td>Jurisdiction type</td>
<td>Represents the type of taxing authority.</td>
<td>Property, sales, etc.</td>
</tr>
<tr>
<td>Location ID</td>
<td>Represents a unique identifier to distinguish location at a level</td>
<td>Subdivide sections of a plant or facility</td>
</tr>
<tr>
<td>Attribute Name</td>
<td>Attribute Description</td>
<td>Sample Values</td>
</tr>
<tr>
<td>-----------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-----------------------------------------</td>
</tr>
<tr>
<td>Geo code</td>
<td>Represents a unique identifier similar to zip code.</td>
<td>77777-12345678</td>
</tr>
<tr>
<td>School District</td>
<td>Represents the name of the school district / tax collector the Geo code area is responsible for remitting taxes to.</td>
<td>Houston I.S.D</td>
</tr>
<tr>
<td>City</td>
<td>Represents the name of the city where Geo code area is located.</td>
<td>Houston</td>
</tr>
<tr>
<td>County</td>
<td>Represents the name of the county where Geo code area is located.</td>
<td>Harris</td>
</tr>
<tr>
<td>State</td>
<td>Represents the name of the state where Geo code area is located.</td>
<td>Texas</td>
</tr>
<tr>
<td>Zip code</td>
<td>The 5 digit zip code where Geo code area is located.</td>
<td>77042</td>
</tr>
<tr>
<td>Zip code+4</td>
<td>The 9 digit zip code where Geo code area is located.</td>
<td>77042-1041</td>
</tr>
<tr>
<td>Country</td>
<td>Represents the name of the country where Geo code area is located.</td>
<td>USA</td>
</tr>
</tbody>
</table>

Various attributes of the tax classification dimension 208 of the tax-related data are shown in Figure 7 and further characterized as follows:

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Attribute Description</th>
<th>Sample Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tax Category</td>
<td>Represents the type of information reported on the return. (This attribute is meant to capture information by section on the tax form.)</td>
<td>Income, Expense, Tax, Credits, Assets</td>
</tr>
<tr>
<td>Description</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Attribute Name</td>
<td>Attribute Description</td>
<td>Sample Values</td>
</tr>
<tr>
<td>----------------</td>
<td>-----------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>Tax Sub - Category Description</td>
<td>Represents the type of information reported on the return. (This attribute is meant to capture information by line item on the tax form.)</td>
<td>Sales, Cash</td>
</tr>
<tr>
<td>Account ID</td>
<td>Represents the GL or other account identifier. These accounts are summarized into sub-categories.</td>
<td>1234</td>
</tr>
<tr>
<td>Account Description</td>
<td>Represents the GL or other account description.</td>
<td>Cash - Bank One #1234</td>
</tr>
<tr>
<td>Account Group Level 3</td>
<td>A collection of GL Accounts as determined by finance</td>
<td>Employee Salary, Contract Labor</td>
</tr>
<tr>
<td>Account Group Level 2</td>
<td>A collection of Account Group Level 3 accounts as defined by finance</td>
<td>Human Resources</td>
</tr>
<tr>
<td>Account Group Level 1</td>
<td>A collection of Account Group Level 2 accounts as defined by finance</td>
<td>Legal Fees, Capital Equipment, Salary &amp; Benefits</td>
</tr>
</tbody>
</table>

Various attributes of the transaction dimension 210 of the tax-related data are shown in Figure 8 and further characterized as follows:

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Attribute Description</th>
<th>Sample Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transaction ID</td>
<td>Identifies the document posting # in the source system.</td>
<td>Invoice #, Check #, Journal Entry #, Batch Posting #</td>
</tr>
<tr>
<td>Transaction Type</td>
<td>Identifies the document posting type in the source system.</td>
<td>Accruals, Sales, Payments</td>
</tr>
</tbody>
</table>
| Transaction Description | Represents the text descriptive field for the document posting. | "To accrue Texas Franchise tax for
<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Attribute Description</th>
<th>Sample Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purchase Order Number</td>
<td>Identifies the document posting purchase order information, if applicable.</td>
<td>1234</td>
</tr>
<tr>
<td>AFE Number</td>
<td>Identifies the document posting project number information, if applicable.</td>
<td>1234</td>
</tr>
<tr>
<td>SUE Type</td>
<td>Describes the transaction in terms to identify the taxability of the transaction. (Needed for RAM)</td>
<td>Maintenance, Equipment rentals, new construction, etc.</td>
</tr>
<tr>
<td>Tax Type</td>
<td>Identifies the type of tax accrued / paid.</td>
<td>sales, use, excise, property, fed income, state income, etc.</td>
</tr>
<tr>
<td>Customer Account ID</td>
<td>Identifies the customer (referenced in the document posting) who products / services were sold to.</td>
<td>XYZ, Inc.</td>
</tr>
<tr>
<td>Vendor Account ID</td>
<td>Identifies the vendor or payee (referenced in the document posting who payments for products / services / taxes were made to.</td>
<td>ABC, Inc.</td>
</tr>
<tr>
<td>Contract Number</td>
<td>Identifies the legal file reference #.</td>
<td>1234</td>
</tr>
<tr>
<td>Contract Type</td>
<td>Identifies the type of contract for tax purposes.</td>
<td>Lump sum, separated</td>
</tr>
<tr>
<td>Sold To Location Geo Code</td>
<td>Identifies the location of where sale / purchase originated.</td>
<td>77042-1041</td>
</tr>
<tr>
<td>Ship To Location</td>
<td>Identifies the location of where</td>
<td>78645-1001</td>
</tr>
<tr>
<td>Attribute Name</td>
<td>Attribute Description</td>
<td>Sample Values</td>
</tr>
<tr>
<td>----------------</td>
<td>--------------------------------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>Geo Code</td>
<td>sale / purchase was shipped / consumed.</td>
<td></td>
</tr>
<tr>
<td>Line Item</td>
<td>Identifies the detail at a level below the total transaction or invoice.</td>
<td>Product, freight costs, account distribution</td>
</tr>
</tbody>
</table>

**Fact Details**

In one embodiment of the present invention, the fact details may include an adjusted book fact, a tax adjustment fact, a schedule M fact, a tax return fact, an apportionment fact, a line item fact, a transaction fact, an invoice fact, an asset fact, a payroll fact, a taxes paid fact, and/or a projected financial fact.

An example of calculation of an adjusted book fact is shown in Figure 9. The dimensions necessary for the exemplary calculation of an adjusted book amount, such as of the level of an account, are shown along with a selected attribute that define the parameters of the final calculated fact. In this example, the month attribute of the date dimension is used, meaning that the level of the account at the end of the month is being calculated. The type of entity is classified under the business unit attribute of the entity dimension. Also, the proper jurisdiction type attribute, such as property, sales, or income tax, is selected along with the jurisdiction of the taxing authority, i.e., state, federal, foreign, or special, are selected from the of the jurisdiction dimension. A particular tax sub category of the tax classification dimension is used to identify the type of information that will be reported on the return, such as for capturing information by line item on the tax form. Examples of the tax sub category for the account in this example include sales and cash. Also, as noted in Figure 9, the transaction and asset dimensions are not used to calculate the adjusted book fact in this example.

The following table illustrates the fact name shown in Figure 9 as well as describes the fact and provides a default aggregation rule:
<table>
<thead>
<tr>
<th>Fact Name</th>
<th>Fact Description</th>
<th>Default Aggregation Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adjusted Book Amount</td>
<td>The dollar amount per category, subcategory, down to the account level of the final audited (adjusted book) financial statements</td>
<td>Business Unit</td>
</tr>
</tbody>
</table>

Examples of the tax adjustment facts 1000 and associated exemplary descriptions are shown in Figure 10 and further characterized as follows:

<table>
<thead>
<tr>
<th>Fact Name</th>
<th>Fact Description</th>
<th>Default Aggregation Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tax Adjustment Amount</td>
<td>The dollar amount per category, subcategory, down to the account level of the tax reclassifications to the adjusted book $ for tax purposes.</td>
<td>Business Unit</td>
</tr>
</tbody>
</table>

Examples of the schedule M facts 1100 and associated exemplary descriptions are shown in Figure 11 and further characterized as follows:

<table>
<thead>
<tr>
<th>Fact Name</th>
<th>Fact Description</th>
<th>Default Aggregation Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schedule M Amount</td>
<td>The current years “Schedule M” book to tax reconciling items - examples include the 50 % nondeductible portion of travel &amp; entertainment expenses, tax exempt interest income, etc.</td>
<td>Business Unit</td>
</tr>
</tbody>
</table>
Examples of the tax return facts 1200 and associated exemplary descriptions are shown in Figure 12 and further characterized as follows:

<table>
<thead>
<tr>
<th>Fact Name</th>
<th>Fact Description</th>
<th>Default Aggregation Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tax Return Amount</td>
<td>The dollar amount per category and subcategory of the items per the tax return.</td>
<td>Return Level</td>
</tr>
</tbody>
</table>

Examples of the tax apportionment facts 1300 and associated exemplary descriptions are shown in Figure 13 and further characterized as follows:

<table>
<thead>
<tr>
<th>Fact Name</th>
<th>Fact Description</th>
<th>Default Aggregation Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sales Percentage</td>
<td>Sales dollar amount by jurisdiction</td>
<td>Jurisdiction City</td>
</tr>
<tr>
<td>Property Percentage</td>
<td>Property dollar amount by jurisdiction</td>
<td>Jurisdiction City</td>
</tr>
<tr>
<td>Payroll Percentage</td>
<td>Payroll dollar amount by jurisdiction</td>
<td>Jurisdiction City</td>
</tr>
<tr>
<td>Apportionment Percentage</td>
<td>The result of the above three facts, weighted according to individual taxing jurisdiction parameters to determine overall tax rate applied on income tax return.</td>
<td>Jurisdiction City</td>
</tr>
</tbody>
</table>

Examples of the line item facts 1400 and associated exemplary descriptions are shown in Figure 14 and further characterized as follows:
Examples of the transaction facts **1500** and associated exemplary descriptions are shown in Figure **15** and further characterized as follows:

<table>
<thead>
<tr>
<th>Fact Name</th>
<th>Fact Description</th>
<th>Default Aggregation Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Line Item Amount</td>
<td>The dollar amount at a level lower than the total transaction, such as the distribution to other GL accounts, freight / other costs, tax paid, etc.</td>
<td></td>
</tr>
<tr>
<td>Transaction Amount</td>
<td>The dollar amount per individual transaction document posting. This is an aggregate of line item.</td>
<td>Transactions will be stored at the account level. When transaction amounts are needed at the Transaction ID level, they will be derived from the Line Item Fact Table.</td>
</tr>
</tbody>
</table>

Examples of the invoice facts **1600** and associated exemplary descriptions are shown in Figure **16** and further characterized as follows:

<table>
<thead>
<tr>
<th>Fact Name</th>
<th>Fact Description</th>
<th>Default Aggregation Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Invoice Amount</td>
<td>Total dollar amount of all line items on the invoice. (* For detailed line item totals see the)</td>
<td>Summarized at the jurisdiction level</td>
</tr>
<tr>
<td>Fact Name</td>
<td>Fact Description</td>
<td>Default Aggregation Rule</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------------------------------------------------------------------------</td>
<td>-----------------------------------------------</td>
</tr>
<tr>
<td>Pre Tax Invoice Amount</td>
<td>Dollar amount of line items on the invoice, before tax is paid or accrued</td>
<td>Summarized at the jurisdiction level</td>
</tr>
<tr>
<td>Exemption Amount</td>
<td>Dollar amount of line items on the invoice which are not taxable</td>
<td>Summarized at the jurisdiction level</td>
</tr>
<tr>
<td>Taxes Paid</td>
<td>Dollar amount of taxes paid on the invoice</td>
<td>Summarized at the jurisdiction level</td>
</tr>
<tr>
<td>Taxes Accrued</td>
<td>Dollar amount of taxes accrued on the invoice, to be paid at a later date</td>
<td>Summarized at the jurisdiction level</td>
</tr>
</tbody>
</table>

Examples of the asset facts 1700 and associated exemplary descriptions are shown in Figure 17 and further characterized as follows:

<table>
<thead>
<tr>
<th>Fact Name</th>
<th>Fact Description</th>
<th>Default Aggregation Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Book Basis</td>
<td>The dollar amount of the asset in accordance with GAAP</td>
<td>Summarized at the jurisdiction level</td>
</tr>
<tr>
<td>Accumulated Depreciation</td>
<td>The dollar amount of the book accumulated depreciation for financial purposes in accordance with GAAP</td>
<td>Summarized at the jurisdiction level</td>
</tr>
<tr>
<td>Assessed Value</td>
<td>Reflects the taxing authority’s opinion of the current market value of the asset</td>
<td>Summarized at the jurisdiction level</td>
</tr>
<tr>
<td>Replacement Cost</td>
<td>The current cost to replace the asset</td>
<td>Summarized at the jurisdiction level</td>
</tr>
</tbody>
</table>
Examples of the payroll facts 1800 and associated exemplary descriptions are shown in Figure 18 and further characterized as follows:

<table>
<thead>
<tr>
<th>Fact Name</th>
<th>Fact Description</th>
<th>Default Aggregation Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Payroll Amount</td>
<td>Total amount of salaries and wages paid in a jurisdiction</td>
<td>Summarized at the city jurisdiction level</td>
</tr>
<tr>
<td>Employee Headcount</td>
<td>The number of employees in a jurisdiction</td>
<td>Summarized at the city jurisdiction level</td>
</tr>
<tr>
<td>Employee Days</td>
<td>The number of days (attendance)</td>
<td>Summarized at the city jurisdiction level</td>
</tr>
<tr>
<td>Officer Salaries</td>
<td>The amount of salaries paid to officers of the business unit within a jurisdiction</td>
<td>Summarized at the city jurisdiction level</td>
</tr>
</tbody>
</table>

Examples of the taxes paid facts 1900 and associated exemplary descriptions are shown in Figure 19 and further characterized as follows:

<table>
<thead>
<tr>
<th>Fact Name</th>
<th>Fact Description</th>
<th>Default Aggregation Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sales Tax Amount</td>
<td></td>
<td>Summarized at the city jurisdiction level</td>
</tr>
<tr>
<td>Fact Name</td>
<td>Fact Description</td>
<td>Default Aggregation Rule</td>
</tr>
<tr>
<td>----------------------</td>
<td>----------------------------------------------------------------------------------</td>
<td>-----------------------------------------------</td>
</tr>
<tr>
<td>Use Tax Amount</td>
<td></td>
<td>Summarized at the city jurisdiction level</td>
</tr>
<tr>
<td>Excise Tax Amount</td>
<td></td>
<td>Summarized at the city jurisdiction level</td>
</tr>
<tr>
<td>Property Tax Amount</td>
<td></td>
<td>Summarized at the city jurisdiction level</td>
</tr>
<tr>
<td>Income Tax Amount</td>
<td></td>
<td>Summarized at the city jurisdiction level</td>
</tr>
<tr>
<td>Franchise Tax Amount</td>
<td></td>
<td>Summarized at the city jurisdiction level</td>
</tr>
<tr>
<td>Other Tax Amount</td>
<td></td>
<td>Summarized at the city jurisdiction level</td>
</tr>
</tbody>
</table>

Examples of the projected financial facts **2000** and associated exemplary descriptions are shown in Figure 20 and further characterized as follows:

<table>
<thead>
<tr>
<th>Fact Name</th>
<th>Fact Description</th>
<th>Default Aggregation Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Projected Book Net Income</td>
<td>Projected income less expenses for financial purposes</td>
<td>Summarized at the business unit level level</td>
</tr>
<tr>
<td>Projected Permanent Differences</td>
<td>Projected items which are to be excluded for tax purposes (505 travel &amp; entertainment, tax exempt interest income)</td>
<td>Summarized at the business unit level level</td>
</tr>
<tr>
<td>Projected Timing</td>
<td>Projected items which need to be</td>
<td>Summarized at the</td>
</tr>
<tr>
<td>Fact Name</td>
<td>Fact Description</td>
<td>Default Aggregation Rule</td>
</tr>
<tr>
<td>-----------</td>
<td>------------------</td>
<td>-------------------------</td>
</tr>
<tr>
<td>Differences</td>
<td>adjusted to reflect tax limitations, methods, etc. (excess contributions carryforwards, book vs. tax depreciation)</td>
<td>business unit level</td>
</tr>
</tbody>
</table>

The following are the various tiers of data within the Shared Dimensions that may be used in one implementation of the present invention. The hierarchical manner in which the data is prioritized or sorted is also illustrated.

Dimension: Account
Levels: Account ID

Dimension: Asset
Levels: • Asset Category
  • Asset ID

Dimension: Date
Levels: • Year
  • Quarter
    • Month

Dimension: Entity
Levels: • Consolidated Group
  • Company
    • Region
      • Division
        • Business Unit

Dimension: Entity_Company
Levels: • Consolidated Group
• Company

Dimension: Invoice Number
Levels: • PO Number

5
Dimension: Jurisdiction_City
Levels: • Country
  • State
    • City

10
Dimension: Jurisdiction_Geo Code
Levels: • Country
  • State
    • City
    • County
      • School District
        • Geo Code

Dimension: Jurisdiction_Type
Levels: • Country
  • Jurisdiction Type
    • Name

Dimension: Tax_Account ID
Levels: • Tax Category
  • Tax Subcategory
    • Account ID
      • Account Description

30
Dimension: Tax_Subcategory
Levels: • Tax Category
  • Tax Subcategory
Dimension: Transaction_Description  
Levels: * Transaction_Description

Dimension: Transaction FSC  
Levels: * Ship To Geo Code
  • Customer Ac ID
    • Transaction ID
    • Line Item
      • Transaction Description

Dimension: Transaction ID  
Levels: * Transaction ID

Dimension: Transaction ID_Descp  
Levels: * Transaction ID
  • Transaction Description

Dimension: Transaction Type_ID_Descp  
Levels: * Transaction_Type
  • Transaction ID
    • Transaction Description

Dimension: Transaction_Line Item  
Levels: * Transaction Type
  • Tax Type
    • Line Item

Dimension: Customer ID  
Levels: * Customer Ac ID
  • OLAP Cubes

Cubes
The following sets illustrate various examples of dimensions and attributes necessary to determine exemplary facts.

**Cube: Adjusted Book**  (See Figure 9.)

5

Fact Table:  adjusted_book_fact_tbl  
Measure(s)  
Adjusted Book Amount

10  
Dimension  
  Grain (attribute)  
  Date  
  month  
  Entity  
  business unit  
  Jurisdiction_Type  
  country  
  Tax_Subcategory  
  sub-category

**Cube: Apportionment Factor Comparison**

Fact Table:  apportionment_fact_tbl  
Measure(s)  
  Apportionment Percent  
  Payroll Percent  
  Property Percent  
  Sales Percent

25  
Dimension  
  Grain  
  Date  
  year  
  Entity_Company  
  business unit  
  Jurisdiction_Geo Code  
  Geo Code  
  Tax_Account ID  
  Account (Description)

30  
**Cube: FSC Transaction-by-Transaction Analysis**

Fact Table:  line_item_fact_tbl
Measure(s)
Line Item Amount

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Grain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date</td>
<td>month</td>
</tr>
<tr>
<td>Entity</td>
<td>business unit</td>
</tr>
<tr>
<td>Jurisdiction_Type</td>
<td>jurisdiction type</td>
</tr>
<tr>
<td>Tax_Subcategory</td>
<td>sub-category</td>
</tr>
<tr>
<td>Transaction_Line_Item</td>
<td>line item</td>
</tr>
<tr>
<td>Transaction FSC</td>
<td>Transaction (Description)</td>
</tr>
</tbody>
</table>

**Cube: IDR Detail Transaction Analysis**

Fact Table: line_item_fact_tbl

Measure(s)
Line Item Amount

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Grain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date</td>
<td>month</td>
</tr>
<tr>
<td>Entity</td>
<td>business unit</td>
</tr>
<tr>
<td>Jurisdiction_City</td>
<td>city</td>
</tr>
<tr>
<td>Tax_Account ID</td>
<td>account ID</td>
</tr>
<tr>
<td>Transaction_Tran ID</td>
<td>transaction ID</td>
</tr>
</tbody>
</table>

**Cube: Payroll**

Fact Table: payroll_fact_tbl

Measure(s)
Payroll

Employee Headcount
Employee Days
Officer Salaries
Cube: Project Financial

Fact Table: projected_financial_fact_tbl
Measure(s)
Projected Book Net Income
Projected Permanent Differences
Projected Timing Differences

Cube: Project Taxable 1

Fact Table: projected_financial_fact_tbl
Measure(s)
Projected Book Net Income
Projected Permanent Differences
Projected Timing Differences

Cube: Project Taxable 2
Fact Table: schedule_m_fact_tbl
Measure(s)
Schedule M Amount

5 Dimension                Grain
  Date                     month
  Entity                   business unit
  Tax_Subcategory          tax subcategory

10 Cube: Property Tax Comparison

Fact Table: asset_fact_tbl
Measure(s)
Accumulated Depreciation
15 Assessed Value
Book Basis
Capacity
Replacement Cost
Utilization

20
  Dimension                Grain
  Asset                    asset ID
  Date                     month
  Entity                   business unit
  Jurisdiction_Geo Code    geo code
  Tax_Account ID           account ID

25 Cube: Purchase Transactions Analysis

30 Fact Table: invoice_fact_tbl
Measure(s)
Pretax Invoice
Taxes Paid
Invoice Total

Dimension 
Date 
Entity_Company 
Jurisdiction_Geo Code 
Account ID 
Transaction ID_Decsp 

Grain 
month 
Company 
geo code 
account ID 
Transaction Description

10 

Cube: Reserve Account Transaction Analysis

Fact Table: line_item_fact_tbl 
Measure(s) 
Line Item Amount 

Dimension 
Date 
Entity 
Jurisdiction_City 
Tax_Account ID 
Transaction Description 
Transaction ID 
Transaction Type 

Grain 
month 
business unit 
city 
account ID 
Transaction Description 
Transaction ID 
Transaction Type

15

20

Cube: Sales Tax Refund

Fact Table: invoice_fact_tbl 
Measure(s) 
Invoice Total 

Pre-tax Invoice Amount 
Exempt Amount 
Tax Paid 
Tax Accrued 

30
Dimension: Grain
Date: month
Entity: business unit
Tax_Subcategory: tax subcategory
Vendor ID: vendor ID
Invoice Number: Invoice Number

Cube: Sales Tax Refund Analysis

Fact Table: invoice_fact_tbl
Measure(s)
Exempt Base Amount
Invoice Amount
Pre-tax Invoice Amount
Tax Accrued
Tax Paid

Dimension: Grain
Date: month
Entity: business unit
Tax_Account ID: account ID

Cube: Schedule M Comparison

Fact Table: schedule_m_fact_tbl
Measure(s)
Schedule M Amount

Dimension: Grain
Date: year
Entity: business unit
Tax_Sub Category: sub-category
Cube: Tax Adjustment

Fact Table: tax_adjustment_fact_tbl

Measure(s)

Tax Adjustment Amount

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Grain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date</td>
<td>year</td>
</tr>
<tr>
<td>Entity</td>
<td>business unit</td>
</tr>
<tr>
<td>Jurisdiction_Type</td>
<td>jurisdiction name</td>
</tr>
<tr>
<td>Tax_Sub Category</td>
<td>sub-category</td>
</tr>
</tbody>
</table>

Cube: Tax Return Analysis

Fact Table: tax_return_fact_tbl

Measure(s)

Tax Return Amount

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Grain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date</td>
<td>year</td>
</tr>
<tr>
<td>Entity</td>
<td>business unit</td>
</tr>
<tr>
<td>Jurisdiction_Type</td>
<td>jurisdiction type</td>
</tr>
<tr>
<td>Tax_Sub Category</td>
<td>sub-category</td>
</tr>
</tbody>
</table>

Cube: Three Year Schedule M Comparison

Fact Table: schedule_m_fact_table

Measure(s)

Schedule M Amount

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Grain</th>
</tr>
</thead>
</table>
Date           year
Entity         business unit
Tax_Account ID Account ID (Description)

**Cube: Virtual Reserve Account Tax Class**

Fact Table:  line_item_fact_tbl
Measure(s)
Line Item Amount

Dimension      Grain
Tax_Account ID Account ID (Description)

**Virtual Cubes**

**Cube: Book to Tax Reconciliation**

Fact Tables:  tax_return_fact_tbl, schedule_m_fact_tbl, adjusted_book_fact_tbl
Measure(s)
tax_return_fact_tbl.Tax Return Amount
schedule_m_fact_tbl.Schedule M Amount
adjusted_book_fact_tbl.Adjusted Book Amount
tax adjustment.Tax Adjustment Amount

Dimension      Grain
Date            month
Entity          business unit
Jurisdiction_Type jurisdiction type
Tax_Sub Category sub-category

**Cube: Projected Taxable Income**

Fact Tables:  projected_financial_fact_tbl, schedule_m_fact_tbl
Measure(s)
Projected taxable income 1. Projected Book Net Income
Projected taxable income 1. Projected Permanent Differences
Projected taxable income 1. Projected Timing Difference

Projected taxable income 2. Schedule M Amount

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Grain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date</td>
<td>month</td>
</tr>
<tr>
<td>Entity</td>
<td>business unit</td>
</tr>
<tr>
<td>Jurisdiction_City</td>
<td>jurisdiction type</td>
</tr>
<tr>
<td>Tax_Sub Category</td>
<td>sub-category</td>
</tr>
</tbody>
</table>

**Cube: Projected Taxable Income (FIX)**

Fact Tables: Reserve Account Transaction Analysis
Measure(s)
Reserve Account Transaction Analysis . Line Item Amount

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Grain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date</td>
<td>month</td>
</tr>
<tr>
<td>Entity</td>
<td>business unit</td>
</tr>
<tr>
<td>Jurisdiction_City</td>
<td>jurisdiction type</td>
</tr>
<tr>
<td>Tax_Account ID</td>
<td>Account ID (Description)</td>
</tr>
</tbody>
</table>

Although only a few embodiments of the present invention have been described in detail herein, it should be understood that the present invention may be embodied in many other specific forms without departing from the spirit or scope of the invention. Therefore, the present examples and embodiments are to be considered as illustrative and not restrictive, and the invention is not to be limited to the details given herein, but may be modified within the scope of the appended claims.
CLAIMS

What is claimed is:

1. A method for organizing multiple dimensions of tax-related data for generating fact
details comprising the steps of:
   (a) providing a plurality of dimensions of tax-related data;
   (b) providing a plurality of attributes for each of the dimensions;
   (c) receiving a plurality of entries associated with the attributes of the dimensions;
   (d) generating a plurality of fact details by aggregating the entries of a predetermined set of
    the attributes of the dimensions; and
   (e) outputting the fact details.

2. The method as set forth in claim 1, and further comprising the step of:
   providing a description of each of the attributes.

3. The method as set forth in claim 1, and further comprising the step of:
   providing a description of each of the dimensions.

4. The method as set forth in claim 1, and further comprising the step of:
   providing a description of each of the fact details.

5. The method as set forth in claim 1, wherein the dimensions are selected from the group of
dimensions consisting of asset, date, entity, jurisdiction, tax classification, and
transaction.

6. The method as set forth in claim 1, wherein the fact details are selected from the group of
fact details consisting of an adjusted book fact, a tax adjustment fact, a schedule M fact, a
tax return fact, an apportionment fact, a line item fact, a transaction fact, an invoice fact,
an asset fact, a payroll fact, a taxes paid fact, and a projected financial fact.

7. A system for organizing multiple dimensions of tax-related data for generating fact
details comprising:
(a) logic that provides a plurality of dimensions of tax-related data;
(b) logic that provides a plurality of attributes for each of the dimensions;
(c) logic that receives a plurality of entries associated with the attributes of the dimensions;
(d) logic that generates a plurality of fact details by aggregating the entries of a predetermined set of the attributes of the dimensions; and
(e) logic that outputs the fact details.

8. The system as set forth in claim 7, and further comprising logic that provides a description of each of the attributes.

9. The system as set forth in claim 7, and further comprising logic that provides a description of each of the dimensions.

10. The system as set forth in claim 7, and further comprising logic that provides a description of each of the fact details.

11. The system as set forth in claim 7, wherein the dimensions are selected from the group of dimensions consisting of asset, date, entity, jurisdiction, tax classification, and transaction.

12. The system as set forth in claim 7, wherein the fact details are selected from the group of fact details consisting of an adjusted book fact, a tax adjustment fact, a schedule M fact, a tax return fact, an apportionment fact, a line item fact, a transaction fact, an invoice fact, an asset fact, a payroll fact, a taxes paid fact, and a projected financial fact.

13. A computer program embodied on a computer readable medium for organizing multiple dimensions of tax-related data for generating fact details comprising:
(a) a code segment that provides a plurality of dimensions of tax-related data;
(b) a code segment that provides a plurality of attributes for each of the dimensions;
(c) a code segment that receives a plurality of entries associated with the attributes of the dimensions;
(d) a code segment that generates a plurality of fact details by aggregating the entries of a predetermined set of the attributes of the dimensions; and
(e) a code segment that outputs the fact details.

14. The computer program as set forth in claim 13, and further comprising a code segment that provides a description of each of the attributes.

15. The computer program as set forth in claim 13, and further comprising a code segment that provides a description of each of the dimensions.

16. The computer program as set forth in claim 13, and further comprising a code segment that provides a description of each of the fact details.

17. The computer program as set forth in claim 13, wherein the dimensions are selected from the group of dimensions consisting of asset, date, entity, jurisdiction, tax classification, and transaction.

18. The computer program as set forth in claim 13, wherein the fact details are selected from the group of fact details consisting of an adjusted book fact, a tax adjustment fact, a schedule M fact, a tax return fact, an apportionment fact, a line item fact, a transaction fact, an invoice fact, an asset fact, a payroll fact, a taxes paid fact, and a projected financial fact.
START

150

Providing a plurality of dimensions of tax-related data to be submitted in a tax return

152

Providing a plurality of attributes for each of the dimensions

154

Receiving a plurality of entries associated with the attributes of the dimensions

156

Generating a plurality of fact details by aggregating the entries of a predetermined set of the attributes of the dimensions

158

Outputting the fact details for submission with the tax return

160

DONE

162

Figure 1A
Figure 3
Figure 4
Figure 6
Figure 8
Figure 9
Figure 10
Figure 11
Figure 12
Figure 13
Figure 14
Figure 17
Figure 18
Figure 20