ELECTRONIC BILL CREATION AND PRESENTMENT SYSTEM

BACKGROUND OF THE INVENTION

Electronic publishing of bills or "electronic bill presentment" is experiencing explosive growth due, in large part, to increased use and reliance on the Internet. Many companies desire to venture into the field of electronic bill presentment. However, they are often dissuaded, due to the complexities involved in creating and operating an electronic bill presentment system.

Bill presentment is not a simple task, since the data, which is ultimately displayed in electronic format, must be queried from a database server, processed and formatted dynamically. Moreover, prior to presentment, various pre-presentment processing tasks must be performed, including among other things, extracting data from hard copies of bills and storing the associated data in a format that can be later queried for presentment.

Another reason bill presentment is complex is that bills are customer-specific. For example, a bill may range from 1 to 10000 or more records and each record may range from 1 to 100 items. This requires a database structure that is flexible enough to accommodate the possible ranges of records and underlying items. As an example, consider a telephone bill. A customer may be a subscriber to a variety of telephone services, e.g. call waiting, conference calling, phone messaging, internet-related services, and so on. For each service, there may be different itemized records of time of use, telephone numbers called, charges etc.

A limiting feature of prior art electronic billing presentment systems, however, is that database schema are fixed to predefined data base fields. This "fixed schema" approach restricts the billing presentment system to industry-specific bill schema templates. In other words, separate billing presentment systems must be separately and specifically designed for each billing community or industry, e.g. telco, utility, insurance etc.

Additional problems may be encountered if a bill is to be presented over the Internet. Because an electronic bill, displayed by a browser, is not simply a pure HTML page, but is embedded with logic, the bill data is dynamic. In other words, the interface to the bill changes dynamically with the data. Consider the telephone bill example above. Because each customer will have made a different number of calls

during a particular billing period, the display of a particular customer's bill may appear unsightly or even unreadable if, for example, the number of calls made is greater than the display format allows. Hence, flexibility of display is needed in the HTML page. Prior solutions require the expertise of, for example, a scripting language programmer. This is undesirable from a user's perspective, particularly for users who lack programming know-how.

SUMMARY OF THE INVENTION

In an exemplary embodiment of the present invention, an electronic bill generator is provided. The system comprises an open schema database generator configured to receive a bill in a format to be determined and configured to create a user-defined database table having a plurality of fields, a database server configured to receive and store the database table, and a bill data processor having a data pump for pumping billing data into the database table fields to create an electronic bill.

In another exemplary embodiment of the invention, an electronic bill is provided, by: providing a bill having a plurality of bill fields containing bill data, selecting tags from a list of tags to identify fields of a first portion of the plurality of bill fields, defining new tags to identify fields of a second portion of the plurality of bill fields, using the selected and defined tags to create a database structure in a database; extracting the bill data from the bill fields, and pumping the bill data into the database structure.

In yet another embodiment of the invention, an electronic bill creation and presentment system comprises: a tags generator configured to create a database structure by associating a set of tags with a set of bill fields, a data mapper configured to define mapping rules for extracting billing data from a bill and pumping the billing data into the bill fields represented by the tags, a bill template generator configured to create an HTML-formatted bill template, and an electronic bill presenting engine configured to display the extracted billing data as rendered through the bill template.

A further understanding of the nature and advantages of the inventions herein may be realized by reference to the remaining portions of the specification and the attached drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a block diagram of a bill creation and presentment system according to an exemplary embodiment of the present invention.

FIG. 2 is a flowchart illustrating a method for creating the database and XML tags by the tags generator of FIG. 1 according to an exemplary embodiment of the present invention.

FIG. 3 is an exemplary illustration of a GUI to the tags generator shown in FIG. 1.

FIG. 4 is a diagram illustrating the database structure of the database created by the tags generator shown in FIG. 1.

FIG. 5 is an exemplary illustration of a supertemplate.

FIG. 6 is a flowchart illustrating the mapping process performed by the data mapper shown in FIG. 1, starting with the mapping process and finishing with the generation of a data extraction program.

FIG. 7 shows an exemplary illustration of a GUI to the data mapper shown in FIG. 1.

FIG. 8 is a flowchart illustrating bill data processing by the XML mapper shown in FIG. 1.

FIG. 9 is a flowchart illustrating a process of pumping data into the database server using the data pump shown in FIG. 1.

FIG. 10 is a flowchart illustrating a process for generating SQL statements.

FIG. 11 shows an exemplary illustration of a GUI to the SQL generator shown in FIG. 1.

FIG. 12 is a flowchart illustrating the bill generation process by the Bill generator shown in FIG. 1.

FIG. 13 shows an exemplary illustration of a GUI to the Bill generator shown in FIG. 1.

FIG. 14 is a flowchart illustrating a method for use by a user to preview a generated bill using a browser.

FIG. 15 is an illustration of a sample bill in text format.

FIG. 16 is an illustration of the table structure (bill schema) for the sample bill shown in FIG. 15.

FIG. 17 is a sample bill template with XML tags for the sample bill shown in FIG. 15.

FIG. 18 is an illustration of the sample bill shown in FIG. 14 as might be displayed within a browser.

DESCRIPTION OF THE SPECIFIC EMBODIMENTS

Generally, the present invention provides an integrated bill creation and presentment a system and method for creating and presenting user-specific electronic bills. Bills of any print format can be accepted and a data conversion server is provided to convert a non-text bill file, e.g., a bill in Advanced Function Presentation (AFP) format, to a text file. The creation and presentment system and method of the present invention provide a novel "open schema" approach, which allows a user to create electronic bills characterized by any specified schemas. This open schema aspect allows the creation and presentment of bills from any industry and of any format and is, therefore, more versatile than the "fixed schema" approaches of the prior art which restrict billing presentment systems to industry-specific bill schema templates.

Bill creation in the present invention is performed using any preferred HTML editor such as the NetObjects or Fusion editors, thereby allowing the user to design the bill graphically. Tags, for example, XML (Extended Markup Language) tags, are inserted at specific locations in the bill to represent bill information, e.g., Bill Number, Bill Date, Account Number, etc. Bills are extracted from the database server via the tags and are rendered through a bill template. Bill data, stored in the database server, is displayed according to logic embedded inside web pages with the tags when the web pages are invoked. By utilizing XML standards, the system and method of the present invention provide users with a high-performance framework for component-based page assembly from multiple data sources, including relational data such as DB2 and Oracle. Finally, the invention provides for check-in and checkout of templates, keeps track of versions and provides for remote check-out and check-in of bill templates from the Web.

More specifically now, there is shown in FIG. 1 an exemplary embodiment of a system 10 that creates and presents an electronic bill according to the present invention. Referring to FIG. 1, there are three primary processing modules: Database and XML tags creation processing module 100, Bill data processing module 104 and bill template processing module 112. All three of these processing modules may reside on a single database server 103, which a user, typically a biller, gains access to via a computer workstation 101.

Database and XML tags creation processing module 100 includes a tags generator 102, which preferably uses a graphical user interface (GUI) so that a user can easily select which XML tags are needed to construct the content of an electronic bill. tags generator 102 simplifies bill schema creation since the user does not need to type in SQL statements to create a table. Rather, by using a graphical interface, the user need only define the table and its fields to generate a data structure.

Bill data processing module 104 includes data mapper 106, XML mapper 108 and data pump 110. Data mapper 106 is preferably implemented with a GUI, which enables the user to easily define mapping rules to convert any billing format of any bill to a format consistent with the data structure in database server 103. XML mapper 108 is similar to data mapper 106, yet differs because it is specially designed to extract bill data from a bill formatted in XML. The ability to define mapping rules allows any bill type to be normalized to a format that can be processed by the system 10. Data Pump 110 uses the rules file created by data mapper 106 to pump the billing data into the database server 103.

Bill template processing module 112 comprises SQL generator 114 and Bill generator 116. SQL generator 114 assists the user in generating SQL statements, preferably via a GUI. Bill generator 116 provides the user with tools for modifying the interface for bill templates.

A more detailed description of the elements within each of the three primary processing components 100, 104, 112 is provided below. First, however, a description of bill data types and tables comprising the database data structure is presented.

All data in a given bill can be categorized as one of two types of data, "header" or "detail." Header data is data having a one-to-one correspondence with its descriptor. For example:

Name of the customer:	John Smith
Account number:	777121
Reference number:	158796315

By contrast, detail data is characterized by a plurality of bases. For example, detail data normally comprises several rows and columns and is presented in a table form as shown in TABLE 1. In TABLE 1, "Date," "Item" and "Amount" are the detail data.

Date	Item	Amount
1999-07-11	CPU	$ 1000.00
1999-08-01	Keyboard	$ 500.00
1999-09-09	Monitor	$ 2000.00

TABLE 1.

Referring now to FIG. 2, there is shown a flowchart illustrating a method of creating the database structure and tags by tags generator 102. To create tables and tags, the user, at step 200, first logs in to gain access to the system.

At step 202, once login has been authorized, the user selects a table type, i.e., Header table or Detail table. The selected table stores the header or detail data.

At step 204, the user enters a description of the selected table and at steps 206 and 208 the user proceeds to select the fields for the table. To select the fields for a particular table, a range of available field names is provided (step 206), via a GUI, for example. However, the user is provided with the option to add new field names if so desired (step 208).

At step 210, the user defines a primary reference key for the table and then at step 212, the table is saved. The above process is repeated until all tables have been defined.

At step 214, after all tables have been defined, the user defines a master table. The master table is one of the header tables. All the primary reference keys are joined to formulate an internal reference value. To complete the table creation process in the database, a bill summary table, at step 216, and a foreign key or link key, at step 218, are defined. The summary table contains fields that represent the summary of the bill and act as a link between customers, account numbers and bills. An extra field (column) is added to each table to store the internal reference value obtained from the master header table. It is used to link the totality of header and detail tables together. At step 220 the table creation process is performed and the created table is stored in server 103. In some embodiments the user can recall previous settings, modify or delete existing fields, or add new fields. As explained above, preferably the user interface to tags generator 102 is a GUI. FIG. 3 shows an exemplary GUI to tags generator 102.

Referring now to FIG. 4, there is shown a diagram of the database structure of the database created by tags generator 102. The database structure can have

different schemas, 302A, 302B 302C, etc. Accordingly, each schema 302A, 302B 302C, etc. represents a type of bill.

Tables under each schema 302A, 302B 302C, etc. can be categorized into three table types, i.e., Header Table 304, Detail Table 306, referred to as H Table and D Table, respectively, and Bill Summary Table 310. The H Table and D Table are created so that they are synchronized with the bill data characteristics. H and D table names are preferably HXXX and DXXX, respectively, where XXX is a table counter. So, for one bill, it can have header and detail tables, both ranging from 001 to 999. Naming tables via use of a table counter simplifies control of the database structure and avoids duplicate tables. Each field in a particular table will have the name of that table in front of it, for example, H001_ACCOUNT or D002_DATE.

Once all the tables have been created, the user proceeds to extract bill data using data mapper 106 (See FIG. 1), which first determines and obtains the position of data in a bill using a supertemplate and then creates an executable data extraction program, which extracts data from the bills. The supertemplate is a text template provided by the user and contains all the possibilities of how data can appear in a single bill. An exemplary supertemplate is shown in FIG. 5. The user can introduce the supertemplate to the system 10 using a variety of transfer modes, for example, lease-line, dial-up, ftp, replication, etc. All bills that have the same format as the supertemplate are subsets of the supertemplate.

In an exemplary embodiment, the extracted data is stored as a text file in delimited format. The same extraction program can be used on bills that have the same format as the supertemplate, or are subsets of the supertemplate. One of the advantages of data mapper 106 is that data from any bills created in the future, which happen to have the same format as the supertemplate, can be operated on by the same extraction program.

In the mapping process, the user sets indicators such as "Page Start" and "Trigger" to tell the system what to do when the indicator appears in the text file. Table 2 lists some exemplary indicators and function names used by data mapper 106.

Indicator	Description
Page Start	A unique string at a particular position in the text file to indicate a starting point of the text file for that particular page format.
Trigger	A unique string at a particular position in the text file to indicate a reference point in the text file for mapping header and detail data.
Start Repeat	Using Trigger(s) as condition to indicate a starting point of a detail data mapping process.
End Repeat	Using Trigger(s) as condition to indicate an ending point of a detail data mapping process.

TABLE 2.

Referring now to FIG. 6, there is shown a flowchart of the operations performed by data mapper 106, starting with the mapping process and ending with the generation of an executable data extraction program. At step 500, a supertemplate is loaded. Next, at step 502, the user sets the "Page Start" indicator to define a starting point in the text file for which data is to be extracted. At step 504, the user either starts mapping the header data or sets the "Trigger" indicator to indicate a reference point in the text file for mapping header and detail data. The user can map the header data using "Page Start" or "Trigger" as the reference point. However, in order to start mapping the detail data, the user must first, at step 506, set the "Start Repeat" condition. Finally, at step 510, the user sets the "End Repeat" condition to complete the detail data mapping process.

At step 512, a rules file is generated, based on the position of the header data, detail data, "Page Start," "Trigger", "Start Repeat" and "End Repeat" indicators. At step 514, the rules file is used to generate a source file for the data extraction program, which can be, for example A C++ program source file, or source code in another suitable programming language. Finally, at step 516, the source file is compiled to produce an executable data extraction program. As explained above, preferably the user interface to data mapper 106 is a GUI. FIG. 7 shows an exemplary GUI to data mapper 106.

As mentioned above, XML mapper 108 is similar to data mapper 106, with a primary distinction being that XML mapper 108 was designed to extract bill data from XML-formatted bills. Accordingly, because some companies save their bills in XML format, XML mapper 108 is included so that data can be extracted from XML-formatted files.

Referring to FIG. 8, there is shown a flowchart illustrating the operations performed by XML mapper 108, starting with the mapping process and ending with the generation of an executable data extraction program, which can operate on XML-formatted files. Initially, at step 600, the user selects a Document Type Definition (DTD) file. A DTD file is a text file that describes what tags and attributes are valid in an XML document and also in what contexts they are valid. In this way, a relationship is formed between the selected DTD file and the database. In step 602, the user selects a database structure file, after which all the elements and attributes are listed. Next the user chooses whether to define one or more Field tags (step 606) or set the "Start Repeat," "Repeat Field" and "End Repeat" tags (steps 608, 610, 612).

As with data mapper 106, the user sets indicators when using XML mapper 108 to inform the extraction program what to do when an indicator is encountered in the text file. The indicators used in XML mapper 108 are almost the same as data mapper 106. For example "Bill" is equivalent to "Page Start' in data mapper 106. Other indicators are "Field', "Start Repeat', "Repeat Field" and "End Repeat". Table 3 lists the indicators and function names used by XML mapper 108.

Indicator	Description
Bill	XML tag that indicates the start of a new bill.
Field	XML tag that contains the header data
Start Repeat	XML tag that indicates the start of the detail portion
Repeat Field	XML tag that contains the detail data
End Repeat	XML tag that indicates the end of the detail portion

TABLE 3.

In step 614, a rules file is produced, based on the position of XML mapper's 108 indicators and the elements and attributes in DTD, which correspond to the tables and fields in the database. In step 616, these rules are used to generate a source program, which, in step 618, is compiled to produce an executable data extraction program. Each different bill will have its own data extraction program, as the extraction process depends on the XML tags the bill contains and the database structure.

Referring now to FIG. 9, there is shown a flowchart of the process of pumping data into the database by data pump 110. The data extraction program is used by data pump 110 to extract data and pump it into the database. The pumping process begins at step 700 upon execution of the extraction program.

At step 702, a delimited file is produced, in which the data is kept in accordance with the table name. The data extraction program also produces a file containing a list of header and detail tables.

At step 704, data pump 110 reads the tables list file and, subsequently, at step 706, an SQL script prepared. Finally, at step 708, data pump 110 runs the SQL script and imports the delimited files to each header and detail table.

For XML mapper 108, the data extraction program generated is similar to a parser. It will read through the XML file and extract all the data according to the encountered tags. As an example, consider an XML file that contains the following tags and indicators.

<Tag A> -> Bill

<Tag B> -> Field -> Header Field B

<Tag C> -> Start Repeat -> Detail Table C

<Tag D> -> Repeat Field -> Detail Field D

<Tag E> -> End Repeat

When the extraction program hits <Tag A>, it determines that <Tag A> is a "Bill" type tag, indicating that it is a new bill or a new record. When the extraction program hits <Tag B>, it determines that <Tag B> is a 'Field" tag, and the data corresponding to that tag is stored under "Header Field B" in the database. When the extraction program hits <Tag C>, it determines that <Tag C> is a "Start Repeat" tag, and all the data for <Tag C> and <Tag D>, between "Start Repeat" tag <Tag C> and "End Repeat" tag <Tag E> will be stored in Detail Table C and Detail Field D, respectively. This process continues until the entire document is processed.

All the extracted data is saved in a file having a delimited format, after which, data pump 110 pumps the data into the database according to steps similar to steps 704 through 708 (FIG. 9) and as described above in relation to data mapper 106.

After completing Bill data processing module 104, bill template processing module 112 is invoked. The first task of bill template processing module 112 is to generate SQL statements for a bill template so that the data can be successfully retrieved from the database, rendered through the bill template for appropriate presentation. Bill data processing module 104 provides a simple method for a user to change the interface of a bill template by plugging in logic or conditions before generating it into a bill. The logic or conditions are in the form of user-definable scripts, executable programs or procedures. Since the interface changes with the data, the logic that is plugged in consists primarily of SQL statements for retrieving data from the database. Accordingly, Bill data processing module 104 also includes an SQL generator 114. The first task of bill template processing module 112 is to generate SQL statements for the bill template so that data can be successfully retrieved from the database and appropriately presented. Referring to FIG. 10, there is shown a flowchart illustrating the generation of SQL statements by SQL generator 114 according to one embodiment of the present invention. SQL generator 114 eases the bill designer's task as it provides a GUI for the user to create SQL statements by only having to click on a few buttons. FIG. 11 shows an exemplary GUI to SQL generator 114.

At step 800, the user logs in to access SQL generator 114. If, at step 802, the system determines by the login that the user is not authorized, the SQL generator 114 program is terminated. If authorized, at step 804, the H Tables and D Tables are loaded and displayed, preferably in a drop down combo box on the interface. At step 806, the user selects a table for the SQL statement from the drop down combo box. Once a table is selected, at step 808 the fields in the selected table are displayed in a list box below it. The tables and fields are those created through the Tagging and Mapping process described above.

To avoid ambiguity, description of the fields is preferably displayed at the bottom of the interface window. Prior to the next step, the user needs to know the general nature of the SQL statement required. For example, the nature of a "Select All" statement. It is not necessary for the user to create actual SQL statements. At step 810, the user selects the desired fields and, at step 812 the user moves the selected fields to a right list box. If all fields have been selected a "*" or other symbol is preferably

displayed to indicate that all fields have been selected. The user can define multiple conditions for the SQL statement by adding in "where clauses". This is performed, at step 814, by clicking a "Where Clause" button displayed on the interface. At step 816, the values for any where clauses the user adds are moved to a text box. If the user makes any mistakes along the way, the user is provided the opportunity to delete the SQL statement and where clauses.

If a customized SQL is required, the user can key in the statement manually, at step 818, by clicking a "Switch to Customized SQL" button displayed on the interface. Then, at step 820, the selected table is loaded into the left listbox. At step 822, the customized SQL statement is moved to the text box.

Prior to directing SQL generator 114 to generate the SQL statement, the user specifies which type of table the SQL statement is referring to, i.e., whether it is an H table or a D Table. At step 824, the user clicks a "Generate" button on the interface and then at step 826, system 10 detects whether the user has specified whether to save the SQL statement in an H Table, D Table or both. If the user has not properly specified the table, at step 828, the program asks the user to specify the type of table the SQL statement it is referring to. Once the user has properly specified the table, at step 830 the SQL statement is generated and saved in the selected table.

Referring now to FIG. 12, there is shown a flowchart illustrating the bill generation process of Bill generator 116 from a new bill template, an existing bill template from the database server 103 and an unchecked-in bill template from a local directory. At step 900, the user provides login information such as database name, schema, ID and password. At step 902, the login information is verified. If the login information is incorrect, an error is detected at step 904 and the program is terminated. If the login information is correct, a main window of Bill generator 116 is displayed. Preferably, the interface to Bill generator 116 is a GUI. FIG. 13 shows an exemplary illustration of a GUI to Bill generator 116.

Next, at step 906, the user opens a bill template file. The user can choose a totally new HTML bill template with XML tags (step 908), an existing template in the database server 103 (step 910) or an unchecked-in template in a temporary local directory (step 912).

If the user chooses a new bill template, at step 908 the selected bill template is displayed. If the user chooses an existing bill template from the database server 103, at step 910, an existing bill template from the database sever 103 is displayed.

In addition to retrieving the existing bill template at step 910, a definition file, a rules-undo file and a rules-function file, typically known as logic files, are also checked out from the database server 103. These files generally store information of previous modifications. A definition file is a file that stores information from the database; a rules-undo file is a file that stores information needed for the undo operation; and a rules-function file is a file that stores information of functions created during previous modifications. Finally, if the user chooses from a temporary local directory, at step 912 an existing bill template in the local directory will be displayed and at step 914 a definition file, a rules-undo file and a rules-function file are stored as hidden files in the same local directory.

At step 916, Bill generator 116 then plugs in logic or conditions, such as, "Create SQL Functions" to retrieve data from the database, create system functions to call other applications, assign values to variables, perform arithmetic operations and/or add conditional if/else and while functions. Regarding variables, the user must declare that they are to be used in functions by adding the relevant variables into the variable list. The addition and removal of variables are carried out prior to creating functions in step 916.

When plugging in the logic, the logic files are modified. So, upon completing the bill generating process, a new set of files are created at step 918, which consist of the modified bill template and logic files. At step 920, the logic files and bill template are merged and XML tags are converted into the user's definable scripts, executables or procedures at step 922.

At step 924, if the bill generation is successful, a duplicate bill filename will be checked. This is done to avoid redundant bill filenames. If there is no duplicate filename, at step 926, the bill template and logic files are checked into the database server 103. If the check in process fails, at step 914 the bill, bill template and logic files are stored in the local directory.

Referring now to FIG. 14, there is shown a flowchart illustrating a method for use by a user to preview a generated bill using a browser. First, at step 1000, the user logs in to gain access to the user site. At step 1010, the user's ID and password are verified. If either the ID or password are invalid, at step 1012 an error message is displayed and the process is terminated. If the ID and password are valid, at step 1014 the user can request to preview a bill from a selection of available bills. After the user selects the bill to preview, at step 1016 user-definable scripts, executables or procedures

at the web server are invoked and executed. At step 1018, the user-definable scripts, executables or procedures process and then format the processing output of the bill before returning it to the web browser for display. If the user is not satisfied with the bill, the user may check it out again and the whole bill generation process will be repeated.

Figures 15 through 18 are included to provide further illustration of what is seen during the bill generation and presentment process. FIG. 15 shows a sample bill in text format. FIG. 16 shows the sample bill table structures (bill schema) after the table creation process in Database and XML creation processing module 100. FIG. 17 depicts a sample bill template with XML tags. FIG. 18 depicts the sample bill when it is displayed by a browser.

WHAT IS CLAIMED IS:

1. An electronic bill generator, comprising:

an open schema database generator configured to receive a bill having a format to be determined and create a user-defined database table having a plurality of fields;

a database server configured to receive and store the database table; and

a bill data processor having a data pump for pumping billing data into the database table fields to create an electronic bill.

2. The system of claim 1, wherein the bill data processor further comprises:

a data mapper configured to determine the format of the bill and the positions of the billing data in the bill.

3. The system of claim 2, wherein the database server is configured to store a supertemplate, the supertemplate containing all the possibilities of how the bill can be formatted.

4. The system of claim 3, wherein the data mapper is configured to use the supertemplate to determine the positions of the billing data and create a data extraction program to extract the billing data from the bill.

5. The system of claim 4, wherein the data pump is configured to receive the data extraction program and use it to extract the billing data and pump it into the database table.

6. The system of claim 1, wherein the bill data processor further comprises:

an XML mapper for extracting bill data from XML-formatted bills.

7. The system of claim 5, further comprising a bill template processor configured to retrieve billing data from the database table and render it for display through a predetermined bill template, the bill template comprising a user-defined subset of the supertemplate.

8. The system of claim 7, wherein the bill template processor comprises an SQL generator configured to create SQL statements for retrieving billing data from the database table.

9. The system of claim 8, wherein the bill template processor further includes a bill generator that allows the user to change an interface of a bill template by plugging in logic or conditions to affect the generation and display of a bill.

10. The system of claim 1, wherein the bill can be accessed remotely and electronically displayed by a browser.

11. The system of claim 1, further comprising a data conversion processor configured to accept a bill of any print format, the data conversion processor operable to convert the bill format to text.

12. A method of creating an electronic bill, the method comprising the steps of:

providing a bill having a plurality of bill fields containing bill data;

selecting tags from a list of tags to identify fields of a first portion of the plurality of bill fields;

defining new tags to identify fields of a second portion of the plurality of bill fields;

using the selected and defined tags to create a database structure in a database;

extracting the bill data from the bill fields; and

pumping the bill data into the database structure.

13. The method of claim 12, wherein the database structure is configured as one or more tables.

14. The method of claim 12, further comprising the steps of:

determining the position of each bill field in the bill using a supertemplate; and

defining mapping rules for pumping the bill data into the one or more tables.

15. The method of claim 12, further comprising the step of:

generating SQL statements for a bill template retrieved from a database server storing the database structure.

16. The method of claim 12, wherein the bill can be accessed remotely and electronically displayed by a browser.

17. The method of claim 12, wherein prior to the step of providing the bill, converting non-text bill data in the bill fields to a text format.

18. An electronic bill creation and presentment system, comprising:

a tags generator configured to create a database structure by associating a set of tags with a set of bill fields;

a data mapper configured to define mapping rules for extracting billing data from a bill and pumping the billing data into the bill fields represented by the tags;

a bill template generator configured to create an HTML-formatted bill template; and

an electronic bill presenting engine configured to display the extracted billing data as rendered through the bill template.

19. The system of claim 18, further comprising a database server configured to store the database structure and the billing data.

20. The system of claim 18, wherein the bill template generator is configured to accept user-defined logic or conditions to affect the display.

21. The system of claim 18, wherein the electronic bill presenting engine includes a merger engine configured to merge the HTML-formatted bill template with the logic or conditions.