US006195662B1

(12) **United States Patent**
Ellis et al.

(10) **Patent No.: US 6,195,662 B1**
(45) **Date of Patent: Feb. 27, 2001**

(54) **SYSTEM FOR TRANSFORMING AND EXCHANGING DATA BETWEEN DISTRIBUTED HETEROGENEOUS COMPUTER SYSTEMS**

(75) Inventors: **Darin Ellis**, Orleans; **Michael Doyle**, Gloucester, both of (CA)

(73) Assignee: **Juxtacomm Technologies Inc.**, Gloucester (CA)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/105,299**

(22) Filed: **Jun. 26, 1998**

Related U.S. Application Data

(60) Provisional application No. 60/051,052, filed on Jun. 27, 1997.

(51) **Int. Cl.**[7] **G06F 17/00**

(52) **U.S. Cl.** **707/103**; 707/1; 707/2; 707/8; 707/10; 707/102

(58) **Field of Search** 707/1–2, 8, 10, 707/102–103

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,497,491	*	3/1996	Mitchell et al.	707/1
5,729,733	*	3/1998	Sharif-Askary	707/8
5,799,310	*	8/1998	Anderson et al.	707/100
5,970,490	*	10/1999	Morgenstern	707/10
5,988,862	*	11/1999	Kacyra et al.	705/5
6,047,291	*	4/2000	Anderson et al.	707/103
6,078,925	*	6/2000	Anderson et al.	707/2

OTHER PUBLICATIONS

IEEE publication, "The HERMES Language for Work Session Specification" by Manolis Marazakis et al., Institute of Computer Science, FORTH, Greece, pp. 542–547, Aug. 1998.*

* cited by examiner

Primary Examiner—Thomas G. Black
Assistant Examiner—Diane D. Mizrahi
(74) *Attorney, Agent, or Firm*—Hardaway/Mann IP Group Nexsen Pruet Jacobs & Pollard, LLP

(57) **ABSTRACT**

A system and method is described for importing data from a source computer system, manipulating and transforming of that data, and exporting the data to a target computer system under control of a script processor using stored metadata definitions. Metadata is used to describe the properties of the data being manipulated. The system includes a means for manipulating the metadata definitions. The metadata definitions are created to import data into the system, export data from the system, create views of the external data, store generic format data within the system, manipulate generic format data within the system and to control data flow through the system. Data is imported into the system using an import data definition to map the external data into an import data bag. Data imported into an import data bag becomes independent of the original data source. Data is manipulated within the system using script control commands and data is transformed within the system using rule sets that act upon data bags. Data is exported from the system using an export data definition to map the import data bag into the required export data bag format and then to write data in the export data bag to the external data target.

19 Claims, 12 Drawing Sheets

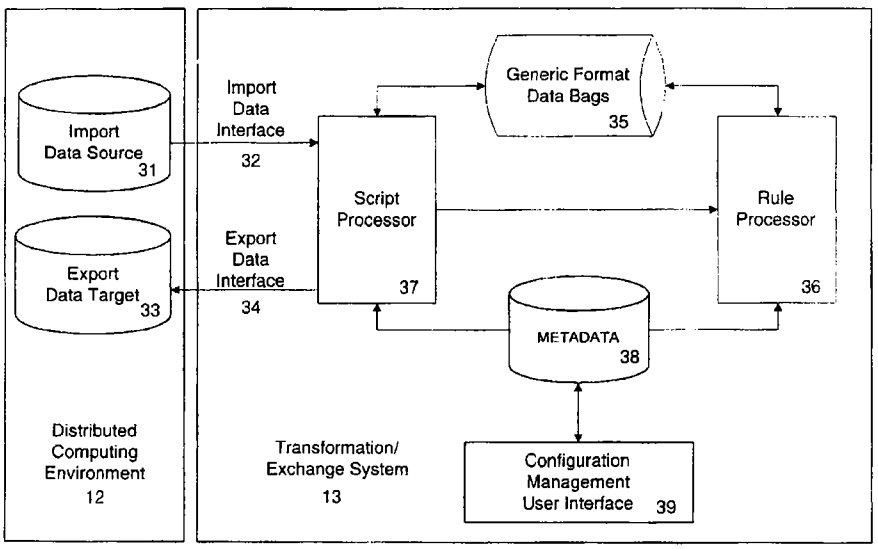

EXHIBIT
A

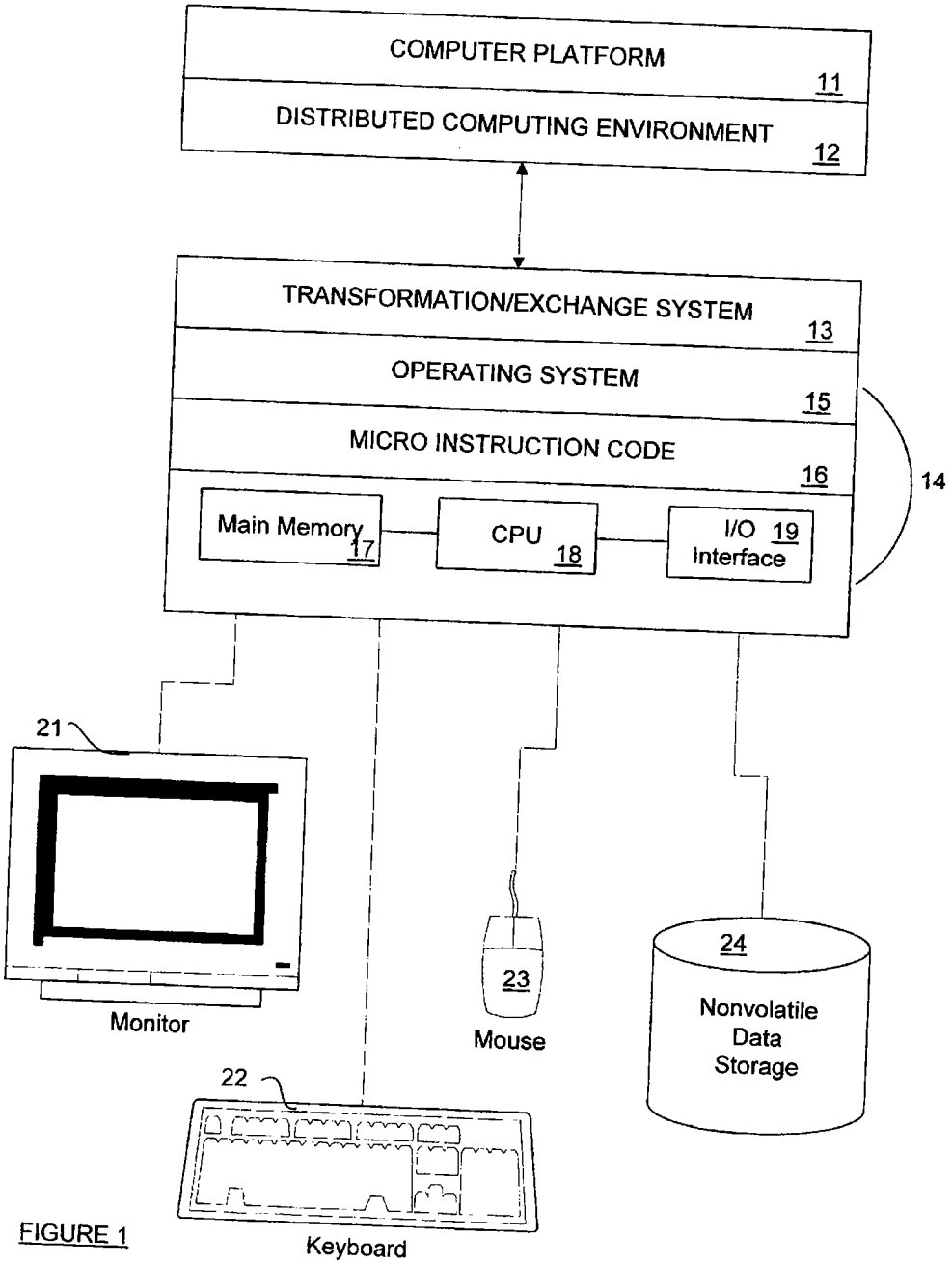
COMPUTER PLATFORM
11
DISTRIBUTED COMPUTING ENVIRONMENT
12
TRANSFORMATION/EXCHANGE SYSTEM
13
OPERATING SYSTEM
15
MICRO INSTRUCTION CODE
16
Main Memory
17
CPU
18
I/O
19
Interface
14
21
Monitor
23
Mouse
24
Nonvolatile
Data
Storage
22
Keyboard

FIGURE 1

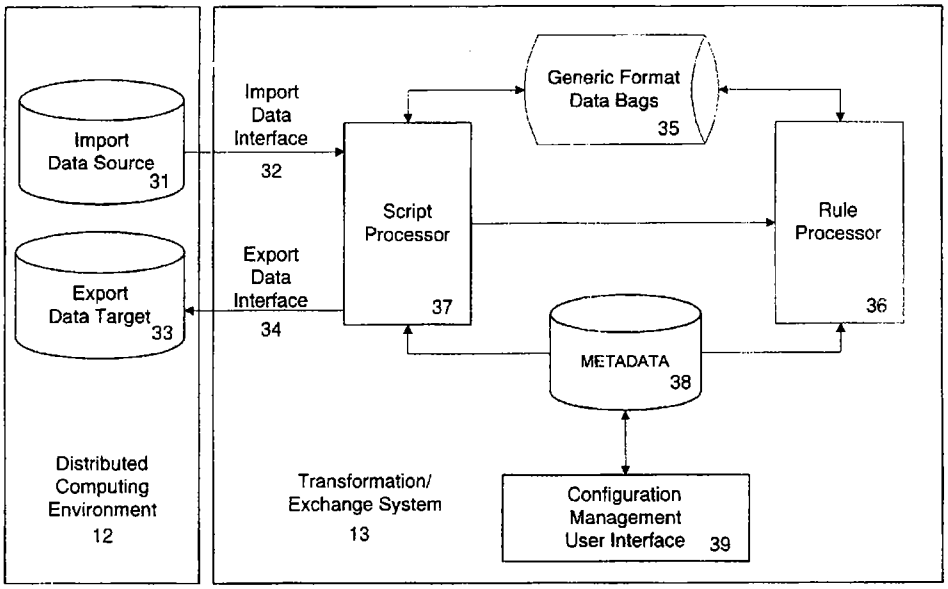

FIGURE 2

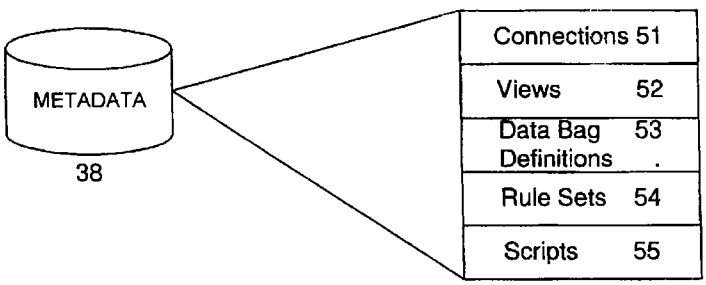

FIGURE 3

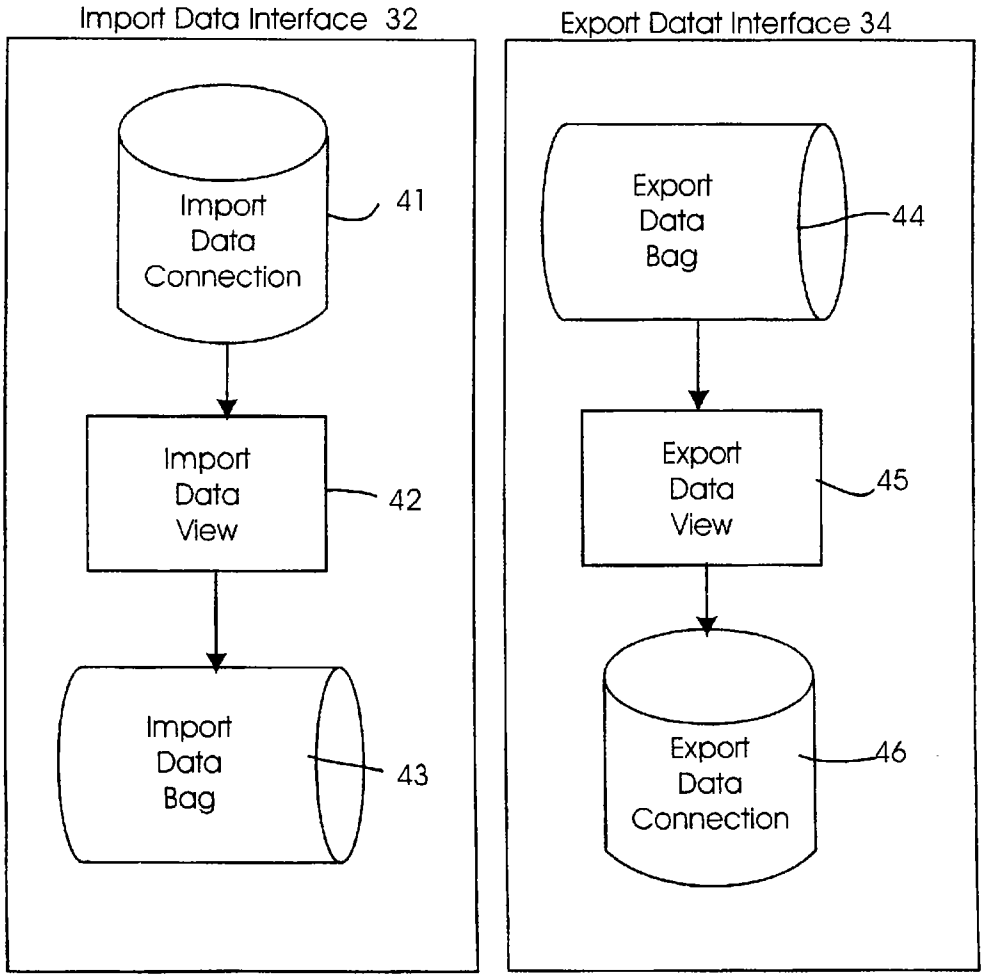
Import Data Interface 32
Import Data Connection
41
Import Data View
42
Import Data Bag
43
Export Datat Interface 34
Export Data Bag
44
Export Data View
45
Export Data Connection
46

FIGURE 4

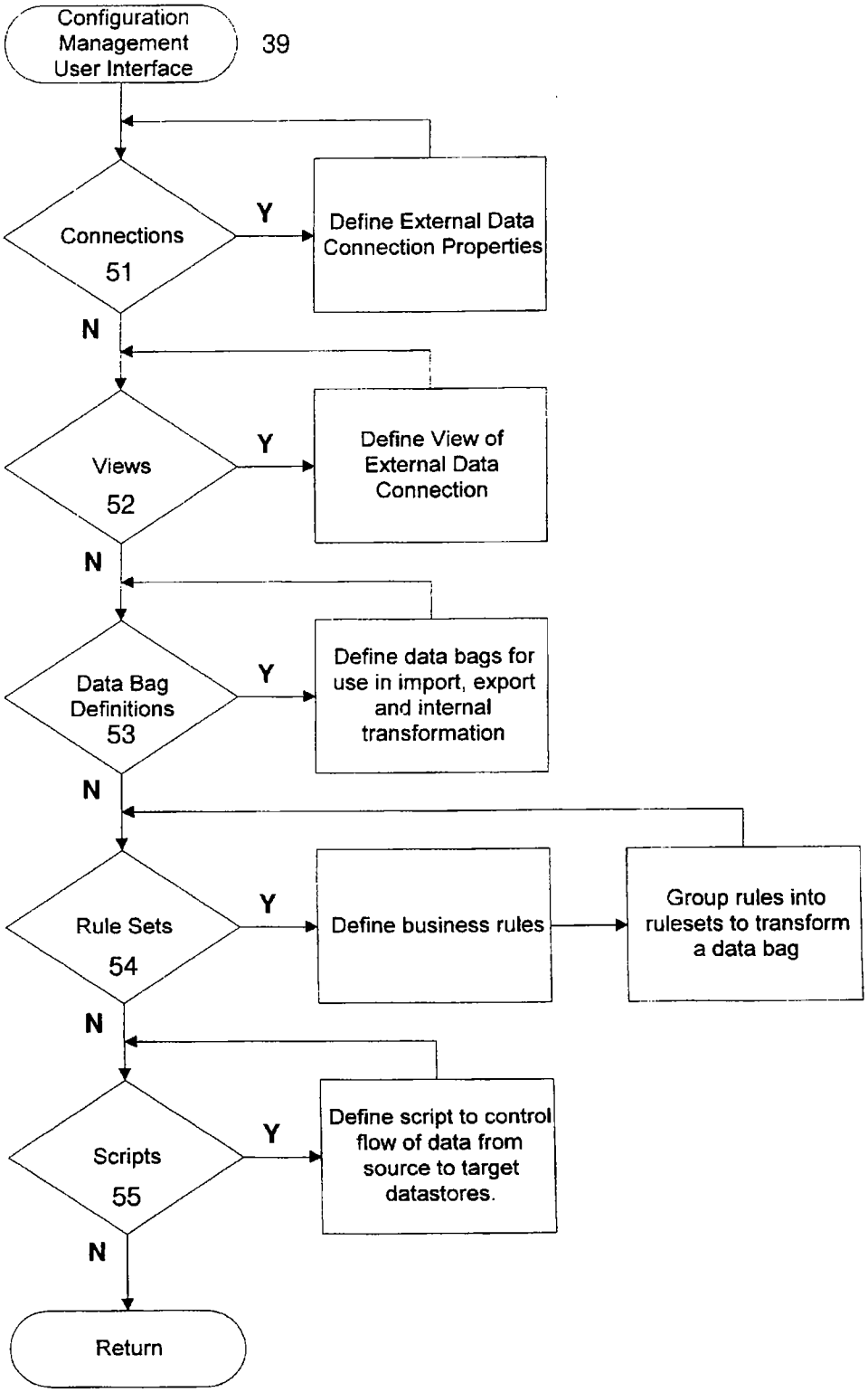
Configuration Management User Interface
39
Connections
51
Y
Define External Data Connection Properties
N
Views
52
Y
Define View of External Data Connection
N
Data Bag Definitions
53
Y
Define data bags for use in import, export and internal transformation
N
Rule Sets
54
Y
Define business rules
Group rules into rulesets to transform a data bag
N
Scripts
55
Y
Define script to control flow of data from source to target datastores.
N
Return

FIGURE 5

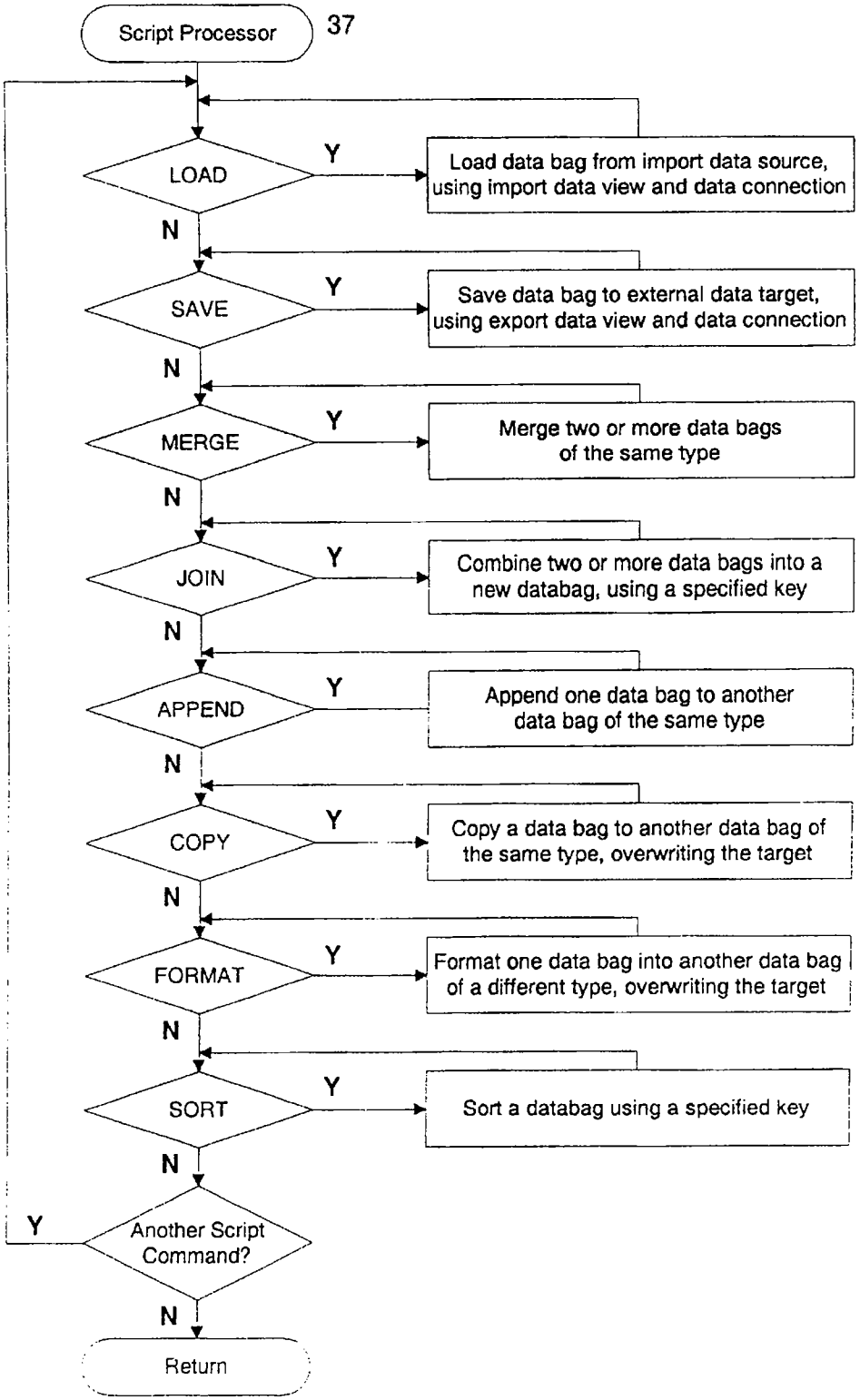
Script Processor
37
LOAD
Y
Load data bag from import data source, using import data view and data connection
N
SAVE
Y
Save data bag to external data target, using export data view and data connection
N
MERGE
Y
Merge two or more data bags of the same type
N
JOIN
Y
Combine two or more data bags into a new databag, using a specified key
N
APPEND
Y
Append one data bag to another data bag of the same type
N
COPY
Y
Copy a data bag to another data bag of the same type, overwriting the target
N
FORMAT
Y
Format one data bag into another data bag of a different type, overwriting the target
N
SORT
Y
Sort a databag using a specified key
N
Y
Another Script Command?
N
Return

FIGURE 6

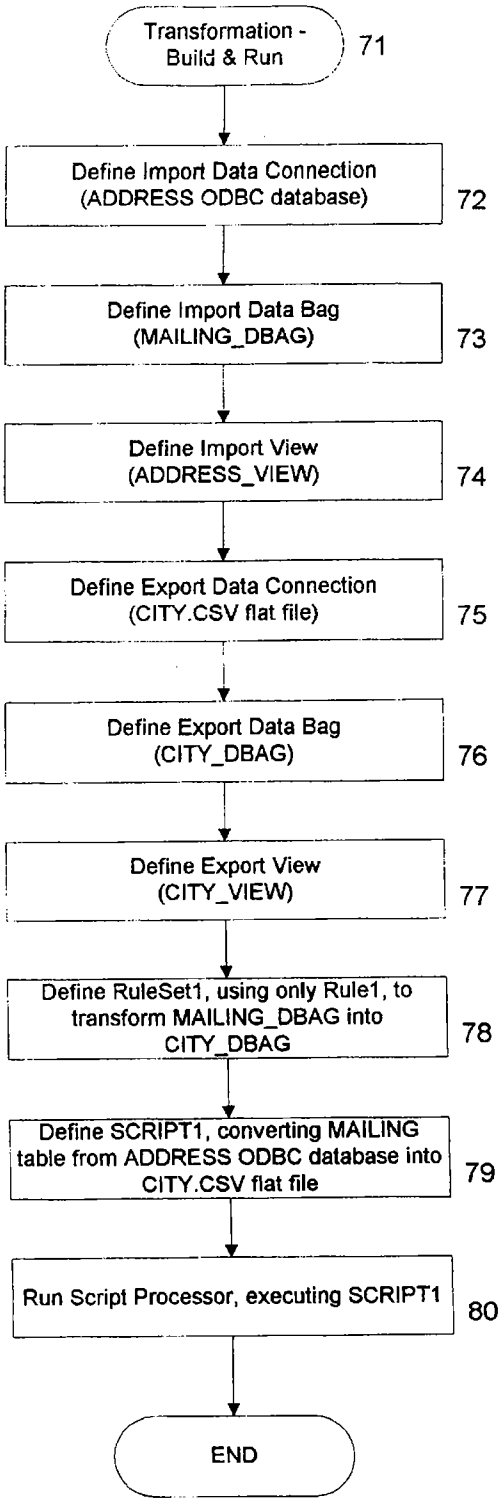
Transformation - Build & Run
71
Define Import Data Connection (ADDRESS ODBC database)
72
Define Import Data Bag (MAILING_DBAG)
73
Define Import View (ADDRESS_VIEW)
74
Define Export Data Connection (CITY.CSV flat file)
75
Define Export Data Bag (CITY_DBAG)
76
Define Export View (CITY_VIEW)
77
Define RuleSet1, using only Rule1, to transform MAILING_DBAG into CITY_DBAG
78
Define SCRIPT1, converting MAILING table from ADDRESS ODBC database into CITY.CSV flat file
79
Run Script Processor, executing SCRIPT1
80
END

FIGURE 7

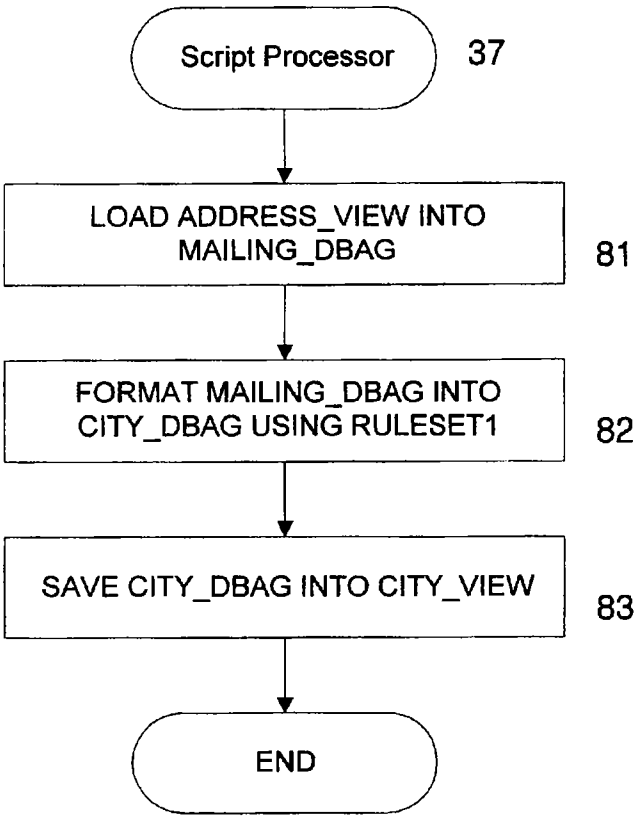

FIGURE 8

```
IF IN.CITY = "OTTAWA"
        OUT.NAME = APPEND( IN.FIRST_NAME, " ", IN.LAST_NAME )
        OUT.ADDR = IN.ADDR
        OUT.CITY = IN.CITY
        OUT.AGE = IN.AGE
END IF
```

FIGURE 9

ODBC database : ADDRESS.MDB 101

FIRST_NAME	LAST_NAME	ADDRESS	CITY	COUNTRY	AGE
Bob	Drake	10 Penny St.	Toronto	CA	29
Susan	Hellman	55 Birch Rd.	Ottawa	CA	35
John	Bull	101 Pullman Ave.	New York	US	23
Josef	Karsch	31 Rideau St.	Ottawa	CA	85

FIGURE 10

Data Bag : MAILING_DBAG 111

DATA DEFINITION COLLECTION 112

KEY	ITEM NAME	ITEM TYPE	PARENT	MAX. OCCURS	ITEM COUNTER	LEVEL
FIRST_NAME	FIRST_NAME			1		1
LAST_NAME	LAST_NAME			1		1
ADDRESS	ADDRESS			1		1
CITY	CITY			1		1
COUNTRY	PROV			1		1
AGE	AGE			1		1

DATA GROUP COLLECTION 113

KEY	VALUE	KEY	VALUE	KEY	VALUE	KEY	VALUE
FIRST_NAME	Bob	FIRST_NAME	Susan	FIRST_NAME	John	FIRST_NAME	Josef
LAST_NAME	Drake	LAST_NAME	Hellman	LAST_NAME	Bull	LAST_NAME	Karsch
ADDRESS	10 Penny St.	ADDRESS	55 Birch Rd.	ADDRESS	101 Pullman Ave	ADDRESS	31 Rideau St.
CITY	Toronto	CITY	Ottawa	CITY	New York	CITY	Ottawa
COUNTRY	CA	COUNTRY	CA	COUNTRY	US	COUNTRY	CA
AGE	29	AGE	35	AGE	23	AGE	85

FIGURE 11

Data Bag : CITY_DBAG 121

DATA DEFINITION COLLECTION 122

KEY	ITEM NAME	ITEM TYPE	PARENT	MAX. OCCURS	ITEM COUNTER	LEVEL
NAME	NAME	TEXT		1		1
ADDRESS	ADDRESS	TEXT		1		1
CITY	CITY	TEXT		1		1
AGE	AGE	INTEGER		1		1

DATA GROUP COLLECTION 113

KEY	VALUE	KEY	VALUE
FIRST_NAME	Susan	FIRST_NAME	Josef
LAST_NAME	Hellman	LAST_NAME	Karsch
ADDRESS	55 Birch Rd.	ADDRESS	31 Rideau St.
CITY	Ottawa	CITY	Ottawa
COUNTRY	CA	COUNTRY	CA
AGE	35	AGE	85

FIGURE 12

Delimited Flat File : OTTAWA.CSV 131

Susan Hellman,55 Birch Rd,Ottawa,35
Josef Karsch,31 Rideau St,Ottawa,85

FIGURE 13

```
FIRST_NAME        CHAR(30)
LAST_NAME         CHAR(30)
ADDR_LINE1        CHAR(30)
ADDR_LINE2        CHAR(30)
CITY              CHAR(30)
PROV              CHAR(30)
NBR_CHILDREN      INTEGER(3)
CHILDREN          OCCURS 1 TO 10 TIMES
    CHILD_NAME        CHAR(30)
    CHILD_AGE         INTEGER(3)
```

FIGURE 14

Repeating Data File : REPEAT.TXT 151

Henry	Smith	61 Falldown Rd.	RR1	Greely	Ont	3	Amy	16	John	13	Bob	23
John	Kierstead	1034 Ayr Place	Apt. 132	Ottawa	Ont	1	Susan	10				

FIGURE 15

Data Bag : REPEATING_DBAG 161

DATA DEFINITION COLLECTION 162

KEY	ITEM NAME	ITEM TYPE	PARENT	MAX. OCCURS	ITEM COUNTER	LEVEL
FIRST_NAME	FIRST_NAME	TEXT		1		1
LAST_NAME	LAST_NAME	TEXT		1		1
ADDR_LINE1	ADDR_LINE1	TEXT		1		1
ADDR_LINE2	ADDR_LINE2	TEXT		1		1
CITY	CITY	TEXT		1		1
PROV	PROV	TEXT		1		1
NBR_CHILDREN	NBR_CHILDREN	INTEGER		1		1
CHILDREN	CHILDREN	GROUP		1		1
CHILDREN[*].CHILD_NAME	CHILD_NAME	TEXT	CHILDREN	10	NBR_CHILDREN	2
CHILDREN[*].CHILD_AGE	CHILD_AGE	INTEGER	CHILDREN	10	NBR_CHILDREN	2

DATA GROUP COLLECTION 163

FIRST_NAME	Henry	FIRST_NAME	John
LAST_NAME	Smith	LAST_NAME	Keirstead
ADDR_LINE1	61 Falldown Rd.	ADDR_LINE1	1034 Ayr Place
ADDR_LINE2	RR1	ADDR_LINE2	Apt. 132
CITY	Greely	CITY	Ottawa
PROV	Ont	PROV	Ont
NBR_CHILDREN	3	NBR_CHILDREN	1
CHILDREN[1].CHILD_NAME	Amy	CHILDREN[1].CHILD_NAME	Susan
CHILDREN[1].CHILD_AGE	16	CHILDRE[1].CHILD_AGE	10
CHILDREN[2].CHILD_NAME	John		
CHILDREN[2].CHILD_AGE	13		
CHILDREN[3].CHILD_NAME	Bob		
CHILDREN[3].CHILD_AGE	23		

FIGURE 16

```
FOR CHILD_NDX = 1 TO NBR_CHILDREN
        IF CHILDREN[ CHILD_NDX ].CHILD_AGE < 20
                OUT.PARENT = APPEND( IN.FIRST_NAME, " ", IN.FIRST_NAME )
                OUT.CHILD_NAME = IN.CHILDREN[ CHILD_NDX ].CHILD_NAME
                OUT.CHILD_AGE = IN.CHILDREN[ CHILD_NDX ].CHILD_AGE
        END IF
END FOR
```

FIGURE 17

SYSTEM FOR TRANSFORMING AND EXCHANGING DATA BETWEEN DISTRIBUTED HETEROGENEOUS COMPUTER SYSTEMS

RELATED APPLICATIONS

This invention claims the benefit of priority under Title 35 USC §119(e) of provisional application for patent Ser. No. 60/051,052, filed Jun. 27, 1997.

FIELD OF THE INVENTION

This invention relates to a system and method for importing, transforming and exporting data between distributed heterogeneous computer systems and in particular to a system of script processing utilizing metadata to control data transformation within the system and data movement into and out of the system.

BACKGROUND OF THE INVENTION

Data exchange between distributed heterogeneous computer systems has been problematic in the industry. Businesses frequently use disparate data formats and data storage types within a corporate structure. As well, business partners almost invariably use different data formats. To permit data exchange when different formats are used, a static intercommunication facility must be maintained for each pair of disparate data formats and/or data storage types. Changes to data formats or data storage types force the re-engineering of the corresponding facility.

A data import/export system is taught in U.S. Pat. No. 5,497,491 which issued on Mar. 5, 1996 to Mitchell et al. That patent describes a system and method for importing and exporting data between an external object oriented computing environment. The system and method requires a datalist object for each field to be moved from the external object oriented computing environment to the external computing environment. A metadata object is required for each datalist object. The system is therefore complex and resource-use intensive. Furthermore, it is only capable of moving data from an object-oriented to some other computer environment. The system is therefore inflexible and unsuitable for use in many applications where import/export must be performed between two computer systems that do not use object oriented data formats.

Therefore, what is needed is a distributed system and method that is capable of transforming data from a source computer system into data usable by a computer system which stores data in a different format. This system must provide a simple means for specifying the transformation definitions and for controlling the flow of data from an input data source to an output data target. Configuration management of the system must be dynamic to respond to the changing business environment and non-intrusive to minimize the effects of changing data formats or data storage types.

SUMMARY OF THE INVENTION

It is therefore an object of this invention to provide a system and method for data transformation and data exchange between distributed heterogeneous computer systems.

It is another object of the present invention to provide a script processing language that defines operations to control data transformation within the system and data movement into and out of the distributed system, utilizing metadata definitions.

It is another object of the present invention to provide a format control language that defines the transformation of an external data source into data bags and of the internal data bags to an external data target.

It is another object of the present invention to provide a means of configuration management that allows a user of the system to define scripts, import data connections, export data connections, data bags, and rule set definitions and to store them in a metadata database.

It is another object of the present invention to provide a means of executing scripts in order to control the distributed transformation system.

According to the invention, there is provided a system for transforming and exchanging datastore data between heterogeneous computer systems using different datastore formats for storing similar information, the system comprising: means for transforming and processing import datastore data into generic format data according to predetermined import transformation rules and functions; means for converting the generic format data into export datastore data according to predetermined export transformation rules and functions; and interface to communications means for receiving the import datastore data and for transmitting the export datastore data.

A datastore refers to the storing of any type of data in a persistent storage system, such as on magnetic media like a disk drive. The types of data stored could include text or binary.

As will be shown below, the present invention can be used to create import data definitions, data bag storage, data bag transformation definitions or rule sets, export data definitions and scripts to control the usage of all those definitions in the process of transforming and exchanging data between dissimilar computer systems.

A generic format data bag contains both the data to be manipulated and the data structure definitions, in a generic format. The present invention will use the title 'data bag' to indicate a generic format data bag.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** shows a block diagram of a data transformation and exchange environment, an external distributed computing environment and the associated hardware platforms.

FIG. **2** shows a block diagram of a system for transforming and exchanging data between heterogeneous distributed computing environments according to the present invention.

FIG. **3** shows the components of the present invention that are defined within the metadata database.

FIG. **4** shows the operations performed by the import data interface **32** and the export data interface **34** when the script processor **37** of FIG. **2** is invoked.

FIG. **5** is a flow diagram showing operations performed by the configuration management user interface **39** of FIG. **2** at program execution time.

FIG. **6** is a flow diagram showing operations performed by the script processor **37** of FIG. **2**.

FIG. **7** shows an example of the operations to define the components for a data transformation.

FIG. **8** shows an example script to control the data transformation defined in FIG. **7**.

FIG. **9** shows an example of part of a rule that could be used in the data transformation defined in FIG. **7**.

FIG. **10** shows the internal storage of an example ODBC-enabled database table used in the data transformation defined in FIG. **7**.

FIG. **11** shows the internal storage of the data bag used to store the imported data defined and used in the data transformation defined in FIG. **7**.

FIG. **12** shows the internal storage of the data bag used to store the data for export that is defined and used in the data transformation defined in FIG. **7**.

FIG. **13** shows the internal storage of the export data target used in the data transformation defined in FIG. **7**.

FIG. **14** shows the data layout, such as might appear in a computer program, of a text file containing personal information records. There is a repeating group of information at the end of each record. This data layout example will be used to show how data bags can handle repeating groups of data.

FIG. **15** shows the text file, defined in FIG. **14**, with some example data.

FIG. **16** shows a data bag containing the data from the text file defined in FIG. **15**. The data group definition **162** shows how the 'CHILDREN' group is defined and the data group collection **163** shows how the 'CHILDREN' group is stored.

FIG. **17** shows an example rule that would act on the data bag defined in FIG. **16** and output only the personal records that contained children whose age is less than **20**.

DETAILED DESCRIPTION OF THE INVENTION

Prior to describing a system and method for data transformation and data exchange between distributed heterogeneous computer systems according to the present invention, a general overview of the computing environment will be provided. A general description of the system and method of the present invention will then be provided, followed by a detailed design description for the system and method for data transformation and data exchange according to the present invention.

Referring to FIG. **1** and FIG. **2**, the hardware and software environment in which the present invention operates will now be described. The present invention is a method and system for data transformation and data exchange between an external distributed computing environment **12** operating on one or more computer platforms **11** and a transformation/exchange system **13** operating on one or more computer platforms **14**. It will be understood by those having skill in the art that each of computer platforms **11** and **14** typically include computer hardware units such as main memory **17**, a central processing unit (CPU) **18** and an input/output (I/O) interface **19**, and may include peripheral components such as a display terminal **21**, an input device such as a keyboard **22** or a mouse **23**, nonvolatile data storage devices **24** such as magnetic or optical disks and other peripheral devices. Computer platform **11** or **14** also typically includes microinstruction code **16**, and an operating system **15**. As one example, each computer platform **11** and **14** may be a desktop computer having an IBM PC architecture. Operating system **15** may be a Microsoft Windows NT operating system. FIG. **2** is a functional block diagram of the current invention. It will be understood by those having skill in the art that this architecture might be implemented on multiple machines and will vary according to the application.

Referring to FIG. **1**, a system **13** for transformation and exchange between distributed heterogeneous computer systems **12**, according to the present invention, is shown. As shown in FIG. **2**, the transformation and exchange system **13** includes an import data interface **32** to import data from an import data source **31** into the transformation and exchange system **13**. As shown in FIG. **4**, the import data interface **32** includes an import data connection **41**, an import data view **42** of the import data source **31** and a generic format data bag **43** where the imported data is to be stored. Those skilled in the art will understand that a view is a logical subset of the content of an actual external data source. The import data view **42** is a logical subset of the content of the import data source **31**. As will be shown below, the import data view **42** will be used during the execution of the script processor **37** (FIG. **2**) to load data from the import data source **31** into the data bag **43**.

Data bags **43** are used in the present invention for the storage and transformation of external data. A data bag contains both the definition of the data contained within the data bag and the actual generic format data. Generic format data refers to data that has been stored within the present invention and is now independent of the original data source. Data stored in this generic format can be transformed into any required format for exporting to an export data target **33** (FIG. **2**). Data bags are stored in non-persistent storage, like main memory **17**, are created by the script processor and exist while the script is running. Data bags can contain fixed format data, data grouping and repeating data groups.

Referring again to FIG. **2**, the system includes an export data interface **34** to export a data bag **44** out to an export data target **33**. As shown in FIG. **4**, the export data interface **34** includes generic format data bag **44** where data for exporting is stored, the export data view **45** of the data bag **44** and the export data connection **46**. As will be shown below, the export data view **45** of the data bag **44** will be used during the execution of the script processor **37** to save data from the data bag **44** out to the export data target **33**.

Also shown in FIG. **2** and FIG. **3**, the system includes a configuration management user interface **39** to define the components of the present invention, which include external data connections **51**, views **52**, data bags **53**, rule sets **54** and scripts **55**. These component definitions are stored in the metadata database **38**. The data bags are stored in the internal datastore **35**. The component definitions will be described in detail below.

The transformation and exchange system **13** includes a script processor **37**, in order to run scripts **55** defined in the metadata database **38**. The script processor **37** identifies the script command and invokes the correct method for that script command. The transformation/exchange system **13** also contains a rule processor **36** that is invoked by the script processor **37** to transform one data bag into another data bag based on a rule. Rules will be described below.

FIG. **5** is a flow diagram showing the components that can be defined using the configuration management user interface **39** and the actions taken when defining each component.

Connections **51** must have their connection type and properties defined. The connection type will be any of the industry standard data storage types, such as ODBC-enabled databases, spreadsheets, message-oriented middleware and text files. The properties will include the name and location of the external data storage.

Views **52** must be associated with either an import data connection **41** (FIG. **4**) or an export data connection **46**. Each data connection has one or more views of the external data. These views are used to import different collections of data from an import data source **31** (FIG. **2**), or to export different collections of data from an export data bag **44** out to an export data target **33**.

Data bag definition **53** contain two types of data collection: a data definition collection and a data group collection.

A collection is a logical grouping of records that use the same format method. All the element definitions for a data bag are stored within the data definition collection. Each row of data in a data bag is stored as one data group in the data group collection. The data group collection contains all the data groups in the data bag. An import data connection **41** must have one or more import data views **42** and each data view must have an associated import data bag **43**. Using the data definitions in the data definition collection **112** (FIG. **11**) of a data bag, the import data view **42** of the import data connection **41** is loaded in the import data bag **43**. An export data connection **46** must have one or more export data views **45** and each data view must have an associated export data bag **44**. Using the data definitions in the data definition **112** of a data bag, the export data view **45** of the export data connection **46** is written using the data contained in the data bag. Data bags are also defined for use by script commands that require import and export data bag(s), where these commands transform the data from the import data bag **43** and place the results in the export data bag **44**.

Rule sets **54** (FIG. **5**) are collections of rules within the present invention. Rule sets are used to transform a data bag in one format into another data bag of a different format. The purpose of a rule is to perform a specific operation to achieve a desired result. A rule is one or more statements. These statements are executed from top to bottom and when the last statement within the rule has been executed, or an Exit statement is encountered, the rule ends.

A statement is a single line in a rule. The types of statements implemented by the present invention, within the rule set processor, includes comments, conditional processing, exiting a rule, looping, variable declaration and variable assignment.

Conditional processing, looping and assignment statements contain expressions. Elementary expressions include strings, numbers, content of a variable and return value of a function. Functions are categorized into character manipulation, string manipulation, including other rules, initialization information, external file manipulation, variable content reporting and user interface.

Complex expressions combine many elementary expressions in some manner, for the purpose of producing a single result. Complex expressions can be either arithmetic or conditional.

Complex arithmetic expressions are numeric elementary expressions that are combined to produce a single arithmetic result. Such expressions follow the standard format of all numeric expressions. Numbers are acted upon by numeric operators such as addition, subtraction, multiplication, division, modulo and exponential. Brackets are used to group numbers and operators which need to be evaluated together.

Conditional expressions return the value True or False. These types of expressions are used to control conditional processing within the rules. Brackets are used to group conditions which need to be evaluated together. Complex conditional expressions are formed by combining simple conditions with 'And' or 'Or' operators.

Simple conditions have a 'left side' 'operator' 'right side' format. The left and right sides are elementary expressions. The logical operators that can be used for these conditions are equals, greater, less, not equal, greater or equal, less or equal, 'like' and 'in'. A simple condition can be negated by using the word 'not' in front of the condition.

Scripts **55** must be defined to control data movement into and out of the system, and to control data transformation within the system.

FIG. **6** is a flow diagram showing the actions taken by the script processor **37** of the present invention.

The LOAD command permits an import data view **42** to be used to load data from an import data source **31** into an import data bag **43**. The import data view **42** is associated with an import data connection **41**, which specifies the import data source **31**.

The SAVE command permits an export data view **45** to be used to save data from an export data bag **44** out to an external data target **33**. The export data view **45** is associated with an export data connection **46**, which specifies the external data target **33**.

The MERGE command permits two or more specified data bags, of the same data bag type, to be merged into another data bag. Only non-duplicate data groups from the input data bags are merged into the output data bag.

The JOIN command permits two or more specified data bags to be joined together into another data bag, dependent on the matching of a specific key value.

The APPEND command permits one data bag to be appended to the end of another data bag of the same data bag type.

The COPY command permits one data bag to be copied to another specified data bag of the same data bag type. If the target data bag exists, it will be overwritten.

The FORMAT command permits the transformation of a specified data bag into another data bag, possibly of a different data bag type, using a specified rule. This command will invoke the rule processor **36** to take the input data bag, transform the data according to the rule statements and populate the output data bag with the transformed data.

The SORT command permits a data bag to be sorted by one or more data elements within the data bag. Each element can have an ascending or descending sort applied to it. The result can be placed back into the original data bag or the result can be written to another data bag of the same type.

An example of a transformation according to a preferred embodiment of the invention will be shown in the following description. The embodiment will describe the definition and usage of an import data connection, export data connection and transformation requirements to convert an Open Database Connectivity (ODBC) enabled database table into a delimited flat file. This example was chosen because of the widespread usage of both ODBC and delimited files in the business community and of the direct application of the invention to the problem of transforming data between these two standards used by heterogeneous computer systems.

FIG. **7** shows the steps taken by the user to set up the definitions required for the preferred embodiment and to initiate the script to carry out the transformation from ODBC database data to delimited data. The user will invoke the configuration management user interface **39** (FIG. **2**) to create the required definitions.

As shown in step **72**, the user defines the import data connection **41** with a connection type of ODBC-enabled database, defines the location of the import data source **31** and defines the table within the database to be used as the data source. In step **73** the user defines the import data bag to hold the data imported from the import data connection **41**.In step **74** the import data view **42** is created to define the fields within the import database table that are to be used when processing the data source. The import data view **42** is associated with the import data bag **43** that will receive the incoming data.

In step **75**, the export data connection **46** is defined with a connection type as file and the location of the target data

file is defined. In step **76**, the user defines the export data bag that will hold the data to be used by the export data connection **46**. In step **77** the export data view **45** is created to define the layout of the target data file. The export data view **45** is associated with the export data bag **44** that will be used to send data to the external data target **33**.

In step **78**, the rule definition allows the user to specify a complex set of statements to control the transformation of one data bag to another data bag. The statements in the rule come from the format control language which includes conditional logic flow control, looping and the ability to define and call functions not defined within the language.

In step **79**, the user then defines the script that will load the import data source **31** into an import data bag **43**, transform the loaded import data bag **43** into an export data bag **44** by executing the rule processor **36** using the specified rule (Rule1), and then exports the export data bag **44** to the external data target **33**.

Finally, in step **80**, the user initiates the script processor **37** to execute the script. The script processor **37** can be initiated from the graphical interface or from an interface external to the system.

FIG. **8** shows the script defined for this example. The first script command **81** uses the import data connection **41** and import data view **42** to load the data from the import data source **31** into the import data bag **43**. The second command **82** transforms the data bag **43** into an export data bag **44** using the specified rule set (RuleSet1). Once the export data bag **44** has been populated with the transformed data it can be saved **83** directly out to the export data target **33**, using the export data view **45** and the export data connection **46**.

FIG. **9** shows an example rule for this example. The example rule demonstrates the use of conditional flow control (IF statement), record selection based on incoming data content (IN.CITY="OTTAWA") and data transformation using assignment statements (for example, OUT.NAME=APPEND(IN.FIRST_NAME, "", IN.LAST_NAME)). In step **78** of FIG. **7**, RuleSet1 is defined to contain one rule (Rule1) which transforms data bag MAILING_DBAG into data bag CITY_DBAG. When Rule1 is executed in the example shown in FIG. **9**, the import data bag refers to MAILING_DBAG and the export data bag refers to CITY_DBAG.

FIG. **10** shows an example import data source **31** for this example. The internal storage of an ODBC-enabled database table is shown. The data in this table will be used to illustrate the data transformation defined in FIG. **7**. The import data connection **41**, defined in step **72**, refers to the exact location of the database file ADDRESS.MDB **101** and indicates that the database is ODBC-enabled. The import data view **42**, defined in step **74**, specifies that all the fields in the data source table will be imported into the import data bag **43**, defined in step **73**.

FIG. **11** shows the internal storage of the import data bag **43**, defined in step **73**, which is used in the data transformation in FIG. **7**. The data definition collection **112** specifies the key name used for locating fields in the data group collection **113** and specifies the data type for a field value associated with each key. All the fields in the data source table have been imported into the MAILING_DBAG data bag **111**. This import data bag is created by the LOAD script command in step **81**, (FIG. **8**) using metadata definitions from the metadata database **38**.

FIG. **12** shows the internal storage of the export data bag **44**, defined in step **76**, which is used in the data transformation described with reference to FIG. **7**. The data group definition **122** is different than the data group definition **112** shown in FIG. **11**. The CITY_DBAG data bag **121** contains three of the original six fields from the MAILING_DBAG, the import **111** data bag, as well as a computed field that is a concatenation of the first and last names from the import data bag. The CITY_DBAG **12** export data bag is created by FORMAT script command in step **82**, using metadata definitions from the metadata database **38** (FIG. **2**). FIG. **9** shows part of Rule1, which is contained in RuleSet1 and defined in step **78**. The rule set in this example filters out all data group collection records in the MAILING_DBAG import data bag that have a city name of 'OTTAWA' and then writes those records into the CITY_DBAG export data bag.

FIG. **13** shows the internal storage of the export data target **33** for this example. The internal storage of a delimited flat file is shown. The export data connection **46**, defined in block **75**, refers to the exact location of the flat file CITY.CSV and indicates that the file is delimited. The export data view **45**, defined in step **77**, specifies that all the fields in the data bag will be exported to the delimited flat file **131**, defined in step **73**. The CITY.CSV flat file is created by the SAVE script command in step **83**, using metadata definitions from the metadata database **38**.

FIG. **14** shows a second import data example. The storage format of a personal information text file is shown. Each record contains a group at the end of the record, with repeating information about children of the specified person. This file definition will be used to illustrate the data storage of repeating group information in a data bag and the rule processing of the repeating group information during a data bag transformation. This file definition will be used to create the import data interface **32** used in this example.

FIG. **15** shows the internal storage of the text file defined in FIG. **14**. Each record contains a common set of fields before the 'CHILDREN' group. At the end of each record the 'CHILDREN' group may contain from zero to ten sets of 'child' information, consisting of the child's name and age. Each record is terminated by an end-of-record indicator appropriate to the computer system on which the file resides.

FIG. **16** shows the internal storage of the import data bag **43**, that contains the imported data of the text file shown in FIG. **15**. The data definition collection **162** now shows an example of a 'group' item type. The 'CHILDREN' group is defined as containing two fields, as specified by the two entries following the 'CHILDREN' group entry. The data group collection **163** shows how each record from the import text file, shown in FIG. **15**, is stored. The number of occurrences of the data group, defined by 'NBR_CHILDREN', must be stored so that the correct number of sets of the 'CHILDREN' group can be processed when manipulating the import data bag.

FIG. **17** shows an example rule created to transform the REPEATING_DBAG import data bag defined in FIG. **16**. This rule is one rule of a rule set. The rule will output the parent name, child name and child age for each input child whose age is less than 20. This example shows how a repeating information group can be manipulated within a data bag.

In the drawings and specification, there have been disclosed typical examples of the use of a preferred embodiment of the invention. Although specific terms have been employed to describe the preferred embodiment, they are used in a generic and descriptive manner only and not for purposes of limitation. The scope of the invention is set forth in the following claims.

We claim:

1. A distribution system for transforming and exchanging data between heterogeneous computer systems, comprising:

a) a systems interface for defining logical import and export data interfaces, data transformation rule sets and scripts;

b) a metadata database for storing said logical import and export data interfaces, data transformation rule sets and scripts;

c) a script processor for utilizing metadata from the metadata database to control data transformation within said systems interface and movement of said data into and out of said distribution system; and

d) a rule set processor responsive to said script processor for manipulating a data bag for storing imported data and a data bag for storing export data.

2. A distribution system as claimed in claim 1, wherein said systems interface comprises a configuration management user interface used by a user to define said logical import and export data interfaces, and create data transformation rule sets and scripts.

3. A distribution system as claimed in claim 2, wherein said logical import and export data interfaces comprise import and export data connections, import and export data views and said import and export data bags.

4. A distribution system as claimed in claim 3, wherein said logical import data interface is used to import data from an import data source into said distribution system.

5. A distribution system as claimed in claim 4, wherein said import data view is used during execution of said script processor to load data from said import data source into said import data bag.

6. A distribution system as claimed in claim 3, wherein said logical export data interface is used to export data in said data bag out to an export data target.

7. A distribution system as claimed in claim 6, wherein export data view of said export data bag is used during execution of said script processor to save data from said export data bag out to said export data target.

8. A distribution system as claimed in claim 1, wherein defined scripts stored in said metadata database are executed by said script processor.

9. A distribution system as claimed in claim 1, wherein said rule processor is invoked by said script processor to transform the import data bag into the export data bag based on predefined data transformation rules.

10. A distribution system as claimed in claim 1 wherein said script processor makes use of a script control language for controlling data transformation within said system interface and movement of said data into and out of said distribution system.

11. A distribution system as claimed in claim 10, wherein said script control language comprises a set of script commands and a script command processor to process and execute each of a number of script command lines.

12. A distribution system as claimed in claim 11, wherein said set of script commands comprises a load command to load data into an import data bag from an import data connection; a sort command for sorting data in a data bag into a different order; a merge command for merging together data in a number of data bags; an append command for appending data from one data bag into another data bag; a copy command for copying one data bag into another data bag; a join command for joining two or more data bags into another data bag; a format command for formatting a data bag into another data bag using a defined rule set; and a save command for saving data from an export data bag out to an export data connection.

13. In a distribution system for transforming and exchanging data between heterogeneous computer systems, a method of controlling data transformation within said distribution system, comprising the steps of:

a) operating a script processor that utilizes metadata stored in a metadata database to control the loading of data into an import data bag from a logical import data interface and performing any one or more of the following steps to convert the data to a desired format in an export data bag;

1) sorting said data according to a predetermined order;

2) merging data from a number of data bags into one data bag;

3) appending data from a first data bag into another data bag of the same type;

4) copying data from a first data bag into another data bag of the same type;

5) joining data from two or more data bags into another data bag using a specified key;

6) formatting data from a data bag into another data bag of a different type, using a defined rule set; and

b) saving the data in the export data bag out to an export data connection.

14. A method as claimed in claim 13 wherein said logical import data interface comprises import data connections, import data views and said import data bag.

15. A method as claimed in claim 14 wherein said logical import data interface is used to import data from an import data source into said distribution system.

16. A method as claimed in claim 15 wherein said import data view is used during execution of a script processor to load data from said import data source into said import data bag.

17. A computer readable memory for transforming and exchanging datastore data between heterogeneous computer systems using different datastore formats for storing similar information, comprising:

a) executable code for providing a systems interface for defining logical import and export data interfaces, data transformation rule sets and scripts;

b) executable code for providing a script processor for utilizing metadata from a metadata database to control data transformation within said systems interface and movement of said data into and out of said distribution system; and

c) executable code for providing a rule set processor responsive to said script processor for manipulating a data bag for storing imported data and a data bag for storing export data.

18. A computer readable memory as claimed in claim 17 wherein the metadata database stores logical import and export data interfaces, data transformation rule sets and scripts executed by the script processor.

19. A computer readable memory as claimed in claim 17 further comprising a script control language used by the script processor to control data transformation within said system interface and movement of said data into and out of said distribution system.

* * * * *