US 20060242160A1

(19) United States

(12) Patent Application Publication (10) Pub. No.: US 2006/0242160 A1

Kanchwalla et al. (43) Pub. Date: Oct. 26, 2006

(54) METHOD AND APPARATUS FOR TRANSPORTING DATA FOR DATA WAREHOUSING APPLICATIONS THAT INCORPORATES ANALYTIC DATA INTERFACE

(76) Inventors: **Firoz Kanchwalla**, Sunnyvale, CA (US); **David Lyle**, Los Gatos, CA (US); **Sujit Bais**, Sunnyvale, CA (US); **Srinivasan Maadapusi**, Sunnyvale, CA (US); **Amol Dongre**, Sunnyvale, CA (US); **Premkumar Somakumar**, Sunnyvale, CA (US)

Correspondence Address:
FENWICK & WEST LLP
SILICON VALLEY CENTER
801 CALIFORNIA STREET
MOUNTAIN VIEW, CA 94041 (US)

(21) Appl. No.: **09/877,370**

(22) Filed: **Jun. 7, 2001**

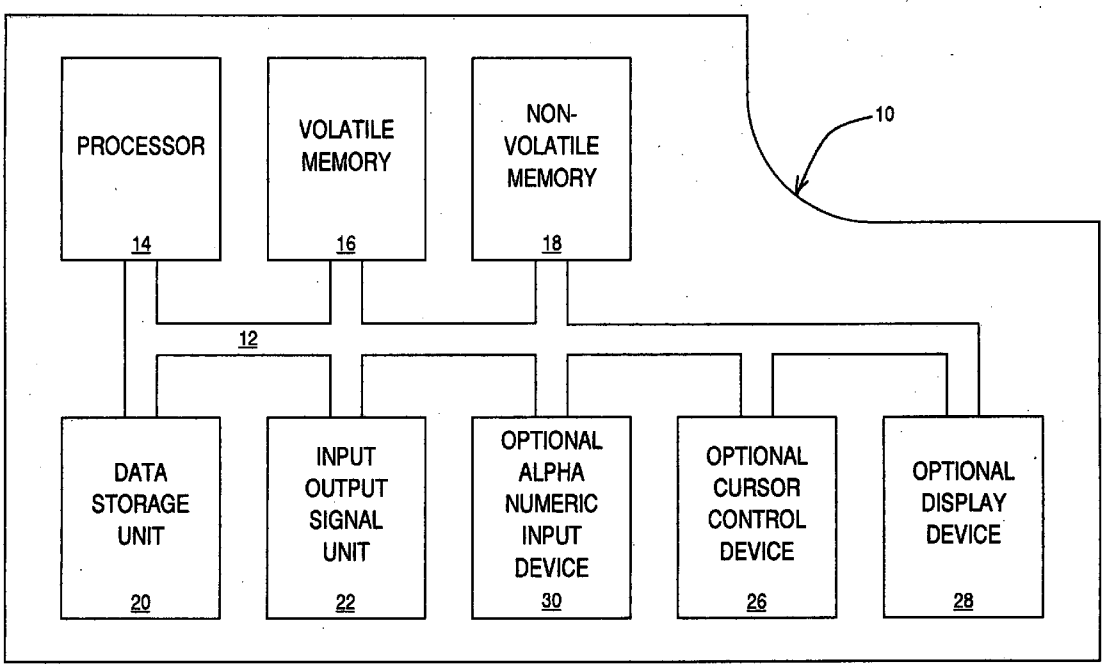
Publication Classification

(51) **Int. Cl.**
G06F 17/00 (2006.01)

(52) **U.S. Cl.** **707/100**

(57) **ABSTRACT**

A method and apparatus for transporting data for a data warehousing application. Data is extracted from one or more source containing data having a standard data structure and is translated into data that contains meaningful business terms. The translated data is then stored. In the present embodiment, an analytic business component is operable for extracting data from the source, translating the extracted data and for storing the translated data into a staging area. The translated data is then processed to obtain data having a common structure. In the present embodiment, a source adapter processes the translated data to obtain data having a common structure. The data having a common structure is then transformed into a format suitable for loading into a data mart. In the present embodiment, an analytic data interface receives the data having a common structure and transforms the data for loading into a data warehouse. The data is then stored in a data warehouse.



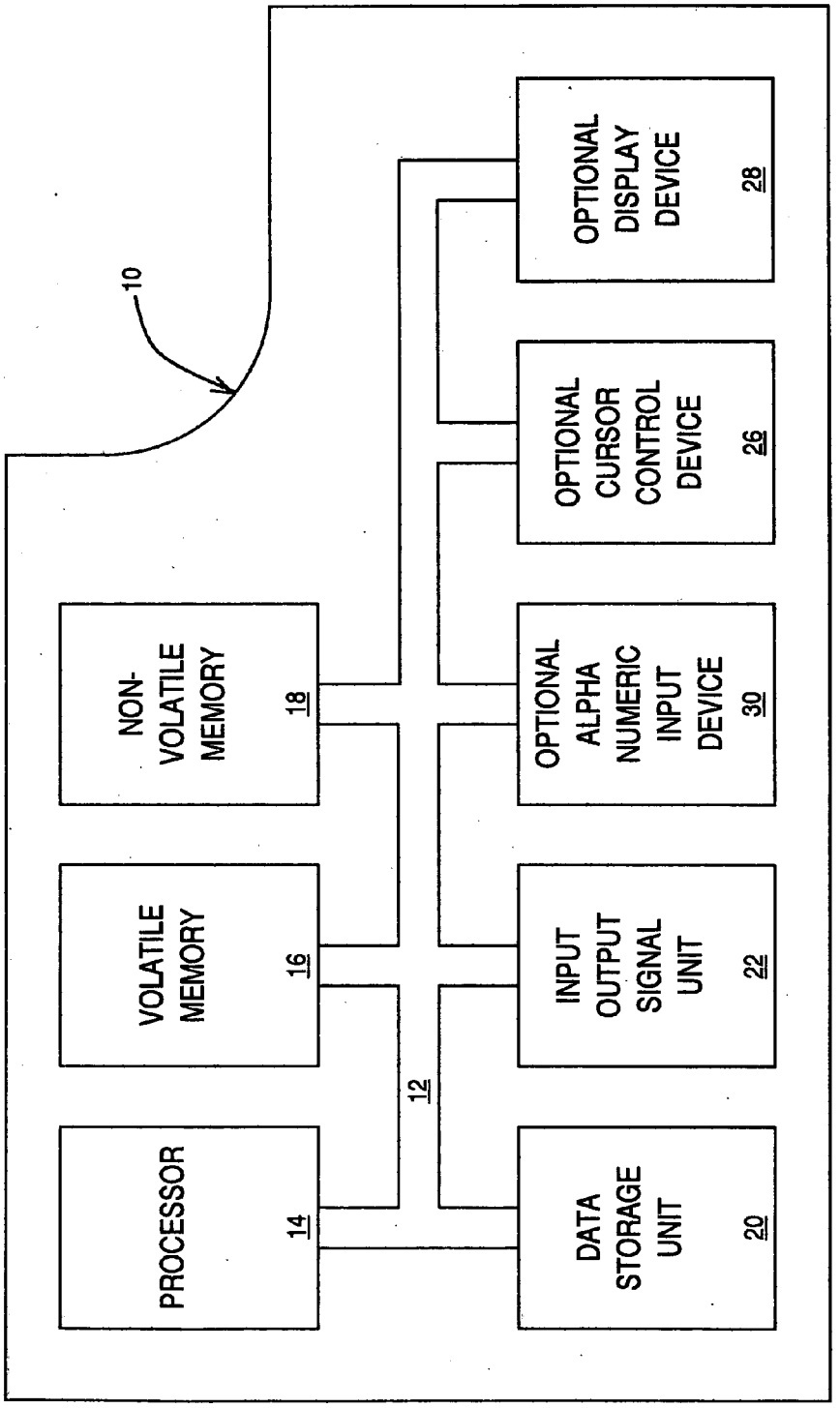

10
PROCESSOR
14
VOLATILE MEMORY
16
NON-VOLATILE MEMORY
18
12
DATA STORAGE UNIT
20
INPUT OUTPUT SIGNAL UNIT
22
OPTIONAL ALPHA NUMERIC INPUT DEVICE
30
OPTIONAL CURSOR CONTROL DEVICE
26
OPTIONAL DISPLAY DEVICE
28

FIG. 1

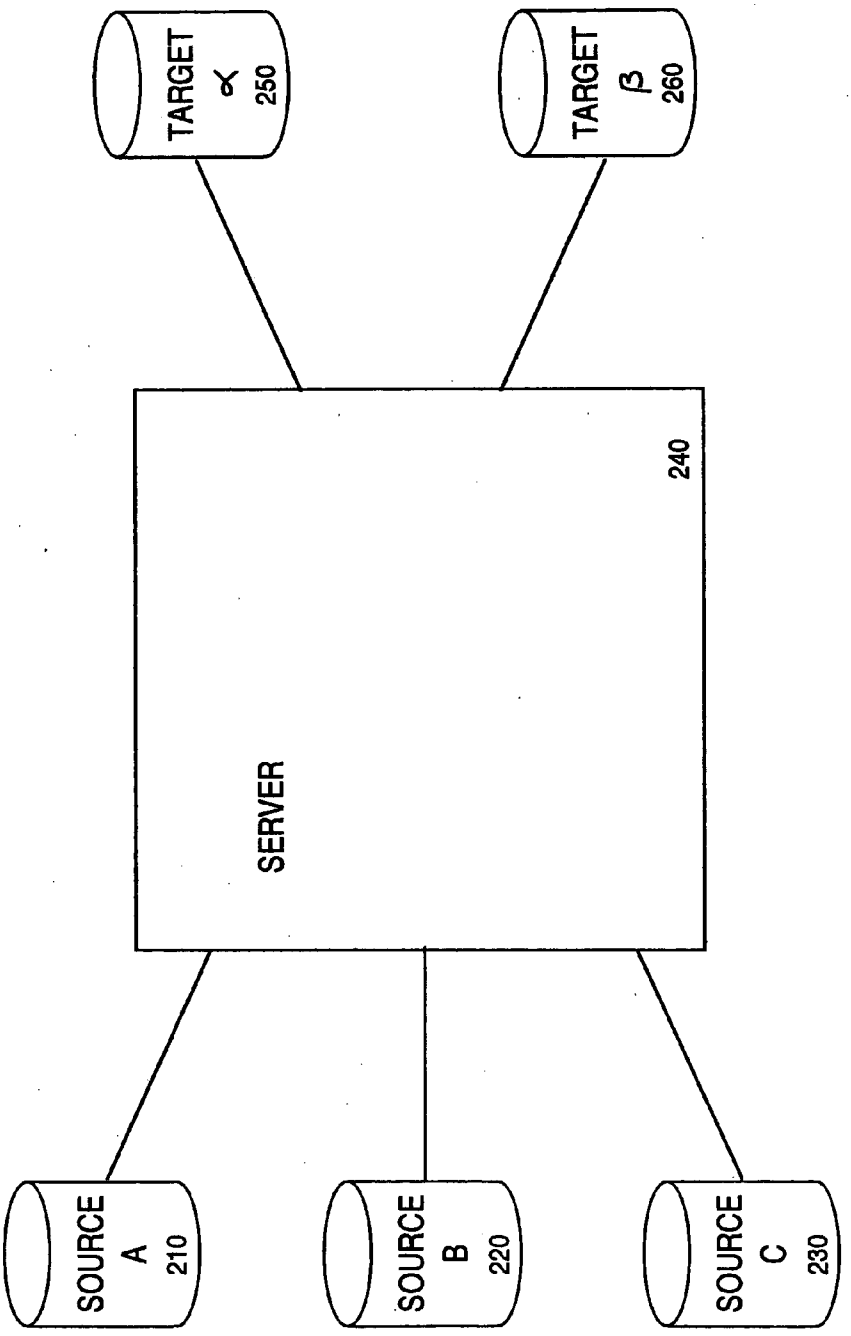

TARGET α 250
TARGET β 260
240
SERVER
SOURCE A 210
SOURCE B 220
SOURCE C 230

FIG. 2

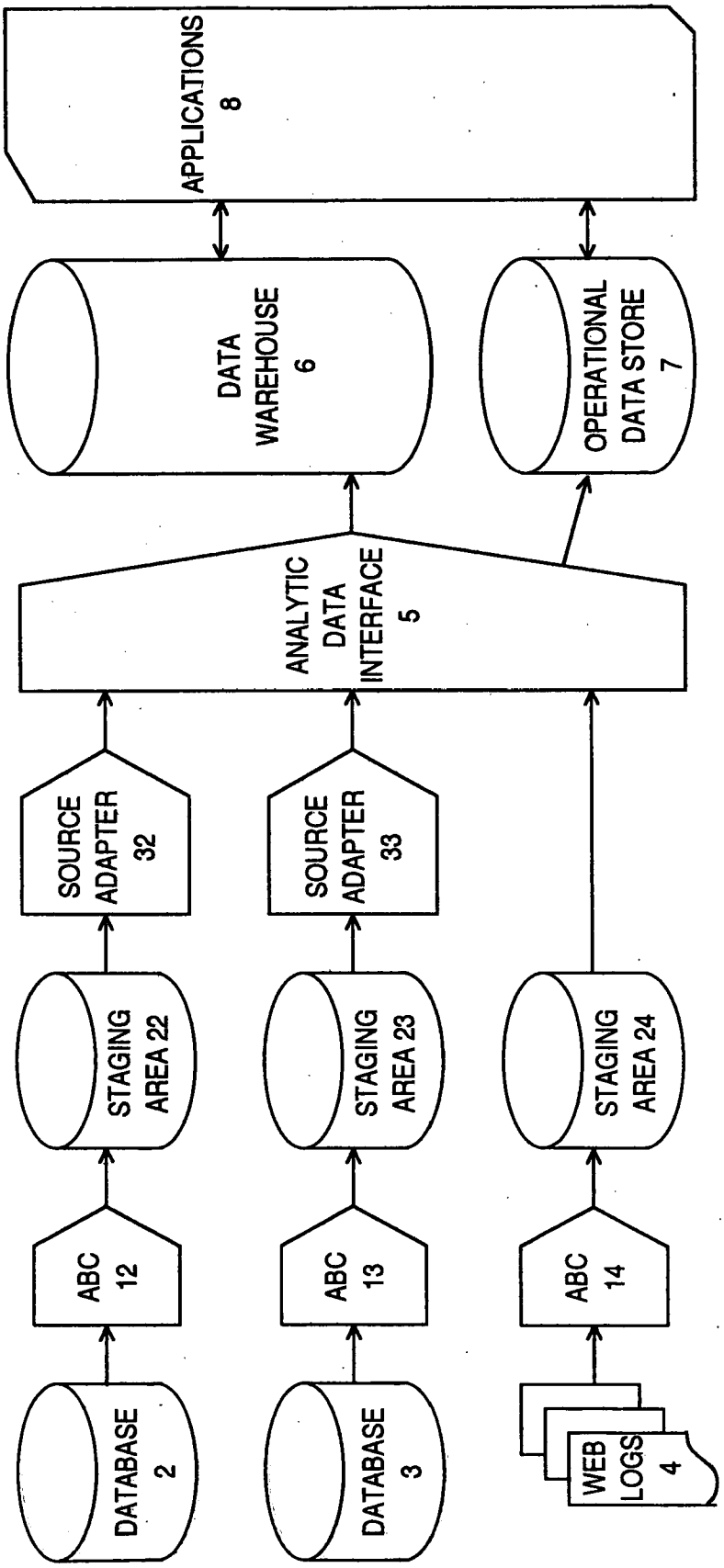

APPLICATIONS
8
DATA
WAREHOUSE
6
OPERATIONAL
DATA STORE
7
ANALYTIC
DATA
INTERFACE
5
SOURCE
ADAPTER
32
SOURCE
ADAPTER
33
STAGING
AREA 22
STAGING
AREA 23
STAGING
AREA 24
ABC
12
ABC
13
ABC
14
DATABASE
2
DATABASE
3
WEB
LOGS
4

FIG. 3

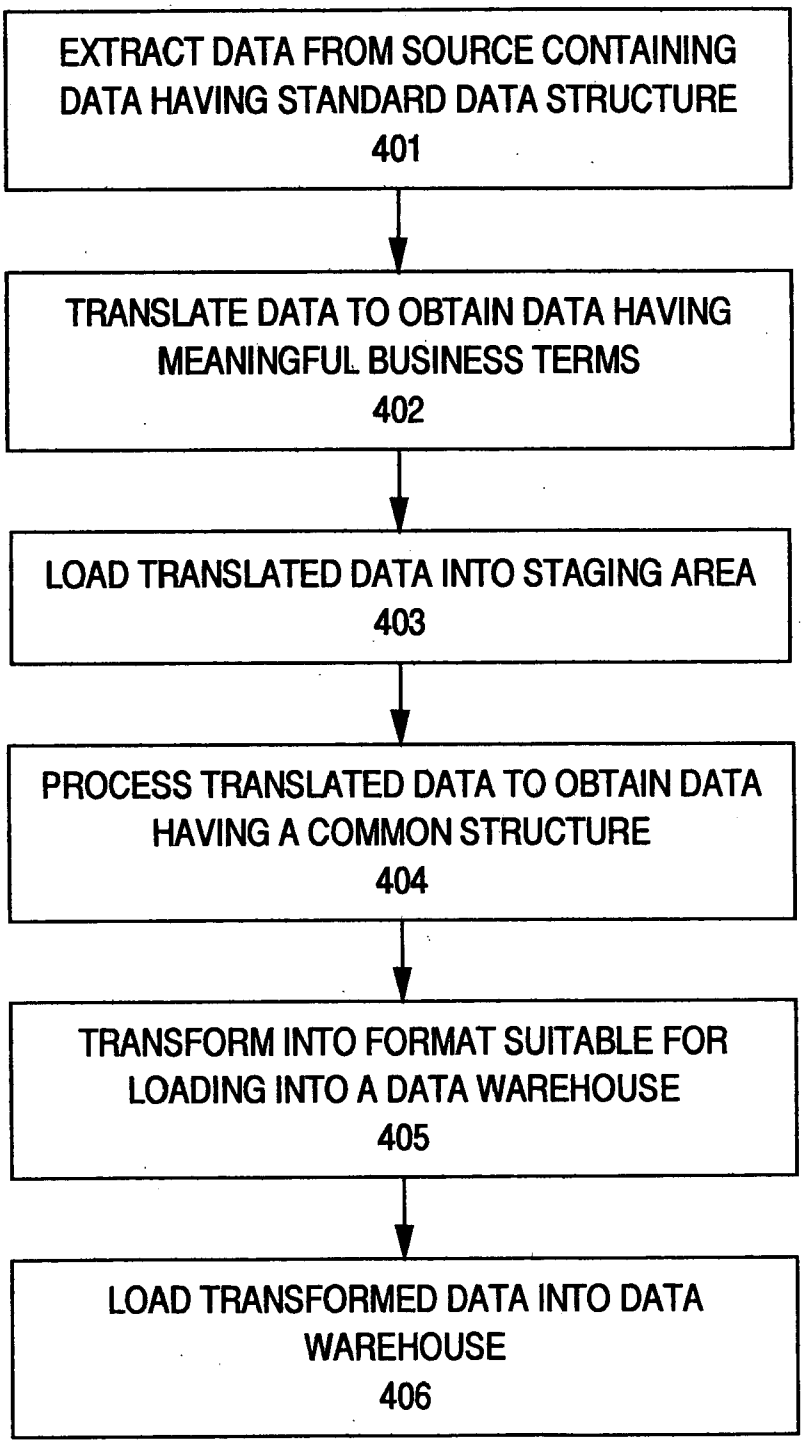

EXTRACT DATA FROM SOURCE CONTAINING DATA HAVING STANDARD DATA STRUCTURE
401
TRANSLATE DATA TO OBTAIN DATA HAVING MEANINGFUL BUSINESS TERMS
402
LOAD TRANSLATED DATA INTO STAGING AREA
403
PROCESS TRANSLATED DATA TO OBTAIN DATA HAVING A COMMON STRUCTURE
404
TRANSFORM INTO FORMAT SUITABLE FOR LOADING INTO A DATA WAREHOUSE
405
LOAD TRANSFORMED DATA INTO DATA WAREHOUSE
406

FIG. 4

METHOD AND APPARATUS FOR TRANSPORTING DATA FOR DATA WAREHOUSING APPLICATIONS THAT INCORPORATES ANALYTIC DATA INTERFACE

FIELD OF THE INVENTION

[0001] The present invention relates to database systems. More particularly, the present invention pertains to an apparatus and method for transporting data for a data warehousing application.

BACKGROUND OF THE INVENTION

[0002] Due to the increased amounts of data being stored and processed today, operational databases are constructed, categorized, and formatted in a manner conducive for maximum throughput, access time, and storage capacity. Unfortunately, the raw data found in these operational databases often exist as rows and columns of numbers and code which appears bewildering and incomprehensible to business analysts and decision makers. Furthermore, the scope and vastness of the raw data stored in modern databases renders it harder to analyze. Hence, applications were developed in an effort to help interpret, analyze, and compile the data so that a business analyst may readily and easily understand it. This is accomplished by mapping, sorting, and summarizing the raw data before it is presented for display. Thereby, individuals can now interpret the data and make key decisions based thereon.

[0003] Extracting raw data from one or more operational databases and transforming it into useful information is the function of data "warehouses" and data "marts." In data warehouses and data marts, the data is structured to satisfy decision support roles rather than operational needs. Before the data is loaded into the target data warehouse or data mart, the corresponding source data from an operational database is filtered to remove extraneous and erroneous records; cryptic and conflicting codes are resolved; raw data is translated into something more meaningful; and summary data that is useful for decision support, trend analysis or other end-user needs is pre-calculated. In the end, the data warehouse is comprised of an analytical database containing data useful for decision support. A data mart is similar to a data warehouse, except that it contains a subset of corporate data for a single aspect of business, such as finance, sales, inventory, or human resources. With data warehouses and data marts, useful information is retained at the disposal of the decision-makers.

[0004] However, establishing a structure for transporting (extracting, transporting and loading) data from an operational database or databases into a structure that can be used for data warehousing applications is quite time consuming. In many instances many months of man-hours are required to define and program a suitable structure for transporting data from an operational database(s) into a format suitable for data warehousing applications.

[0005] The complexities in designing a data model for transporting data from an operational database into target tables in a data warehouse are not simply technical problems. They also involve complex business semantic problems.

[0006] Recently, many operational databases have begun to use standardized database structures. Several companies have recently created Business Application Programming Interfaces for getting data into and out of business databases that use these standardized database structures. Business application programming interfaces are effective for getting information into and out of a business database. However, the user must still perform the process of defining and programming for data transport in order to obtain output that is suitable for use as input to a data warehousing application. This is expensive and time consuming. In addition, these business application programming interfaces require extensive knowledge and programming to learn and use.

[0007] The time and cost for defining and programming such that the data is suitable for use as input to a data warehousing application is particularly problematic for companies that use multiple different operational databases. More particularly, the process of defining and programming for data transport must be repeated for each different operational database. That is, for example, if a company has both a SAP database and an Oracle database, the process of defining and programming for data transport must be performed for both databases and the process is unique to each database.

[0008] What is needed is a method and apparatus that allows for transporting data such that the data can be used in data warehousing applications. In addition, a method and apparatus is needed that meets that above need and that takes advantage of the standardization of database components. Moreover, a method and apparatus is needed that reduces the time required to define and program data transport for data warehousing applications. The present invention provides a method and apparatus that meets the above needs.

SUMMARY OF THE INVENTION

[0009] The present invention includes a method and apparatus for transporting data for a data warehousing application. More particularly, the present invention introduces a method and a data transport process architecture that uses standardized structures of different types of source databases to achieve source-specific configuration for extraction, transformation, and loading in a data warehousing application.

[0010] A system is disclosed that includes an analytic business component that translates operational data from data source having a standardized data structure. The system also includes a staging area for storing the translated data. In addition, the system includes a source adapter that is coupled to the staging area. An analytic data interface couples to the source adapter and receives the data having a common structure and transforms the data for loading into a data warehouse.

[0011] In one embodiment of the present invention, data is extracted from one or more source containing data having a standard data structure and is translated into data that contains meaningful business terms. The translated data is then stored. In the present embodiment, the analytic business component is operable for extracting data from the source, translating the extracted data and for storing the translated data into a staging area.

[0012] The translated data is then processed to obtain data having a common structure. In the present embodiment, a source adapter processes the translated data to obtain data having a common structure.

[0013] The data having a common structure is then transformed into a format suitable for loading into a data mart. In the present embodiment, an analytic data interface receives the data having a common structure and transforms the data for loading into a data warehouse. The data can then be loaded into a data warehouse.

[0014] In the present embodiment, the analytic data interface includes a graphical user interface that makes it easy to configure and customize how business data is loaded into an analytic applications system such as a data warehouse. The analytic data interface includes a simplified abstraction layer for the data warehouse administrator, allowing the warehouse administrator to configure how data is loaded into the analytic applications in a fraction of the time it takes to configure these capabilities programmatically as occurs in prior art systems. In addition, most of the complex technical problems are solved prior to data entering the analytic data interface. In many instances, these technical problems are solved without any required configuration or analysis by the warehouse administrator. This greatly simplifies the task of loading data into a data warehouse, saving significant expense and time.

[0015] The benefits are particularly apparent for companies that use multiple different operational databases. More particularly, there is no need to define and program for data transport for each different operational database. The warehouse administrator needs only define and program for data transport a single time using the graphical user interface of the analytic data interface.

[0016] Accordingly, the present invention provides a method and apparatus that allows for transporting data such that the data can be used in data warehousing applications. In addition, the present invention provides a method and apparatus that takes advantage of the standardization of database components. Moreover, the present invention provides a method and apparatus that reduces the time required to define and program data transport for data warehousing applications.

[0017] These and other objects and advantages of the present invention will no doubt become obvious to those of ordinary skill in the art after having read the following detailed description of the preferred embodiments which are illustrated in the various drawing figures.

BRIEF DESCRIPTION OF THE DRAWINGS

[0018] The accompanying drawings, which are incorporated in and form a part of this specification, illustrate embodiments of the present invention and, together with the description, serve to explain the principles of the invention.

[0019] **FIG. 1** illustrates an exemplary computer system used as part of a data warehousing system in accordance with one embodiment of the present invention.

[0020] **FIG. 2** illustrates an exemplary architecture that includes a server in accordance with one embodiment of the present invention.

[0021] **FIG. 3** illustrates an exemplary architecture that includes an analytic data interface in accordance with one embodiment of the present invention.

[0022] **FIG. 4** shows a method for transporting data to a data warehousing application in accordance with one embodiment of the present invention.

DETAILED DESCRIPTION

[0023] An apparatus and method for transporting data to a data warehousing application is described. In the following description, for purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It will be obvious, however, to one skilled in the art that the present invention may be practiced without these specific details. In other instances, well-known structures and devices are shown in block diagram form in order to avoid obscuring the present invention.

[0024] Some portions of the detailed descriptions that follow are presented in terms of procedures, logic blocks, processing, and other symbolic representations of operations on data bits within a computer memory. These descriptions and representations are the means used by those skilled in the data processing arts to most effectively convey the substance of their work to others skilled in the art. In the present application, a procedure, logic block, process, etc., is conceived to be a self-consistent sequence of steps or instructions leading to a desired result. The steps are those requiring physical manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared, and otherwise manipulated in a computer system. It has proven convenient at times, principally for reasons of common usage, to refer to these signals as bits, values, elements, symbols, characters, terms, numbers, or the like.

[0025] It should be borne in mind, however, that all of these and similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantities. Unless specifically stated otherwise as apparent from the following discussions, it is appreciated that throughout the present invention, discussions utilizing terms such as "extracting,""translating,""loading, ""processing,""transforming,""storing" or the like, can refer to the actions and processes of a computer system, or similar electronic computing device. The computer system or similar electronic computing device manipulates and transforms data represented as physical (electronic) quantities within the computer system's registers and memories into other data similarly represented as physical quantities within the computer system memories or registers or other such information storage, transmission, or display devices.

[0026] With reference to **FIG. 1**, portions of the present invention are comprised of the computer-readable and computer executable instructions which reside, for example, in computer system **10** used as a part of a data warehousing system in accordance with one embodiment of the present invention. It is appreciated that system **10** of **FIG. 1** is exemplary only and that the present invention can operate within a number of different computer systems including general-purpose computer systems, embedded computer systems, and stand-alone computer systems specially adapted for data warehousing applications. Computer system **10** includes an address/data bus **12** for conveying digital information between the various components, a central processor unit (CPU) **14** for processing the digital information and instructions, a main memory **16** comprised of volatile random access memory (RAM) for storing the digital information and instructions, a non-volatile read only

memory (ROM) **18** for storing information and instructions of a more permanent nature. In addition, computer system **10** may also include a data storage unit **20** (e.g., a magnetic, optical, floppy, or tape drive) for storing vast amounts of data, and an I/O interface **22** for interfacing with peripheral devices (e.g., computer network, modem, mass storage devices, etc.). It should be noted that the software program for performing the transport process can be stored either in main memory **16**, data storage unit **20**, or in an external storage device. Devices which may be coupled to computer system **10** include a display device **28** for displaying information to a computer user, an alphanumeric input device **30** (e.g., a keyboard), and a cursor control device **26** (e.g., mouse, trackball, light pen, etc.) for inputting data, selections, updates, etc.

[0027] Furthermore, computer system **10** may be coupled in a network, such as in a client/server environment, whereby a number of clients (e.g., personal computers, workstations, portable computers, minicomputers, terminals, etc.), are used to run processes for performing desired tasks (e.g., inventory control, payroll, billing, etc.).

[0028] **FIG. 2** illustrates an exemplary computer network upon which an embodiment of the present invention may be practiced. Operational databases **210**, **220**, and **230** store data resulting from business and financial transactions, and/or from equipment performance logs. These databases can be any of the conventional RDMS systems (such as from Oracle, Informix, Sybase, Microsoft, Ariba, SAP, Web Logs, etc.) that reside within a high capacity mass storage device (such as hard disk drives, optical drives, tape drives, etc.). Databases **250** and **260** are the data warehouses or data marts that are the targets of the data transportation process.

[0029] In the present embodiment, software operable on server **240** performs data transport operations. That is, in the present embodiment, data from databases **210**, **220**, and **230** is extracted, transformed, and loaded by server **240** into databases **250** and **260**. In the present embodiment, server **240** includes multiple microprocessors and data warehousing related software that operates in conjunction with an installed operating program such as, for example, Windows, NT, UNIX, etc.

[0030] **FIG. 3** shows an embodiment of the present invention that includes Analytic Business Components (ABCs) **12-14** that extract data from sources of data (e.g. databases **2-3** and web logs **4**). Typically, sources contain data in the form of relational tables, Enterprise Resource Planning (ERP) objects, or flat files. Analytic business components **12-14** also translate operational data from each source of data into meaningful business terms. These business terms are then stored in staging areas **22-24**.

[0031] Staging areas **22-24** hold data received from analytic business components **12-14**. In the present embodiment, staging areas **22-24** consolidate data from disparate systems. The staging area denormalizes data where necessary, preparing it for storing in a data warehouse. More particularly, data is cleansed and remains formalized, tables from different databases are joined, and a refresh policy is carried out.

[0032] Continuing with **FIG. 3**, staging areas **22-24** provide a structure that is flexible to allow for user configuration. The only structures that are not configurable are the primary key columns. In the present embodiment, staging areas **22-24** are not organized by subject area, and are not customized for viewing or reporting by end users. Out of the box, the staging area tables consist of all fields required by the analytic data interface **5**, including all extension fields. In one embodiment, the default database for the staging area is oracle. However, at implementation, the database type can be changed.

[0033] The use of staging areas **22-24** provide for quick and efficient data extract. This minimizes the time required for loading the database, allowing for a smaller operational window. In addition, the staging area prepares the data consistently for loading into the analytic data interface from various sources.

[0034] Continuing with **FIG. 3**, source adapters **32-33** convert source-specific terminology into analytic data interface terminology. Source adapters are provided for each source with the exception of web logs, which do not require any further source-specific conversion. In the present embodiment, source adapters **32-33** combine extract-specific staging area objects into the analytic data interface **5**. In the present embodiment, source adapters **32-33** combine staging area objects to match up the inputs and their interrelationship as required by the analytic data interface **5**.

[0035] Analytic data interface **5** transforms data for loading into data warehouse **6** for use in applications **8**. In the present embodiment, the analytic data interface cleans data by enforcing commonalties in dates, names and other data types that appear across multiple systems and prepares it for the source-independent data warehouse.

[0036] In the present embodiment, analytic data interface **5** includes a graphical user interface that makes it easy to configure and customize how business data is loaded into an analytic applications system such as a data warehouse **6**. Analytic data interface **5** includes a simplified abstraction layer for the data warehouse administrator, allowing the warehouse administrator to configure how data is loaded into the analytic applications in a fraction of the time it takes to configure these capabilities programmatically as occurs in prior art systems. In addition, most of the complex technical problems are solved prior to data entering the analytic data interface. Often these technical problems are solved automatically by the preconfigured programming of the analytic business components without any required configuration or analysis by the warehouse administrator. This greatly simplifies the task of loading data into a data warehouse, saving significant expense and time.

[0037] In one embodiment of the present invention, analytic business components **12-14**, source adapters **32-33**, and analytic data interface **5** are implemented as maplets within the warehouse designer. In this embodiment, staging areas **42-43** exists as targets in the warehouse designer.

[0038] An analytic applications system is illustrated that includes data warehouse **6**, operational data store **7** and applications **8**. It is appreciated that the method and apparatus of the present invention is well adapted for transporting data to any of a number of different types of analytic applications systems. For example, the present embodiment in well adapted for transporting data to an analytic applications system that includes a data mart and analytic applications systems having different structures and configurations.

[0039] Continuing with **FIG. 3**, in the present embodiment, data warehouse **6** includes a bus architecture and dimensional data models with conformed dimensions and facts for star schema analysis of data. In addition, data warehouse **6** includes support for slowly changing dimensions with effective dates for type II slowly changing dimensions. Moreover, data warehouse **6** includes pre-packaged dimensions (e.g. time, time of day, IP addresses, etc.), and provision for intelligent extension fields, normalized measures, aggregate tables, and analytic fields with additional complex measures. In the present embodiment, data warehouse **6** is packaged as targets in a warehouse designer.

[0040] Operational data store **7** consolidates and stores references data for loading into data warehouse **6**. In the present embodiment, operational data store **7** retains customized, source-specific-fields that will not exist in data warehouse **6** such as reference data to help standardize other formats (e.g. zip codes, currency conversion rates, and product-code to product-name translations).

[0041] Continuing with **FIG. 3**, applications **8** include data warehousing applications. In the present embodiment applications **8** include pre-packaged analytic tools that provide queries and reports for evaluating business performance such as, for example, applications built on the Power Center™ data integration platform made by Informatica Corporation of Palo Alto, Calif. In the present embodiment, applications **8** include programs that allow for drill-down capability, drill-up capability and drill-across capability. In addition, applications **8** allow for the performance of more detailed analysis using a multi-dimensional approach (slice-and-dice capability). Moreover, in the present embodiment, applications **8** include a user interface with web analytic capabilities that allows for generating standard reports and that includes ad-hoc query tools (that allow for drag and drop of measurements and dimensions to create custom reports).

[0042] In the present embodiment, applications **8** are pre-configured with some aggregate tables on commonly used dimensions and facts. However, these default aggregate tables can be changed based on specific business needs. Aggregate tables contain pre-calculated pre-stored summaries that are stored in the data warehouse to improve query performance.

[0043] **FIG. 4** shows a method **400** for transporting data for a data warehousing application. In one embodiment of the present invention, method **400** is performed by server **240** of **FIG. 2** which includes some or all of the components of computer **1** of **FIG. 1**. In another embodiment, the structure of **FIG. 3** is used to perform method **400**.

[0044] As shown by step **401**, data is extracted from one or more source containing data having a standard data structure. In the embodiment shown in **FIG. 2**, data is extracted from databases **210**, **220**, and **230** by server **240**. In the embodiment shown in **FIG. 3**, analytic business components **12-14** are operable to extract data from data sources **2-4**. More particularly, analytic business component **12** extracts data from database **2**. Similarly, analytic business component **13** extracts data from database **3** and analytic business component **14** extracts data from web logs **4**.

[0045] Data is then translated as shown by step **402**. In one embodiment of the present invention, step **402** is performed by server **240** of **FIG. 2**. In another embodiment, analytic business components **12-14** of **FIG. 3** translate the data extracted in step **401** in order to produce translated data that contains meaningful business terms. More particularly, analytic business component **12** translates data from database **2**. Similarly, analytic business component **13** translates data from database **3** while analytic business component **14** translates data from web logs **4**.

[0046] Continuing with **FIG. 4**, in the present embodiment, analytic business components **12-14** are source-specific. That is, each analytic business component is adapted to extract and translate data from a specific type of database standard data structure. Thus, for example, if database **2** is an Ariba database, analytic business component **12** is an analytic business component for an Ariba database. Similarly, if database **3** is a SAP database, analytic business component **13** is an analytic business component for a SAP database. In an embodiment in which database **4** includes web logs analytic business component **14** is an analytic business component adapted to extract and translate web logs.

[0047] The translation process hides the complexity of the source systems (e.g. databases **2-3** and web logs **4**). In the present embodiment, analytic business components **12-14** perform joins in the data source that help to present the data in simple business terms. In addition, the analytic business components **12-14** present the source fields in form that is understandable to the user. For example, for SAP, business components **12-14** provide English descriptions.

[0048] In the present embodiment, analytic business components **12-14** can be customized at implementation to provide additional business abstractions over and above those business abstractions preprogrammed into analytic business components **12-14**. Moreover, analytic business components **12-14** encapsulate extraction logic as they move data from its source.

[0049] The translated data is then loaded into staging areas as shown by step **403**. In the embodiment of **FIG. 2**, the translated data is loaded into staging areas stored on server **240**. In the embodiment shown in **FIG. 3**, the translated data is loaded into staging areas **22-24**. More particularly, analytic business component **12** loads translated data into staging area **22**. Similarly, analytic business component **13** loads translated data into staging area **23** and analytic business component **14** loads translated data into staging area **24**.

[0050] As shown by step **404**, the translated data is processed to obtain data having a common structure. In the embodiment shown in **FIG. 2**, server **240** is operable to process the translated data to obtain data having a common structure.

[0051] In the embodiment shown in **FIG. 3**, the processing of step **404** converts source-specific terminology into analytic data interface terminology. In the present embodiment, staging area objects are combined to match up the inputs and their interrelationship as required by the analytic data interface **5**.

[0052] Following are several specific examples of the conversion of source-specific terminology that is processed into analytic data interface terminology. It is appreciated that these examples are exemplary and that other source specific

data is processed to be compatible with the inputs of the analytic data interface **5**. Data indicators and data indicator flags are configured based on the data. Physical deletions are performed and a delete flag is set to provide a common way to flag a record to be deleted. In addition, data type conversion and source-related clean up are performed. Also, unique key identification is configured based on the source and its data to take care of problems arising from the fact that the number of keys differ in each source. In the present embodiment, the set of rows that will be put in the data warehouse is also determined.

[**0053**] In the embodiment shown in **FIG. 3**, source adapters **32-33** are operable for processing the translated data within staging areas **22-23** to obtain data having a common structure. More particularly, source adapters **32-34** convert source-specific terminology into analytic data interface terminology. Source adapters are provided for each source with the exception of web logs, which do not require any further source-specific conversion. In the present embodiment, source adapters **32-33** combine extract-specific staging area objects into the analytic data interface **5**. In the present embodiment, source adapters **32-33** combine staging area objects to match up the inputs and their interrelationship as required by the analytic data interface **5**.

[**0054**] Continuing with step **404** of **FIG. 4**, in the present embodiment source adapters **32-33** set the delete flag. More particularly, because each source handles deletes differently, there is a need to provide a uniform delete flag to the analytic data interface. To meet this need, source adapters **32-33** handle physical deletions and delete indicators and provide a common way to flag a record to be deleted.

[**0055**] In the present embodiment source adapters **32-33** handle data type conversions. More particularly, because the same concepts are represented differently in each source, there is a need to provide data type conversion. In the present embodiment, source adapters **32-33** publish the structure of each field and convert the data type using a consistent approach. In addition, source adapters **32-33** handle any source-related clean up.

[**0056**] Continuing with **FIG. 3**, in the present embodiment source adapters **32-33** configure unique key identification based on the source and its data. This takes care of problems arising from the fact that the number of keys differ in each source.

[**0057**] Now referring to **FIG. 4**, the data having a common structure is transformed into a format suitable for loading into an analytic applications system such as a data warehousing application. In the present embodiment, the data having a common structure is transformed into a format suitable for loading into a data warehouse as shown by step **405**. In the embodiment shown in **FIG. 2**, server **240** is operable to transform the data having a common structure into a format suitable for loading into a data warehouse.

[**0058**] In the present embodiment step **405** includes consolidation of business concepts into integrated structures that are suitable for querying and reporting. In addition, source definitions differences are normalized into a single, common definition. In the present embodiment, step **405** includes, for example, code lookup (e.g. currency conversion, unit of measure conversion and code to description field resolution), data-driven updates, intelligent expansion fields, dimension table specific features, fact table specific features, key resolution, key generation, and "bad" data flagging. In the present embodiment, the user can also insert, update, or reject a determination.

[**0059**] In the embodiment shown in **FIG. 3**, analytic data interface **5** is operable to transform data received from source adapters **32-33** and staging area **24** into a format suitable for loading into a data mart. In the present embodiment, analytic data interface **5** provides complex transformation logic that integrates data in two ways. First, business concepts are consolidated across an entire value chain into integrated structures that are suitable for querying and reporting. In addition, the analytic data interface **5** provides complex transformation logic that normalizes source definitions differences into a single, common definition. Some of the transformation logic performed in the analytic data interface **5** include codes lookup (e.g. currency conversion, unit of measure conversion and code to description field resolution) and data-driven updates. In addition, intelligent expansion fields that extend out of the box capability such as pass through fields, codes fields, dare fields, money fields. Also, in the present embodiment, the analytic data interface includes dimension table specific features (e.g. surrogate key generation, slowly changing dimension support, and support for effective dates), fact table specific features (e.g. support for exchange rate lookup and support for dimension key lookup), key resolution, key generation, "bad" data flagging. In the present embodiment, the analytic data interface allows a user to insert, update, or reject a determination.

[**0060**] In the present embodiment, analytic data interface **5** includes slowly changing dimension logic for tracking historically important data. A historically significant attribute is one that you want to retain for your records, even if subsequent records show that a change has been made. In the present embodiment, records within analytic data interface **5** can be configured using two different types of slowly changing dimension categories: historically insignificant attributes (Type 1 slowly changing dimensions) and historically significant attributes (Type 2 slowly changing dimensions). For type 1 slowly changing dimensions, the data field is simply overwritten. Although type 1 slowly changing dimensions does not maintain history, it is the simplest and fastest slowly changing dimension. Type 1 slowly changing dimension is used when the old value of the changed dimension is not deemed important or of interest to track, or is a historically insignificant attribute. For example, a user may want to use type 1 when changing incorrect values in a field. This way, there is no information for that record based on incorrect values. For example, when state name in a supplier table is a type 1 slowly changing dimension, upon changes to the state in which a supplier is located in, the previous value is overwritten (the previous state name) and the previous value is not saved.

[**0061**] Type 2 slowly changing dimensions create a new record. This is the most common slowly changing dimension because it allows the user to track history. The old record allows for pointing to all history prior to the change, and the new record points to all history after the change. Because each change generates a new record, old and new records allow for partition history exactly. In the previous example, when state name in a supplier table is a type 2 slowly changing dimension, upon changes to the state in which a

supplier is located in, a new, current record is generated. The previous value remains a record, and the new current record is a separate record.

[0062] The slowly changing dimension logic of the present invention gives four types of records that are stored in the staging area, new records, changed records with data that is not historically tracked, changed records having historical significance, and changed records having historical significance, and changed records whose changes have no significance of any kind.

[0063] In one embodiment, a new customer key is used for the old sales record while the old customer key continues to be used for the new record. By assigning a new customer key, there is no need for a new addition to the customer table. A simple overwrite of the record showing the new combination suffices. As changed slowly changing dimension records come into a fact and dimension tables, the dimension table key is resolved only when both of the following facts are true: the key does not already exist in the data mart, and the key resolution attributes of the fact change.

[0064] In the present embodiment, a predetermined alphanumeric character is used to indicate a need for data cleansing. That is, because most analytic data interface fields are mapped to fields in the transaction system, be it Ariba, ORMS or SAP, some fields may not be populated with values. For instance, a row in the supplier table may have information on a supplier's address, but may have no value in the supplier's region field. If a report is run on supplier prices by region, the suppliers for whom region information is missing would normally be excluded. However, analytic data interface **5** provides a feature to identify all occurrences of missing values. In the present embodiment the identifier for missing values is a question mark ("?"). When the missing value fields are populated with the question mark, and a report is run on supplier prices by region, the suppliers for whom region information was missing are shown under a region identified by the character "?." The question mark is a sign that the organization's data needs to be "cleansed."

[0065] Cleansing the data in this case involves drilling into the category marked as "?" to learn, perhaps the supplier names or numbers within that group. The data warehouse administrator can then correct each of those suppliers by entering in the regional information on the back end.

[0066] In the present embodiment, the logic for populating null fields is in the analytic data interface **5**. More particularly, the analytic data interface looks for columns that are both linked to a character data type and that are null, and populates them with a "?." It is appreciated that the use of a character such as a question mark is simply the default setting to represent missing data. The present invention is well adapted for using a different character or multiple characters.

[0067] Because the data input into analytic data interface **5** has a common structure, there is no need to process data from each data source independently. More particularly, the data received at analytic data interface **5** has already been translated to obtain meaningful business terms (step **402**) and has been processed to obtain data having a common structure (step **404**). Therefore, the data from different sources (e.g. databases **2-3** and web logs **4**) can be treated as a single data source for the purpose of transformation (step **405**).

[0068] In the present embodiment, because the analytic data interface includes a graphical user interface, it is easy to configure and customize how business data is loaded into an analytic applications system such as a data warehouse. In addition, because the analytic data interface includes a simplified abstraction layer for the data warehouse administrator, the warehouse administrator can configure how data is loaded into the analytic applications in a fraction of the time it takes to configure these capabilities programmatically. In addition, because most of the complex technical problems are solved prior to data entering the analytic data interface, without any required configuration or analysis by the warehouse administrator, the task of configuring data is greatly simplified. Thus, the present invention greatly simplifies the process of loading data into a data warehouse, saving significant expense and time.

[0069] The benefits are particularly apparent for companies that use multiple different operational databases. More particularly, there is no need to define and program for data transport for each different operational database. The warehouse administrator needs only define and program for data transport a single time using the graphical user interface of the analytic data interface.

[0070] In the present embodiment, maplets (reusable objects that represent a set of transformations) are used for code lookup, for address lookup, and for extraction. Also, maplets are used to identify all business locations and identify all business hierarchical structures.

[0071] The transformed data is loaded into an analytic applications system such as a data warehousing application. In the embodiment shown in **FIG. 4**, the data is loaded into a data warehouse as shown by step **406**. In the embodiment shown in **FIG. 2**, the data is loaded into target databases **250-260** which can be databases of a data warehouse. In the embodiment shown in **FIG. 3**, the data is loaded into data warehouse **6**. In addition, in the present embodiment, data is also loaded into operational data store **7**. Applications **8** then extract data from data warehouse **6** and operational data store **7** for performing analysis.

[0072] The method and apparatus of the present invention are illustrated in the following example in which database **2** is an Ariba database and database **3** is a SAP database. In this embodiment, analytic business component **12** is an analytic business component for an Ariba database and analytic business component **13** is an analytic business component for a SAP database. Thus, staging area **32** will contain data from an Ariba database that has been translated in order to include meaningful business terms, staging area **33** will contain data from an SAP database that has been translated in order to include meaningful business terms. Similarly, staging area **34** will contain data from web logs that has been translated in order to include meaningful business terms. In this embodiment, source adapter **32** will be an Ariba source adapter and source adapter **33** will be a SAP source adapter while no source adapter is required for data from web logs **4**. Because the data from each of sources **2-4** is provided to analytic data interface **5**, the data can be treated as a common data source for the purpose of transforming the data into a format suitable for loading into a data warehousing application (step **405** of **FIG. 4**). Therefore, the warehouse administrator needs only define and program for data transport a single time using the graphical user interface of the

analytic data interface. Not three separate times as would be required by prior art systems.

[0073] Accordingly, the present invention provides a method and apparatus that allows for transporting data such that the data can be used in data warehousing applications. In addition, the present invention provides a method and apparatus that takes advantage of the standardization of database components. Moreover, the present invention provides a method and apparatus that reduces the time required to define and program data transport for data warehousing applications.

[0074] While the present invention has been described in particular embodiments, it should be appreciated that the present invention should not be construed as limited by such embodiments, but rather construed according to the below claims.

1-23. (canceled)

24. The method of claim 26, further comprising:

in response to a change to a dimension without a historically significant attribute, overwriting the dimension.

25. (canceled)

26. A method for tracking historical data from different sources, comprising:

processing source-specific data originating at sources with disparate formats into source-independent data with a single, common format;

storing the source-independent data;

automatically determining dimensions of the stored data having historically significant attributes;

in response to a change to a dimension having a historically significant attribute, creating a historical record of the change, wherein creating a historical record of the change further comprises:

creating a first record for the stored dimension having a historically significant attribute;

generating a first key;

associating the first key with the first record;

creating a second record to store the change to the dimension having a historically significant attribute;

re-associating the first key with the second record;

generating a second key; and

associating the second key with the first record.

27. The method of claim 26, wherein processing source-specific data originating at sources with disparate formats into source-independent data with a single, common format further comprises:

performing source-related clean up.

28. The method of claim 26, wherein processing source-specific data originating at sources with disparate formats into source-independent data with a single, common format further comprises:

configuring unique key identification information.

29. A computer program product for tracking historical data from different sources, comprising:

a computer-readable medium; and

computer program code, encoded on the medium, for controlling a computer system to perform the operations of:

processing source-specific data originating at sources with disparate formats into source-independent data with a single, common format;

storing the source-independent data;

automatically determining dimensions of the stored data having historically significant attributes; and

in response to a change to a dimension having a historically significant attribute, creating a historical record of the change by:

creating a first record for the dimension having a historically significant attribute;

generating a first key;

associating the first key with the first record;

creating a second record to store the change to the dimension having a historically significant attribute;

re-associating the first key with the second record;

generating a second key; and

associating the second key with the first record.

30. (canceled)

31. A system for tracking historical data from different sources, comprising:

a source adapter for processing source-specific data originating at sources with disparate formats into source-independent data with a single, common format; and

an analytic data interface for storing the source-independent data, automatically determining dimensions of the stored data having historically significant attributes, and creating a historical record of a change to a dimension having a historically significant attribute, wherein the analytic data interface further comprises:

a record generation module for creating a first record for the dimension having a historically significant attribute and creating a second record to store the change to the dimension having a historically significant attribute;

a key generation module for generating a first key and a second key to uniquely identify records; and

an association module for associating the first key with the first record, re-associating the first key with the second record, and associating the second key with the first record.

32. The computer program product of claim 29, further comprising computer program code, encoded on the medium, for controlling a computer system to perform the operation of, in response to a change to a dimension without a historically significant attribute, overwriting the dimension.

33. The system of claim 31, wherein the analytic data interface is further configured for, in response to a change to a dimension without a historically significant attribute, overwriting the dimension.

* * * * *