PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification ^{6} : G06F 17/30 | A2

(11) International Publication Number: **WO 99/56230**

(43) International Publication Date: 4 November 1999 (04.11.99)

(21) International Application Number: PCT/US99/09442

(22) International Filing Date: 29 April 1999 (29.04.99)

(30) Priority Data:

60/083,715	30 April 1998 (30.04.98)	US
09/203,925	2 December 1998 (02.12.98)	US

(71) Applicant: EC CUBED, INC. [US/US]; Suite 310, 15 River Road, Wilton, CT 06897 (US).

(72) Inventors: REDDY, Sathish; 93 Richards Avenue #810, Norwalk, CT 06853 (US). RANGARAJAN, Shridhar; 970 Hope Street #2D, Stamford, CT 06907 (US). KAREDDY, Vidyadhar; 93 Richards Avenue, Norwalk, CT 06854 (US). HOQUE, Faisal; 96 Glenbrook Road #38, Stamford, CT 06902 (US).

(74) Agent: COHEN, Neil, G.; Cummings & Lockwood, Four Stamford Plaza, Stamford, CT 06904 (US).

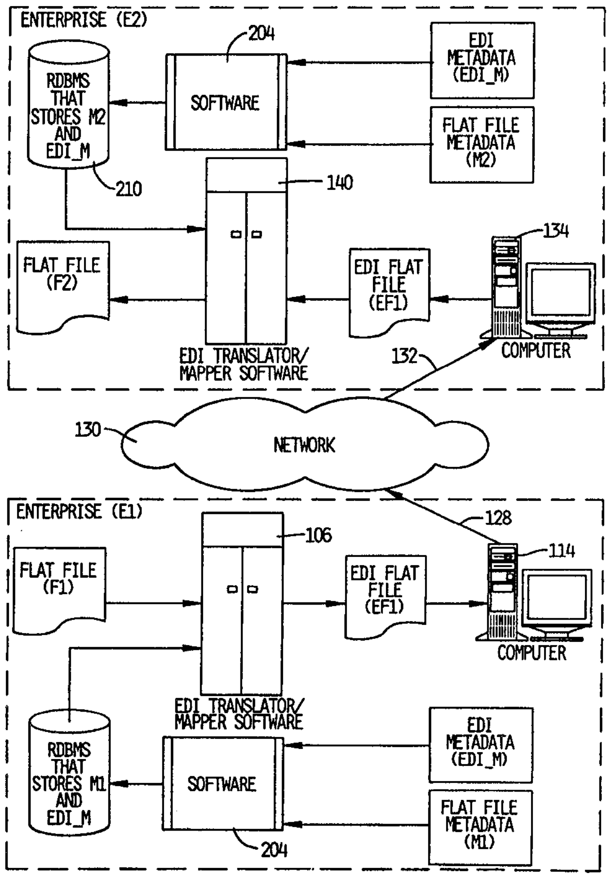
(81) Designated States: CN, IN, JP, RU, European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published
Without international search report and to be republished upon receipt of that report.

(54) Title: METHOD AND SYSTEM FOR STORING METADATA IN A RELATIONAL DATABASE

(57) Abstract

Diclosed herein is a method and system for storing metadata in a relational database, wherein the metadata describes data in a flat file. As described in one aspect of the disclosure, the metadata comprises (i) a header specification describing a general content of the data stored in the flat file, (ii) a record specification describing characteristics of a plurality of types of records contained in the flat file, and (iii) a field specification describing characteristics of fields of a record. The metadata also has a relation specification associated therewith indicating a relation between the plurality of types of records. The method includes creating a relational database and creating data structures for storing data representing the header specification, the record specification, the field specification, and the relation specification in the relational database. The method also includes storing data representing the header specification, the record specification, the field specification, and the relation specification in the data structures.



METHOD AND SYSTEM FOR STORING METADATA IN A RELATIONAL DATABASE

CROSS REFERENCE TO RELATED APPLICATION

This application claims priority to United States Provisional Application Serial Number 60/083,715, filed on April 30, 1998, the disclosure of which is incorporated herein by reference.

BACKGROUND OF THE INVENTION

The present invention relates generally to data storage and more specifically, to the storage of metadata in a relational database, which metadata describes data stored in a flat file.

Enterprises use electronic commerce (e-commerce) to conduct business transactions over electronic mediums. The electronic mediums include, for example, computer networks (e.g., the Internet), telephones, and facsimile machines.

Enterprises usually exchange business documents when conducting business transactions over a computer network. The business documents may include a request for bid ("RFB"), request for quote ("RFQ"), purchase order ("PO"), shipping information, and the like. The business documents are exchanged over the computer network in an electronic form.

A "flat file" is a data repository that contains information in an electronic form. There are several different types of flat files. A first type of flat file is a "fixed-width" flat file, which contains records having fields that are of a fixed-width. The records in a fixed-width flat file are referred to as "fixed-width records." Records in a fixed-width flat file share a common structure that is defined by the fields of the records.

A second type of flat file is a "delimited" flat file, which contains records having fields whose widths may vary. The records in a delimited flat file are referred to as "delimited records." Delimited records use a field delimiter (e.g., a predefined character) to separate the fields of a record. Records in delimited flat files share a common structure that is defined by the fields of the records.

A third type of flat file is a "fixed-block" flat file, which contains two or more different types of fixed-width records. Each type of fixed-width record is referred to as a block. In a fixed-block flat file, each type of fixed-width record has its own structure that is defined by the fields of that type of record. Fixed-block flat files are the most common types of files used in e-commerce.

A fourth type of flat file is a "delimited-block" file, which contains two or more different types of delimited records. Each type of delimited record is also referred to as a block. In a delimited-block flat file, each type of delimited record has its own structure that is defined by the fields of that type of record. Similar to delimited flat files, a field delimiter is used to separate the fields of a record in a delimited-block flat file.

A fifth type of flat file is known as a "general-block" flat file. A general-block flat file contains both fixed-width and delimited records. The fixed-width records may be of two or more different types (i.e., have different structures), as may the delimited records. Thus, a general-block file is the most complex type of flat file because it may store records normally contained in fixed-block and delimited-block flat files. Records of a general-block flat file that have a common structure are referred to as a block.

FIG. 1 illustrates six records 300(1)-300(6) contained in an exemplary general-block flat file 300. While each record in a flat file typically appears on a single line, it is not uncommon for two or more records to appear on the same line, as illustrated by records 300(5) and 300(6). When two or more records appear on the same line, a row delimiter (e.g., a predefined character such as a vertical bar 302) is used to separate the records. Records 300(2)-300(4) are fixed-width records. Records 300(1), 300(5), 300(6) are delimited records having comma 305, semicolon 307, and semicolon 309 field delimiters, respectively.

A flat file typically has a description associated with it that defines the structure and content of the flat file. This description, which is known as metadata, is necessary in order to be able to identify and parse the contents of the flat file. The metadata for two or more flat files may differ depending on the structure and content of the flat files. For example, the metadata of a purchase order document may differ from the metadata of a RFQ or a RFB. Additionally, the metadata of the same type of business document (e.g., two purchase orders) may differ between two enterprises that use that type of business document. In this latter case, for a second enterprise to read the purchases order of a first enterprise, the purchase order must be translated into a form that the second enterprise can parse.

To illustrate, consider the e-commerce architecture illustrated in FIG. 2. in which a first enterprise E1 desires to submit a purchase order to a second enterprise E2. Enterprise E1 operates a computer 114 that is able to communicate with computer network 130 over a communication channel 128. Computer 114 communicates with Electronic Document Interchange ("EDI") translator/mapper software running on a computer 106. Enterprise E2 operates a computer 134 that is able to communicate with computer network 130 over a communication channel 132. Computer 134 communicates with EDI translator/mapper software running on a computer 140. The computer network 130 may be a local area network ("LAN"), a wide area network ("WAN"), the Internet, an Intranet, or some other communications network that allows enterprise E1 to communicate with enterprise E2.

Enterprise E1 maintains a flat file F1 that contains the purchase order information. Flat file F1 is typically stored in a data repository. The purchase order information is stored in one particular format that is described by metadata M1. In this prior art example, flat file F1 will be translated to an industry-standard document, such as an EDI flat file EF1, which is described by EDI metadata EDI_M. As is well known, industry standards specify the metadata for different types of business documents to establish a common ground for identifying and parsing these documents by different enterprises. In the prior art, metadata M1 and EDI metadata EDI_M are typically stored as flat files in a data repository 118, which causes a number of problems as will be described below.

To perform the translation, EDI translator/mapper software running on computer 106 accesses flat file F1, metadata M1, and EDI metadata EDI_M. With regard to the metadata, the software typically has a proprietary module that has been programmed to access the metadata. The EDI translator/mapper software uses flat file F1, metadata M1, and metadata EDI_M to translate flat file F1 into EDI flat file EF1 in a well-known manner. Computer 114 receives EDI flat file EF1 from computer 106 and transmits EDI flat file EF1, over computer network 130, to computer 134 residing in enterprise E2.

Enterprise E2 will store the purchase order as a flat file F2. The format of flat file F2 differs from the format in which enterprise E1 maintains the purchase order. In particular, the format in which enterprise E2 maintains the purchase order is described by metadata M2, which differs from metadata M1. In order for enterprise E2 to identify and parse the contents of the purchase order represented by flat file F1 (which was sent to enterprise E2 as EDI flat file EF1), enterprise E2 must translate EDI flat file EF1 into flat file F2.

To do so, enterprise E2 stores metadata M2 and metadata EDI_M as flat files in data repository 146. Again, as will be described below, the storage of metadata in flat files causes a number of problems in the prior art. EDI translator/mapper software running on computer 140 accesses flat file EF1, metadata M2, and EDI metadata EDI_M. With regard to the metadata, the software typically has a proprietary module that has been programmed to access the metadata. The EDI translator/mapper software uses flat file EF1, metadata M2, and metadata EDI_M to translate flat file EF1 into flat file F2 in a well-known manner. The output of the EDI translator/mapper software is flat file F2. Thus, the purchase order information represented by flat file F2 is in a format that enterprise E2 can use.

The prior art techniques that store metadata in flat files suffer from number of problems. In particular, the structure of the flat files used to store metadata may be quite complex. Thus, special, non-standard, proprietary translator/mapper software is required to access and use the metadata. In some circumstances, such as when metadata is used to describe delimited-block and/or general-block files, such software is not known to the inventors as having been developed.

Further, data repositories (e.g., 118 and 146) that store the metadata may not be persistent. Thus, the time taken to access the information in the data repositories may be long, which may hinder performance.

Still further, the data repositories that store the metadata are not always secure. Therefore, the security of the information contained in these data repositories may be severely compromised.

In view of the foregoing, what is needed is a method and system that stores and permits quick and secure access to metadata that describes even the most complex flat files.

SUMMARY OF THE INVENTION

A first aspect of the present invention is directed to a method for storing metadata in a relational database, wherein the metadata describes data in a flat file. The metadata comprises (i) a header specification describing a general content of the data stored in the flat file, (ii) a record specification describing characteristics of a plurality of types of records contained in the flat file, and (iii) a field specification describing characteristics of fields of a record. The metadata also has a relation specification associated therewith indicating a relation between the plurality of types of records.

The method comprises creating a relational database and creating data structures for storing data representing the header specification, the record specification, the field specification, and the relation specification in the relational database. The method also includes storing data representing the header specification, the record specification, the field specification, and the relation specification in the data structures.

A second aspect of this invention relates to a method for storing metadata in a relational database, wherein the metadata describes data in a flat file. The metadata comprises (i) a header specification describing a general content of the data stored in the flat file, (ii) a record specification describing characteristics of a plurality of types of records contained in the flat file, and (iii) a field specification describing characteristics of fields of a record. The metadata has a relation specification associated therewith indicating a relation between the plurality of types of records.

The method includes creating a first data structure for storing data representing the header specification and storing the data representing the header specification in the first data structure. The method also includes creating a second data structure for storing data representing the record specification and storing the data representing the record specification in the second data structure. The method further includes creating a third data structure for storing data representing the field specification and storing the data representing the field specification in the third data structure. Still further, the method includes creating a fourth data structure for storing data representing the relation specification and storing the data representing the relation specification in the fourth data structure.

A third aspect of the present invention is directed to a method for storing metadata in a relational database, wherein the metadata describes data in a flat file. The metadata includes (i) a header specification describing a general content of the data stored in the flat file, (ii) a record specification describing characteristics of a plurality of types of records contained in the flat file, and (iii) a field specification describing characteristics of fields of a record. The metadata has a relation specification associated therewith indicating a relation between the plurality of types of records, and the relational database has a database engine associated therewith.

The method includes obtaining data representing the header specification, constructing a first database query relating to the data representing the header specification, and passing the first database query to the database engine for execution so that the header specification can be stored in the relational database. The method also includes obtaining data representing the record specification, constructing a second database query relating to the data representing the record specification, and passing the second database query to the database engine for execution so that the record specification can be stored in the relational database. The method further includes obtaining data representing the field specification, constructing a third database query relating to the data representing the field specification, and passing the third database query to the database engine for execution so that the field specification can be stored in the relational database. Still further, the method includes obtaining data representing the relation specification, constructing a fourth database query relating to the data representing the relation specification, and passing the fourth database query to the database engine for execution so that the relation specification can be stored in the relational database.

A fourth aspect of the present invention pertains to a method for translating a first file containing data in a first format into a second file containing data in a second format, wherein the first format differs from the second format. The method includes populating a relational database with metadata relating to the first file. The method also includes using translating software to access the data in the first file and the metadata in the relational database and translate the first file into the second file using the metadata and the data in the first file.

A fifth aspect of this invention is directed to a system for storing metadata in a relational database, wherein the metadata describes data in a flat file. The metadata comprises (i) a header specification describing a general content of the data stored in the flat file, (ii) a record specification describing characteristics of a plurality of types of records contained in the flat file, and (iii) a field specification describing characteristics of fields of a record. The metadata has a relation specification associated therewith indicating a relation between the plurality of types of records.

The system includes a relational database and a processor in communication with the relational database. The processor is programmed to create data structures for storing data representing the header specification, the record specification, the field specification, and the relation specification in the relational database. The processor is also programmed to store data representing the header specification, the record specification, the field specification, and the relation specification in the data structures.

BRIEF DESCRIPTION OF THE FIGURES

Representative embodiments of the present invention will be described with reference to the following figures:

FIG. 1 illustrates an exemplary general-block flat file.

FIG. 2 is a block diagram illustrating a prior art e-commerce architecture.

FIG. 3 is a block diagram illustrating an e-commerce architecture in which the present invention may be used.

FIG. 4 illustrates metadata that describes the structure and content of the general-block flat file shown in FIG. 1.

FIG. 5 illustrates a relational database for storing the metadata illustrated in FIG. 4 and relation specifications.

FIG. 5A is a flowchart illustrating a process for creating data structures for storing metadata and relation specifications.

FIG. 6 is a flowchart illustrating a process for processing and storing metadata in a relational database.

FIG. 7 is a flowchart illustrating a process for processing and storing a header specification in a relational database.

FIG. 8 is a flowchart illustrating a process for processing and storing a record specification in a relational database.

FIG. 9 is a flowchart illustrating a process for processing and storing a field specification in a relational database

FIG. 10 is a flowchart illustrating a process for processing and storing a relation specification in a relational database.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

Reference is now made to the accompanying Figures for the purpose of describing, in detail, the preferred embodiments of the present invention. The Figures and accompanying detailed description are provided as examples of the invention and are not intended to limit the scope of the claims appended hereto.

The present invention, as defined by the claims, provides a novel and unique method and system for storing metadata in, and using metadata from, a relational database, which metadata describes data stored in a flat file. In so doing, the method and system permits standard database software to be used to securely and quickly access metadata that describes even the most complex flat files.

Referring now to the figures wherein like reference numerals identify similar elements, FIG. 3 is a block diagram illustrating an e-commerce architecture in which the present invention may be used. In particular, a first enterprise E1 may transfer flat file business documents to an enterprise E2. Enterprise E1 operates a computer 114 that is in communication with a computer network 130 over a communication channel 128. Computer 114 communicates with translator/mapper software running on a computer 106 having one or more processors as is well known. Enterprise E2 operates a computer 134 that is in communication with computer network 130 over a communication channel 132. Computer 134 communicates with translator/mapper software running on a computer 140 having one or more processors as is well known.

In one embodiment, the translator/mapper software running on computers 106 and/or 140 is well-known Electronic Document Interchange ("EDI") translator/mapper software such as IBM Corp.'s (Armonk, NY) "Data Interchange/MVS" and/or "Data Interchange/MVS-CICS" software. Of course, other translator/mapper software may be chosen as desired to suit particular file formats used by enterprises E1 and E2.

While the particular type of computer network 130 is not important to the present invention, according to one embodiment, computer network 130 is the Internet. However, computer network 130 may be a local area network ("LAN"), a wide area network ("WAN"), an Intranet, or some other communications network that allows enterprise E1 to communicate with enterprise E2. Communication channels 128 and 132 may be wired or wireless.

Enterprise E1 maintains a flat file F1, which is to be transferred to enterprise E2 for processing. In one embodiment, flat file F1 represents a purchase order business document indicating goods and/or services that enterprise E1 desires to purchase from enterprise E2. In alternate embodiments, flat file F1 may represent any other document or business document to be used by enterprise E2 for any purpose.

Enterprise E1 stores flat file F1 in a data repository. The purchase order represented by flat file F1 is stored in one particular format that is described by metadata M1.

Enterprise E2 will maintain the purchase order as a flat file F2, but in a format that differs from the format in which enterprise E1 maintains the purchase order. The format in which enterprise E2 maintains the purchase order is described by metadata M2, which differs from metadata M1. In an alternative embodiment, metadata M1 and M2 may be similar.

In order for enterprise E2 to identify and parse the contents of the purchase order represented by flat file F1, flat file F1 is translated into flat file F2. The translation process uses an intermediate file format, which, in the present embodiment, is an Electronic Data Interchange ("EDI") format. Of course, other intermediate file formats may be used as desired. Thus, flat file F1 will be translated into an EDI flat file EF1, which will be sent to enterprise E2 over computer network 130. Enterprise E2 will translate EDT flat file EF1 into flat file F2, which enterprise E2 can parse.

The metadata that describes EDI flat file EF1 is described by EDI metadata EDI_M, which is maintained by enterprises E1 and E2. Enterprise E1 stores metadata M1 and EDI metadata EDI_M in a relational database 500. Similarly, enterprise E2 stores metadata M2 and EDI metadata EDI_M in a relational database 210. The storage of metadata is facilitated by computer software 204 as will be explained in more detail below.

The translation of flat file F1 into EDI flat file EF1 is performed by translator/mapper software running on computer 106, which accesses flat file F1, metadata M1, and EDI metadata EDI_M. Unlike the prior art, the metadata can be accessed from relational database 500 using standard database queries. The translator/mapper software uses flat file F1, metadata M1, and metadata EDI_M to translate flat file F1 into EDI flat file EF1 in a well-known manner. Computer 114 receives EDI flat file EF1 from computer 106 and transmits EDI flat file EF1, over the computer network 130, to computer 134 residing in enterprise E2.

The translator/mapper software running on computer 140 receives EDI flat file EF1 from computer 134. This software accesses metadata M2 and EDI metadata EDI_M from relational database 210. Because the metadata is stored in relational database 210, it may be accessed using standard database queries. The translator/mapper software uses flat file EF1, metadata M2, and metadata EDI_M to translate flat file EF1 into flat file F2 in a well-known manner. The output of the translator/mapper software is flat file F2. Thus, the purchase order represented by flat file F2, which corresponds to the purchase order represented by flat file F1, is in a format that enterprise E2 can parse and use.

FIG. 4 illustrates metadata that describes the structure and content of a flat file. In this example, the metadata of FIG. 4 describes the general-block flat file 300 (FIG. 1). Of course, metadata can be used to describe other flat files. General block flat file 300 may be considered as all or part of a purchase order requested by that file F1.

The metadata comprises a header specification 400 and block specifications 405, 410, and 415.

Header specification 400 describes the general content of a flat file, here, flat file 300. As shown in row R1, header specification 400 includes a HeaderName and a Description. In this example, the HeaderName is "CompanyInformation" and the description is "Company, Products, and Owner Information" as shown in row R2. Thus, header specification 400 indicates that the flat file contains information pertaining to companies, products that the companies manufacture, and the owner(s) of the companies.

A block specification is maintained for each type of record that is contained in a flat file. In the example shown in FIG. 4, block specifications 405, 410, and 415 each represent one of three different types of records in a flat file -- that is, block specifications 405, 410, and 415 represent a first type of delimited record, a first type of fixed-width record, and a second type of delimited record, respectively.

Each block specification comprises a record specification and a field specification. In this exemplary embodiment, block specifications 405, 410, and 415 comprise record/field specifications 405A/405B, 410A/410B, and 415A/415B, respectively.

A record specification describes the characteristics of a record. In the present embodiment, record specifications 405A, 410A, and 415A describe the characteristics of records represented by block specifications 405, 410, and 415.

In this example, each record specification 405A, 410A, and 410B includes a RecordName, Type, FieldDelimiter, and RowDelimiter as shown in rows R3, R13, and R21, respectively. The RecordNames of record specifications 405A, 410A, and 415A contain the "Company" (R4), "Product" (R14), and "Owner" (R22), respectively. This indicates that the "Company" record type contains information pertaining to a given company, the "Product" record type contains information pertaining to a given product of the Company, and the "Owner" record type contains information pertaining to a given owner of the company.

The Type of a record specification indicates a type of record represented by the record specification. Each Type is unique and need not be present when the flat file described by the metadata is a fixed-width or delimited flat file because they contain only one type of record. In the example shown in FIG. 4, record specifications 405A, 410A, and 415A contain the Types "0100" (R4), "0200" (R14), and "0300" (R22), respectively.

The FieldDelimiter of a record specification represents a character that separates two consecutive fields in the record. In fixed-width records, this character need not be present. In this example, records specifications 405A and 415A have a comma "," and a semicolon ";" as respective field delimiters as indicated in rows R4 and R22. Row R14 of record specification 410A does not have a field delimiter because it is a fixed-width record.

The RowDelimiter of a record specification represents the character that separates two records that have the same record specification. In this example, "NEWLINE" is a RowDelimiter for record specifications 405A and 410A as indicated in rows R4 and R 14, respectively. "NEWLINE" is used to indicate that any two consecutive records, which have the same record specification, appear on two consecutive lines in the file. A vertical bar "|" (R22) is shown as the exemplary RowDelimiter for record specification 415A.

A field specification describes the characteristics of the fields of a record. In this embodiment, field specifications 405B, 410B, 415B describe the characteristics of the fields of records described by record specifications 405A, 410A, and 415A, respectively. Each field specification 405B, 410B, and 415B includes a FieldName, DataType, Size, Position, Offset, and Null indicator as shown in rows R5, R15, and R23, respectively.

FieldName identifies the name of a field. DataType indicates the type of data that can be stored in the field. For example, DataTypes "A," "N," "AN," and "DT" indicate that alpha, numeric, alphanumeric, and date values can be stored in the field, respectively. Size indicates the maximum size of a value that may be contained in the field. Position indicates a logical position of the field in a record. Offset indicates that the value contained in the field begins in a certain column in the record (that is, the physical position in the record from where the value of a field begins). Null indicates whether or not the value of the field can be null.

As stated above, the storage of metadata in a relational database is facilitated by computer software 204, which controls one or more processor(s) of an appropriate computer in order to execute processes described herein. This is now described in more detail.

FIG. 5A illustrates a process for creating data structures in a relational database, which data structures will be used to store one or more header specifications, record specifications, field specifications, and relation specifications. As is well-known, a relation specification indicates a relation between two or more types of records. In one embodiment, a relation specification indicates a relation between two fields across two different record types in a flat file. As is well-known, the relation specification typically does not appear explicitly as part of the metadata, but may appear as input from a user.

At step 550, software 204 creates a relational database 500 (FIG. 5) in a Relational Database Management System (RDBMS). Any RDBMS software can be used as desired. This step 550 is executed in a well-known manner.

At step 552, software 204 creates a table 505 (FIG. 5) to store data representing header specification 400. In this embodiment, the columns in table 505 are the HeaderID, HeaderName, and Description. The HeaderID is the primary key column that identifies each row in table 505 uniquely. The manner in which the value contained in the HeaderID column is generated is specific to the RDBMS. The HeaderName and

Description columns of table 505 will store the HeaderName and Description contained in row R2 of header specification 400, respectively.

At step 554, software 204 creates a table 510 (FIG. 5) to store data representing record specifications. In this embodiment, a row is created for each record specification, here, 405A, 410A, 415A. The columns of table 510 include the RecordID, HeaderID, RecordName, Type, FieldDelimiter, and RowDelimiter. The RecordID is the primary key column that identifies each row in the table uniquely. The manner in which the value contained in the RecordID column is generated is specific to the RDBMS.

The HeaderID is the foreign key column that links table 510 to its parent table 505. This link establishes a relation between record specifications 405A, 410A, and 415A and header specification 400. The value contained in this column must be present in the HeaderID column of parent table 505. The RecordName, Type, FieldDelimiter and RowDelimiter columns of rows in table 510 having RecordIDs 1/2/3 will store the RecordName, Type, FieldDelimiter and RowDelimiter contained in rows R4/R14/R22 of record specifications 405A/410A/415A, respectively.

Software 204 creates a table 515 (FIG. 5) to store field specifications at step 556. In this embodiment, a row is created in table 515 for each field specification 405B, 410B, 415B. The columns of table 515 include FieldID, RecordID, FieldName, DataType, Size, Position, Offset, and Null.

The FieldID is the primary key column that identifies each row in the table uniquely. The manner in which the value contained in this column is generated is specific to the type of the RDBMS. The RecordID is the foreign key column that links this table 515 to its parent table 510. This link establishes a relation between field specifications 405B, 410B, 415B and its corresponding record specification 405A, 410A, 415A. The value contained in this column must be present in the RecordId column of parent table 510.

The FieldName, DataType, Size, Position, Offset, and Null columns of rows in table 515 having FieldIDs 1-16 will store the FieldName, DataType, Size, Position, Offset, and Null values contained in rows R6-R12/R16-R20/R24-27 of field specifications 405B/410B/415B, respectively. The "VARCHAR," "DATE," and "NUMERIC" data represents alphanumeric, numeric, and date data types in table 515. The values may vary depending on the RDBMS.

At step 558, software 204 creates table 520 to store a relation specification. The columns in table 520 include a RelationID, ParentColumnId, and ChildColumnID. The RelationID is the primary key column that identifies each row in table 520 uniquely. The manner in which the value contained in this column is generated is specific to the type of the RDBMS.

The ParentColumnID is the foreign key column that indicates the parent field in a relation specification. For example, the row in table 520 having RelationID 1 indicates that the RecIdentifier (FieldID 1 in table 515) is the parent column. This RecIdentifier field belongs to the "Company" record specification as indicated by RecordID 1 in table 510. The value contained in the ParentColumnID column must be present in the parent table 515.

The ChildColumnID is the foreign key column that indicates the child field in a relation specification. For example, the row in table 520 having RelationID 1 indicates that the RecIdentifier field (FieldID 8 in table 515) is the child column. This RecIdentifier field belongs to the Product record specification (RecordID 2 in table 510). The value contained in the ChildColumnID column must be present in the parent table 515.

The relation just described indicates that a single parent record (e.g., type "0100" of row R4 of FIG. 4) could have one or more child records of a different type (e.g., "0200" of row R14 of FIG. 4). This is similar to a primary-foreign key relation in a relational database. Therefore, record type "0100" (R4) shares a one-to-many relation with record type "0200" (R14).

The data representing the header specification, the record specifications, the field specifications, and the relation specification is stored in the data structures. The flowchart of FIG. 6 illustrates a preferred method, which begins at step 600.

At step 605, header specification 400 is input from a user to software 204. At step 610, software 204 processes and stores header specification 400 in table 505. The flowchart in FIG. 7, which begins at step 700, illustrates step 610 in more detail.

At step 710, software 204 constructs a program data-structure containing the dates shown in row R2 of header specification 400 (FIG. 4). At step 715, software 204 stores row R2 of header specification in the program data-structure created at step 710. At step 720, software 204 constructs a Structured Query Language (SQL) query to insert the data contained in the program data-structure into a row in table 505. At step 725, software 204 passes the SQL query to a database engine, which is part of the RDBMS. At step 730, the database engine executes the SQL query and inserts the data contained in the program data-structure into an appropriate row in table 505.

Referring again to FIG. 6, processing proceeds to step 615 where software 204 may check whether all record specifications 405A, 410A, 415A for the current header specification 400 have been stored in relational database 500. If all such record specifications have been stored, then processing continues at step 645.

If there are record specifications to store, then the next row R4/R14/R22 of record specification 405A/410A/415A are input from a user to software 204 at step 620. At step 625, the next record specification is stored in relational database 500.

The flowchart of FIG. 8 illustrates step 625 in detail. At step 810, software 204 constructs a program data-structure containing current record specification 405A/410A/415A. At step 815, software 204 stores the current record specification 405A/410A/415A in the program data-structure created at step 810. At step 820, software 204 constructs a Structured Query Language (SQL) query to insert the data contained in the program data-structure into a row in table 510. At step 825, software 204 passes the SQL query to the database engine for execution. At step 830, the database engine executes the SQL query and inserts the data contained in the current program data-structure into an appropriate row in table 510.

Referring again to FIG. 6, at step 630, software 204 may check whether all field specifications for the current record specification have been stored in table 515. If all field specifications for the current record specification have been stored in table 515, then processing returns to step 615 where the next record specification is processed.

If all field specifications for the current record specification have not been stored in table 515, then the next field specification (i.e., R6-R12/R16-R20/R24-R27 of record specifications 405B/410B/415B) is input from the user to software 204 at step 635. At step 640, software 204 processes and stores the current field specification in table 515. The flowchart in FIG. 9 illustrates this process in more detail.

At step 910, software 204 constructs a program data-structure containing current field specification 405B/410B/415B. At step 915, software 204 stores the current field specification the program data-structure created at step 910. At step 920, software 204 constructs a Structured Query Language (SQL) query to insert the data contained in the program data-structure into a row in table 515. At step 925, software 204 passes the SQL query to the database engine for execution. At step 930, the database engine executes the SQL query and inserts the data contained in the program data structure into an appropriate row in table 515.

At step 645, software 204 may check whether all of the relation specifications for the current header specification 400 have been stored in table 520. If all of the relation specifications for the current header specification 400 have been stored in table 520, then processing is done at step 660. At that point, metadata for the flat file has been stored in the relational database 500. It is noted that the metadata for other flat files can be stored in a relational database in a similar manner.

If all of the relation specifications for the current header specification 400 have not been stored in table 520, then processing continues at step 655 where software 204 processes and stores the relation specification in table 520. The flowchart of FIG. 10 illustrates the method in more detail. At step 1010, software 204 constructs a program data-structure containing the relation specification, which includes a RelationId, ParentColumnID, and the ChildColumnID. The values of these columns may be obtained from the FieldID column in table 515 and input by a user to the software 204 at step 650.

At step 1015, software 204 stores the ParentColumnID and the ChildColumnID in the program data-structure created at step 1010. At step 1020, software 204 constructs a Structured Query Language SQL query to insert the data contained in the program data-structure as a row in table 520. At step 1025, software 204 passes the SQL query to the database engine for execution. At step 1030, the database engine executes the SQL query and inserts the data contained in the program data structure into an appropriate row in table 520. Processing returns to step 645 and may end at step 660. The metadata and relation specification stored in relational database 500 may then be used by translator/mapper software as described above.

While the foregoing described the storage of the metadata of FIG. 4 in relational database 500, it should be appreciated that the processes of FIGS. 5A-10 can be used to store any metadata in an appropriate relational database.

It is noted that while the foregoing description has referred to specific individual databases, formats, structures, records, and fields, those skilled in the art will readily appreciate that various modifications and substitutions may be made thereto without departing from the spirit and scope of the present invention.

In view of the foregoing, it is apparent that the present invention offers advantages over the prior art. In particular, with the present invention, the metadata is stored in a standard format in relational databases. Therefore, the metadata may be accessed using industry-standard, non-proprietary database queries such as those offered in Structured Query Language (SQL). Moreover, relational databases have a standard security protocol built into the system, which ensures that the metadata remains secure.

Still further, existing software tools enable storage of metadata in a variety of data repositories, although not in relational databases. Further, they mostly cater to only fixed-width and delimited files. While a small number cater to fixed-block files, such tools do not handle metadata that describe very generic types of flat files such as delimited-block files or general-block files. From that respect, the present invention is highly generic and therefore powerful because it is not restricted to storing metadata for any one single type of a flat file just described. It facilitates the storage of metadata that describe very general and complex files containing two or more different types of records that may be fixed-width, delimited, or both.

Although the particular embodiments shown and described above are useful in many applications relating to the arts to which the present invention pertains, further modifications of the present invention herein disclosed will occur to persons skilled in the art. All such modifications are deemed to be within the scope and spirit of the present invention.

What is claimed is:

1. A method for storing metadata in a relational database, wherein the metadata describes data in a flat file, wherein the metadata comprises (i) a header specification describing a general content of the data stored in the flat file, (ii) a record specification describing characteristics of a plurality of types of records contained in the flat file, and (iii) a field specification describing characteristics of fields of a record, wherein the metadata has a relation specification associated therewith indicating a relation between the plurality of types of records, and wherein the method comprises:

(a) creating a relational database;

(b) creating data structures for storing data representing the header specification, the record specification, the field specification, and the relation specification in the relational database; and

(c) storing data representing the header specification, the record specification, the field specification, and the relation specification in the data structures.

2. The method of Claim 1, wherein the step of creating the data structures comprises:

(a) creating a first table for storing the header specification;

(b) creating a second table for storing the record specification;

(c) creating a third table for storing the field specification; and

(d) creating a fourth table for storing the relation specification.

3. The method of Claim 1, wherein the step of storing data comprises:

(a) storing data representing the header specification in a first table;

(b) storing data representing the record specification in a second table;

(c) storing data representing the field specification in a third table;

(d) storing data representing the relation specification in a fourth table.

4. The method of Claim 1, further comprising the step of providing data indicating structures of the header specification, the record specification, the field specification, and the relation specification to database software, and wherein the step of creating the data structures comprises causing the database software to create the data structures based on the provided data indicating the structures.

5. The method of Claim 4, wherein the step of providing comprises providing data indicating the record specification that describes characteristics of fixed-width and delimited records stored in the flat file.

6. The method of Claim 4, wherein the step of providing comprises providing data indicating the record specification that describes characteristics of fixed-width records stored in the flat file.

7. The method of Claim 4, wherein the step of providing comprises providing data indicating the record specification that describes characteristics of delimited records stored in the flat file.

8. A method for storing metadata in a relational database, wherein the metadata describes data in a flat file, wherein the metadata comprises (i) a header specification describing a general content of the data stored in the flat file, (ii) a record specification describing characteristics of a plurality of types of records contained in the flat file, and (iii) a field specification describing characteristics of fields of a record, wherein the metadata has a relation specification associated therewith indicating a relation between the plurality of types of records, and wherein the method comprises:

(a) creating a first data structure for storing data representing the header specification;

(b) storing the data representing the header specification in the first data structure;

(c) creating a second data structure for storing data representing the record specification;

(d) storing the data representing the record specification in the second data structure;

(e) creating a third data structure for storing data representing the field specification;

(f) storing the data representing the field specification in the third data structure;

(g) creating a fourth data structure for storing data representing the relation specification; and

(h) storing the data representing the relation specification in the fourth data structure.

9. The method of Claim 8, further comprising the step of providing data indicating structures of the header specification, the record specification, the field specification, and the relation specification to database software, and wherein the steps of creating the first, second, third, and fourth data structures comprises causing the database software to create the first, second, third, and fourth data structures based on the data indicating the structures.

10. The method of Claim 9, wherein the step of creating the second data structure comprises creating a data structure for storing data representing fixed-width and delimited records stored in the flat file.

11. The method of Claim 9, wherein the step of creating the second data structure comprises creating a data structure for storing data representing fixed-width records stored in the flat file.

12. The method of Claim 9, wherein the step of creating the second data structure comprises creating a data structure for storing data representing delimited records stored in the flat file.

13. A method for storing metadata in a relational database, wherein the metadata describes data in a flat file, wherein the metadata comprises (i) a header specification describing a general content of the data stored in the flat file, (ii) a record specification describing characteristics of a plurality of types of records contained in the flat file, and (iii) a field specification describing characteristics of fields of a record, wherein the metadata has a relation specification associated therewith indicating a relation between the plurality of types of records, wherein the relational database has a database engine associated therewith, and wherein the method comprises:

(a) obtaining data representing the header specification;

(b) constructing a first database query relating to the data representing the header specification;

(c) passing the first database query to the database engine for execution so that the header specification can be stored in the relational database;

(d) obtaining data representing the record specification;

(e) constructing a second database query relating to the data representing the record specification;

(f) passing the second database query to the database engine for execution so that the record specification can be stored in the relational database;

(g) obtaining data representing the field specification;

(h) constructing a third database query relating to the data representing the field specification;

(i) passing the third database query to the database engine for execution so that the field specification can be stored in the relational database;

(j) obtaining data representing the relation specification;

(k) constructing a fourth database query relating to the data representing the relation specification; and

(l) passing the forth database query to the database engine for execution so that the relation specification can be stored in the relational database.

14. The method of Claim 13, wherein steps (b), (e), (h), and (k) comprise constructing a SQL query.

15. A method for translating a first file containing data in a first format into a second file containing data in a second format, wherein the first format differs from the second format, and wherein the method comprises:

(a) populating a relational database with metadata relating to the first file; and

(b) using translating software,

(i) accessing the data in the first file and the metadata in the relational database; and

(ii) translating the first file into the second file using the metadata and the data in the first file.

16. The method of Claim 15, wherein the step of translating comprises translating the first file into the second file having an electronic document interchange format.

17. A system for storing metadata in a relational database, wherein the metadata describes data in a flat file, wherein the metadata comprises (i) a header specification describing a general content of the data stored in the flat file, (ii) a record specification describing characteristics of a plurality of types of records contained in the flat file, and (iii) a field specification describing characteristics of fields of a record, wherein the metadata has a relation specification associated therewith indicating a relation between the plurality of types of records, and wherein the system comprises:

(a) a relational database; and

(b) a processor in communication with the relational database, wherein the processor is programmed to

(i) create data structures for storing data representing the header specification, the record specification, the field specification, and the relation specification in the relational database; and

(ii) store data representing the header specification, the record specification, the field specification, and the relation specification in the data structures.

18. The system of Claim 17, wherein the processor is programmed to

(a) create a first table for storing the header specification;

(b) create a second table for storing the record specification;

(c) create a third table for storing the field specification; and

(d) create a fourth table for storing the relation specification.

19. The system of Claim 17, wherein the processor is programmed to

(a) store data representing the header specification in a first table;

(b) store data representing the record specification in a second table;

(c) store data representing the field specification in a third table;

(d) store data representing the relation specification in a fourth table.

20. The system of Claim 17, wherein the processor is further programmed to provide data indicating structures of the header specification, the record specification, the field specification, and the relation specification to database software, and wherein the processor is programmed to cause the database software to create the data structures based on the data indicating the structures.

21. The system of Claim 17, wherein the data indicating the structures of the record specification comprises data indicating characteristics of fixed-width and delimited records stored in the flat file.

22. The system of Claim 17, wherein the data indicating the structures of the record specification comprises data indicating characteristics of fixed-width records stored in the flat file.

23. The system of Claim 17, wherein the data indicating the structures of the record specification comprises data indicating characteristics of delimited records stored in the flat file.

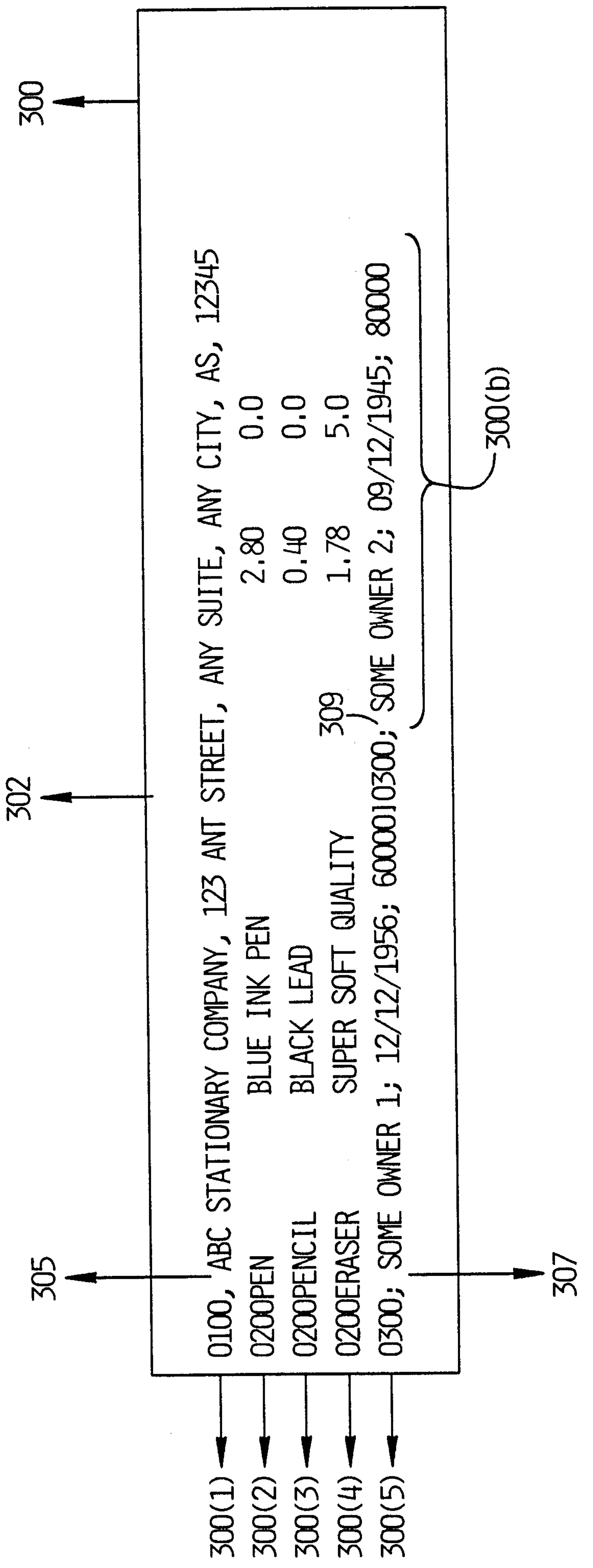

300
302
305
307
309
300(b)
0100, ABC STATIONARY COMPANY, 123 ANT STREET, ANY SUITE, ANY CITY, AS, 12345
0200PEN BLUE INK PEN 2.80 0.0
0200PENCIL BLACK LEAD 0.40 0.0
0200ERASER SUPER SOFT QUALITY 1.78 5.0
0300; SOME OWNER 1; 12/12/1956; 6000010300; SOME OWNER 2; 09/12/1945; 80000
300(1)
300(2)
300(3)
300(4)
300(5)

FIG_1
PRIOR ART

FIG_2
PRIOR ART

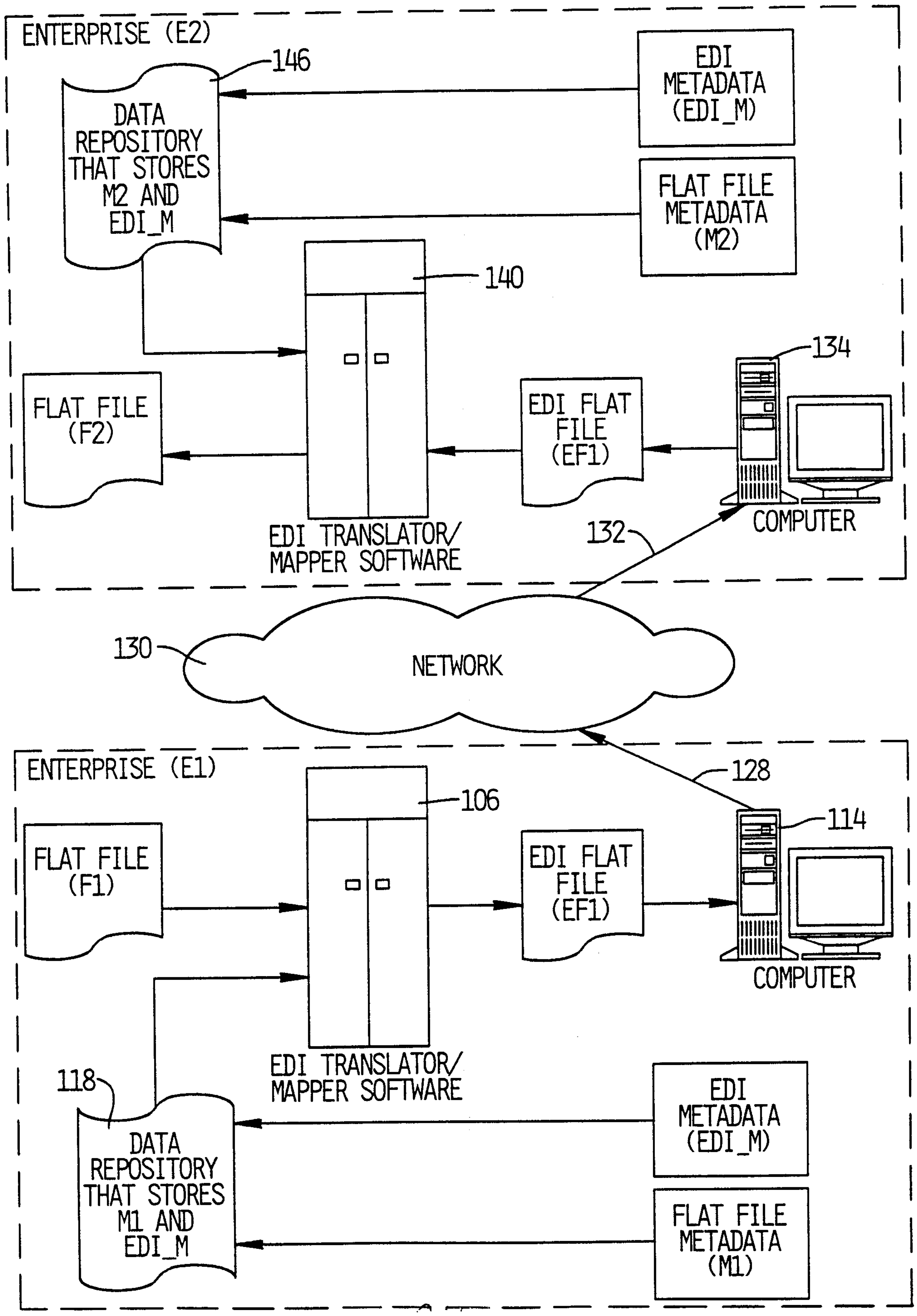


FIG_3

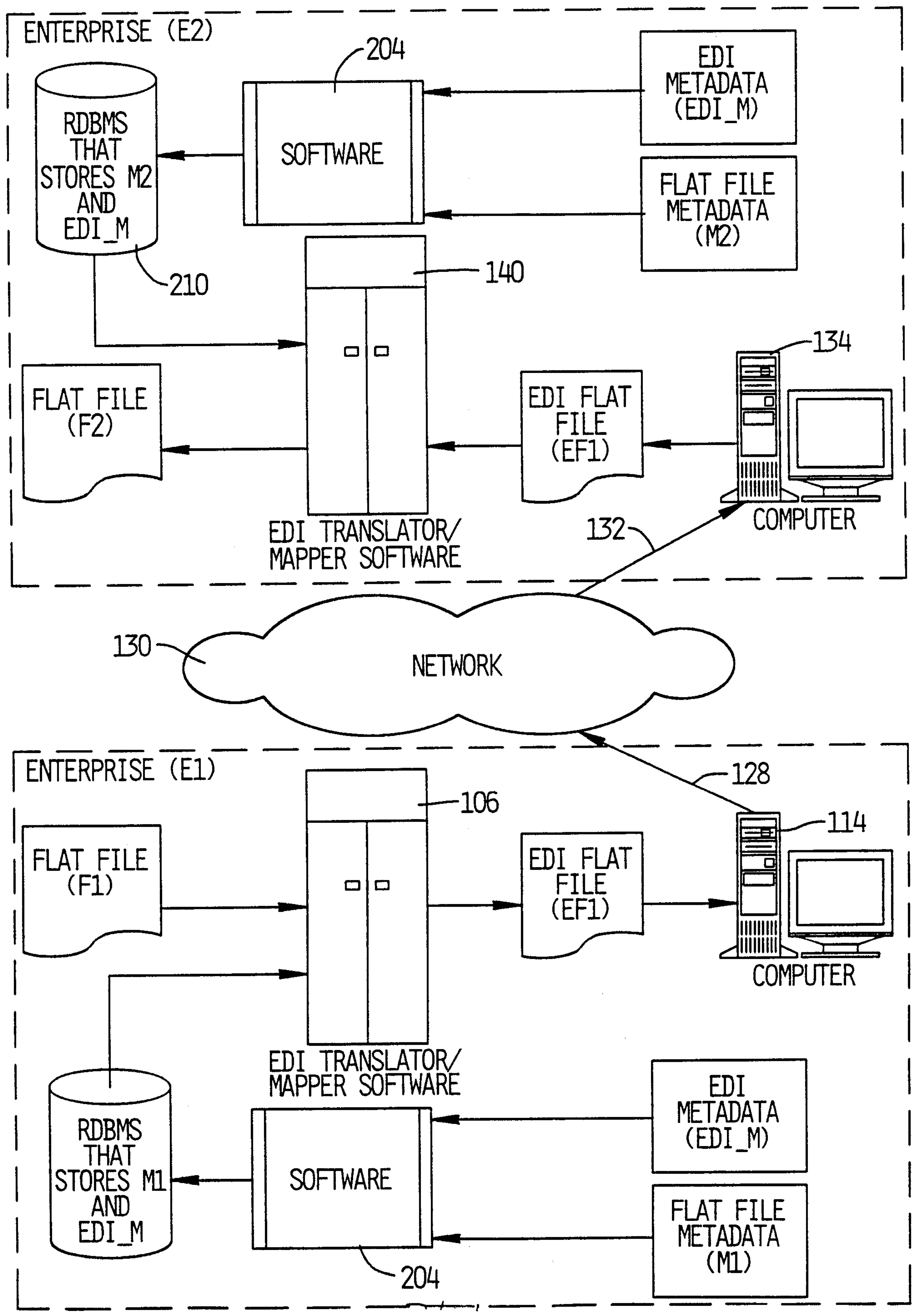

ENTERPRISE (E2)
204
EDI METADATA (EDI_M)
RDBMS THAT STORES M2 AND EDI_M
SOFTWARE
FLAT FILE METADATA (M2)
140
210
134
FLAT FILE (F2)
EDI FLAT FILE (EF1)
EDI TRANSLATOR/ MAPPER SOFTWARE
132
COMPUTER
130
NETWORK
ENTERPRISE (E1)
128
106
114
FLAT FILE (F1)
EDI FLAT FILE (EF1)
COMPUTER
EDI TRANSLATOR/ MAPPER SOFTWARE
EDI METADATA (EDI_M)
RDBMS THAT STORES M1 AND EDI_M
SOFTWARE
FLAT FILE METADATA (M1)
204

FIG_4

HEADER SPECIFICATION - 400

R1 HEADERNAME	DESCRIPTION
R2 COMPANYINFORMATION	COMPANY, PRODUCTS, AND OWNER INFORMATION

BLOCK SPECIFICATION - 405

RECORD SPECIFICATION - 405A

R3 RECORDNAME	TYPE	FIELDELIMITER	ROWDELIMITER
R4 COMPANY	"0100"	","	NEWLINE

FIELD SPECIFICATION-405B

R5 FIELDNAME	DATATYPE	SIZE	POSITION	OFFSET	NULL
R6 RECLDENTIFIER	AN	4	1	N/A	N
R7 NAME	AN	30	2	N/A	N
R8 ADDRESS 1	AN	25	3	N/A	Y
R9 ADDRESS 2	AN	25	4	N/A	Y
R10 CITY	A	15	5	N/A	Y
R11 STATE	A	2	6	N/A	Y
R12 ZIP	N	5	7	N/A	Y

BLOCK SPECIFICATION - 410

RECORD SPECIFICATION - 410A

R13 RECORDNAME	TYPE	FIELDELIMITER	ROWDELIMITER
R14 COMPANY	"0200"		NEWLINE

FIELD SPECIFICATION-410B

R15 FIELDNAME	DATATYPE	SIZE	POSITION	OFFSET	NULL
R16 RECLDENTIFIER	AN	4	1	1	N
R17 NAME	AN	20	2	5	N
R18 DESCRIPTION	AN	50	3	25	Y
R19 PRICE	N	10	4	75	Y
R20 DISCOUNT	N	6	5	85	Y

BLOCK SPECIFICATION - 415

RECORD SPECIFICATION - 415A

R21 RECORDNAME	TYPE	FIELDELIMITER	ROWDELIMITER
R22 OWNER	"0300"	";"	"I"

FIELD SPECIFICATION-415B

R23 FIELDNAME	DATATYPE	SIZE	POSITION	OFFSET	NULL
R24 RECLDENTIFIER	AN	4	1	N/A	N
R25 NAME	A	20	2	N/A	N
R26 DATEOFBIRTH	DT	15	3	N/A	Y
R27 CURRENTASSETS	N	15	4	N/A	Y

HEADER_SPECIFICATION - 505

HEADERID	HEADERNAME	DESCRIPTION
1	COMPANYINFORMATION	COMPANY, PRODUCTS, AND OWNER INFORMATION

RECORD_SPECIFICATION - 510

RECORDID	HEADERID	RECORDNAME	TYPE	FIELDELIMITER	ROWDELIMITER
1	1	COMPANY	"0100"	","	NEWLINE
2	1	PRODUCT	"0200"	NULL	NEWLINE
3	1	OWNER	"0300"	";"	"I"

FIELD_SPECIFICATION - 515

FIELDID	RECORDID	FIELDNAME	DATATYPE	SIZE	POSITION	OFFSET	NULL
1	1	RECLDENTIFIER	VARCHAR	4	1		N
2	1	NAME	VARCHAR	30	2		N
3	1	ADDRESS1	VARCHAR	25	3		Y
4	1	ADDRESS2	VARCHAR	25	4		Y
5	1	CITY	VARCHAR	15	5		Y
6	1	STATE	VARCHAR	2	6		Y
7	1	ZIP	VARCHAR	5	7		Y
8	2	RECLDENTIFIER	VARCHAR	4	1	1	N
9	2	NAME	VARCHAR	20	2	5	N
10	2	DESCRIPTION	VARCHAR	50	3	25	Y
11	2	PRICE	NUMERIC	10	4	75	Y
12	2	DISCOUNT	NUMERIC	6	5	85	Y
13	3	RECLDENTIFIER	VARCHAR	4	1		N
14	3	NAME	VARCHAR	20	2		N
15	3	DATEOFBIRTH	DATE	15	3		Y
16	3	CURRENTASSETS	NUMERIC	15	4		Y

RELATION_SPECIFICATION - 520

RELATIONID	PARENTCOLUMNID	CHILDCOLUMNID
1	1	8
2	1	13

FIG_5

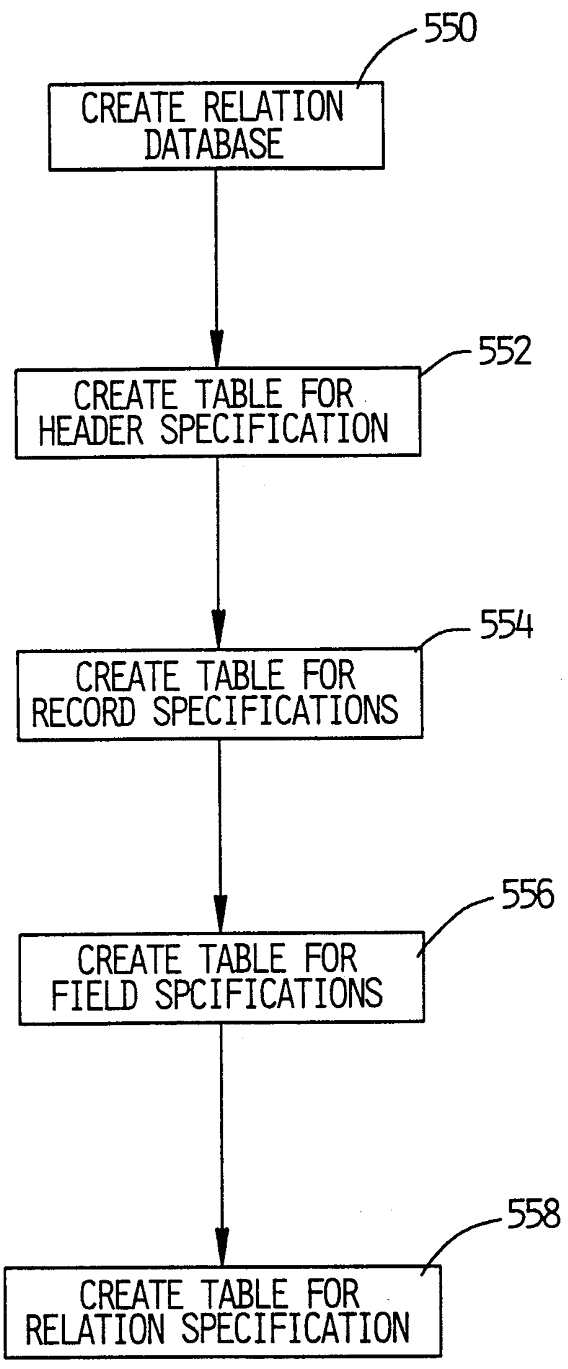

550
CREATE RELATION DATABASE
552
CREATE TABLE FOR HEADER SPECIFICATION
554
CREATE TABLE FOR RECORD SPECIFICATIONS
556
CREATE TABLE FOR FIELD SPCIFICATIONS
558
CREATE TABLE FOR RELATION SPECIFICATION

FIG_5A

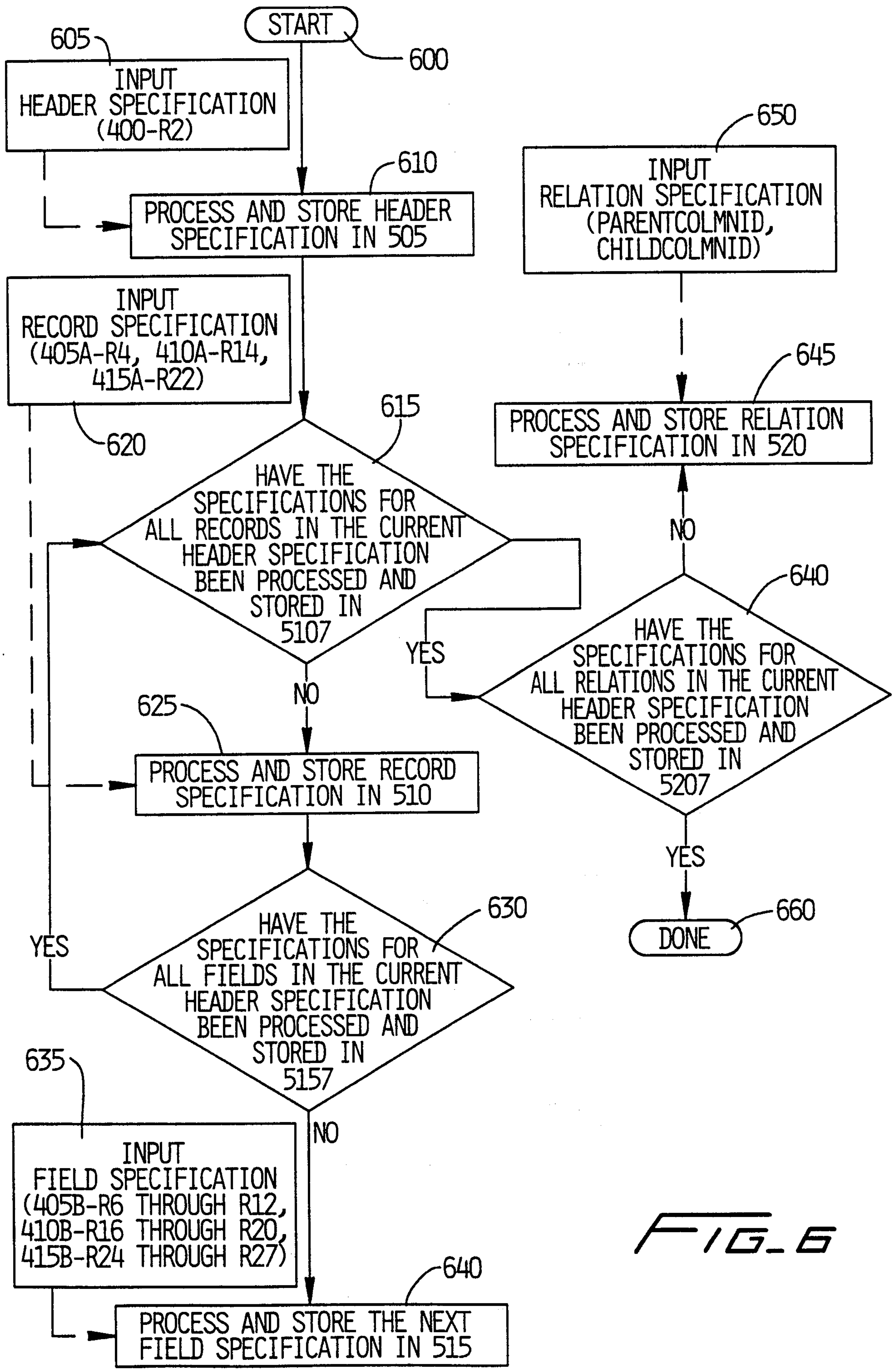

START
600
605
INPUT HEADER SPECIFICATION (400-R2)
610
PROCESS AND STORE HEADER SPECIFICATION IN 505
INPUT RECORD SPECIFICATION (405A-R4, 410A-R14, 415A-R22)
620
615
HAVE THE SPECIFICATIONS FOR ALL RECORDS IN THE CURRENT HEADER SPECIFICATION BEEN PROCESSED AND STORED IN 510?
NO
YES
625
PROCESS AND STORE RECORD SPECIFICATION IN 510
630
HAVE THE SPECIFICATIONS FOR ALL FIELDS IN THE CURRENT HEADER SPECIFICATION BEEN PROCESSED AND STORED IN 515?
YES
NO
635
INPUT FIELD SPECIFICATION (405B-R6 THROUGH R12, 410B-R16 THROUGH R20, 415B-R24 THROUGH R27)
640
PROCESS AND STORE THE NEXT FIELD SPECIFICATION IN 515
650
INPUT RELATION SPECIFICATION (PARENTCOLMNID, CHILDCOLMNID)
645
PROCESS AND STORE RELATION SPECIFICATION IN 520
NO
640
HAVE THE SPECIFICATIONS FOR ALL RELATIONS IN THE CURRENT HEADER SPECIFICATION BEEN PROCESSED AND STORED IN 520?
YES
660
DONE

FIG_6

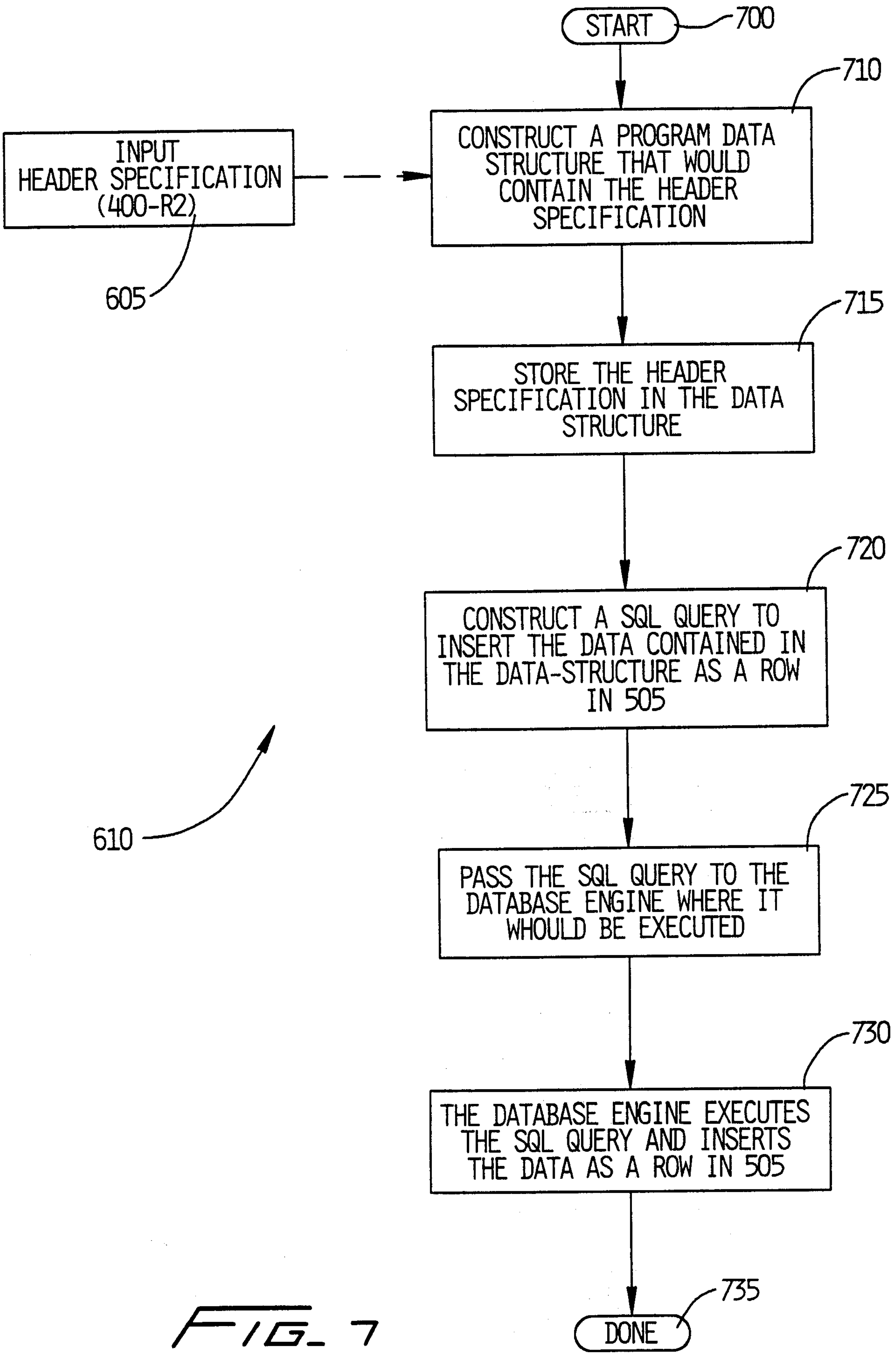

START
700
710
CONSTRUCT A PROGRAM DATA STRUCTURE THAT WOULD CONTAIN THE HEADER SPECIFICATION
INPUT HEADER SPECIFICATION (400-R2)
605
715
STORE THE HEADER SPECIFICATION IN THE DATA STRUCTURE
720
CONSTRUCT A SQL QUERY TO INSERT THE DATA CONTAINED IN THE DATA-STRUCTURE AS A ROW IN 505
610
725
PASS THE SQL QUERY TO THE DATABASE ENGINE WHERE IT WHOULD BE EXECUTED
730
THE DATABASE ENGINE EXECUTES THE SQL QUERY AND INSERTS THE DATA AS A ROW IN 505
735
DONE

FIG. 7

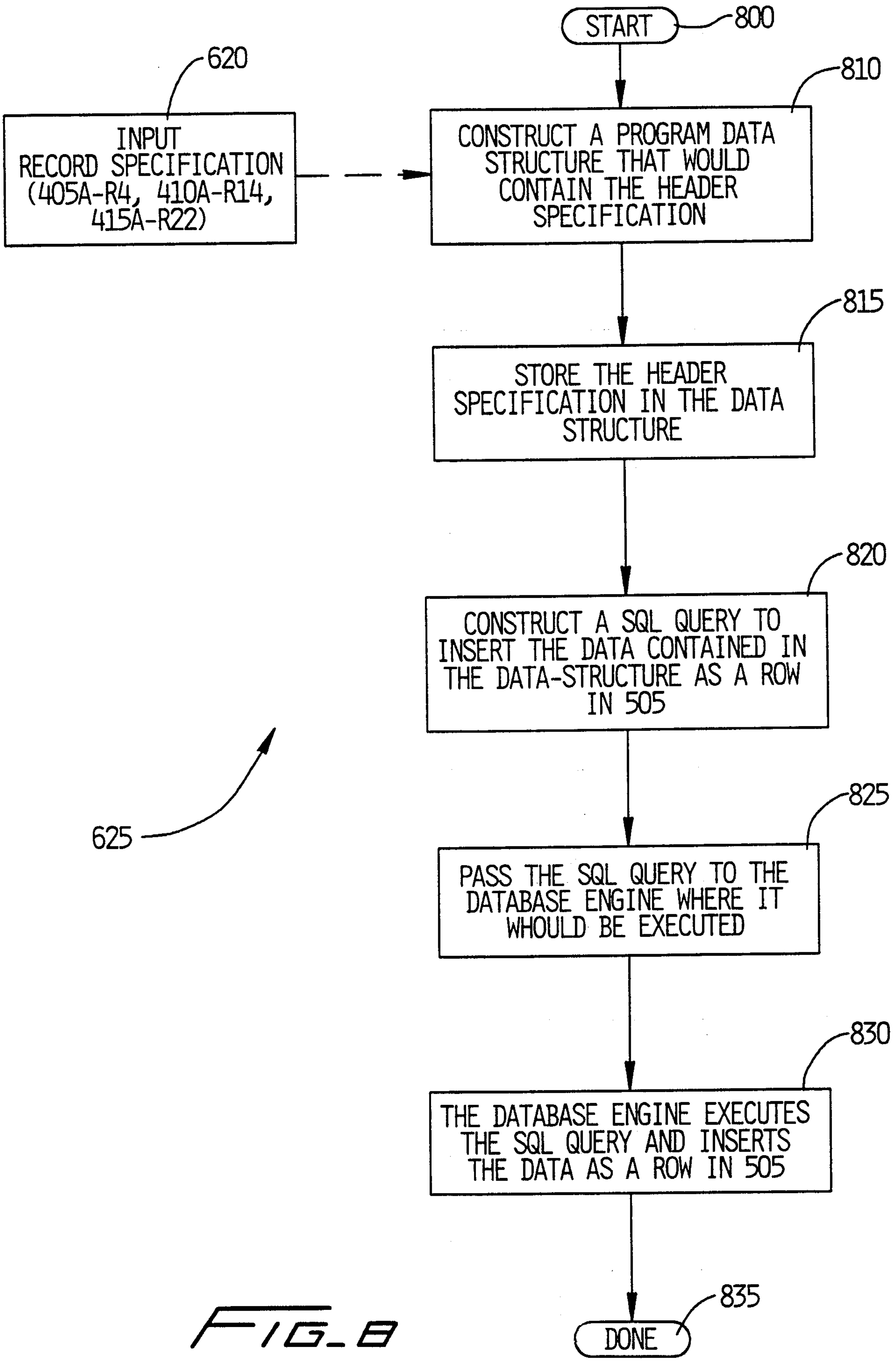

START
800
620
810
INPUT RECORD SPECIFICATION (405A-R4, 410A-R14, 415A-R22)
CONSTRUCT A PROGRAM DATA STRUCTURE THAT WOULD CONTAIN THE HEADER SPECIFICATION
815
STORE THE HEADER SPECIFICATION IN THE DATA STRUCTURE
820
CONSTRUCT A SQL QUERY TO INSERT THE DATA CONTAINED IN THE DATA-STRUCTURE AS A ROW IN 505
625
825
PASS THE SQL QUERY TO THE DATABASE ENGINE WHERE IT WHOULD BE EXECUTED
830
THE DATABASE ENGINE EXECUTES THE SQL QUERY AND INSERTS THE DATA AS A ROW IN 505
835
DONE

FIG_8

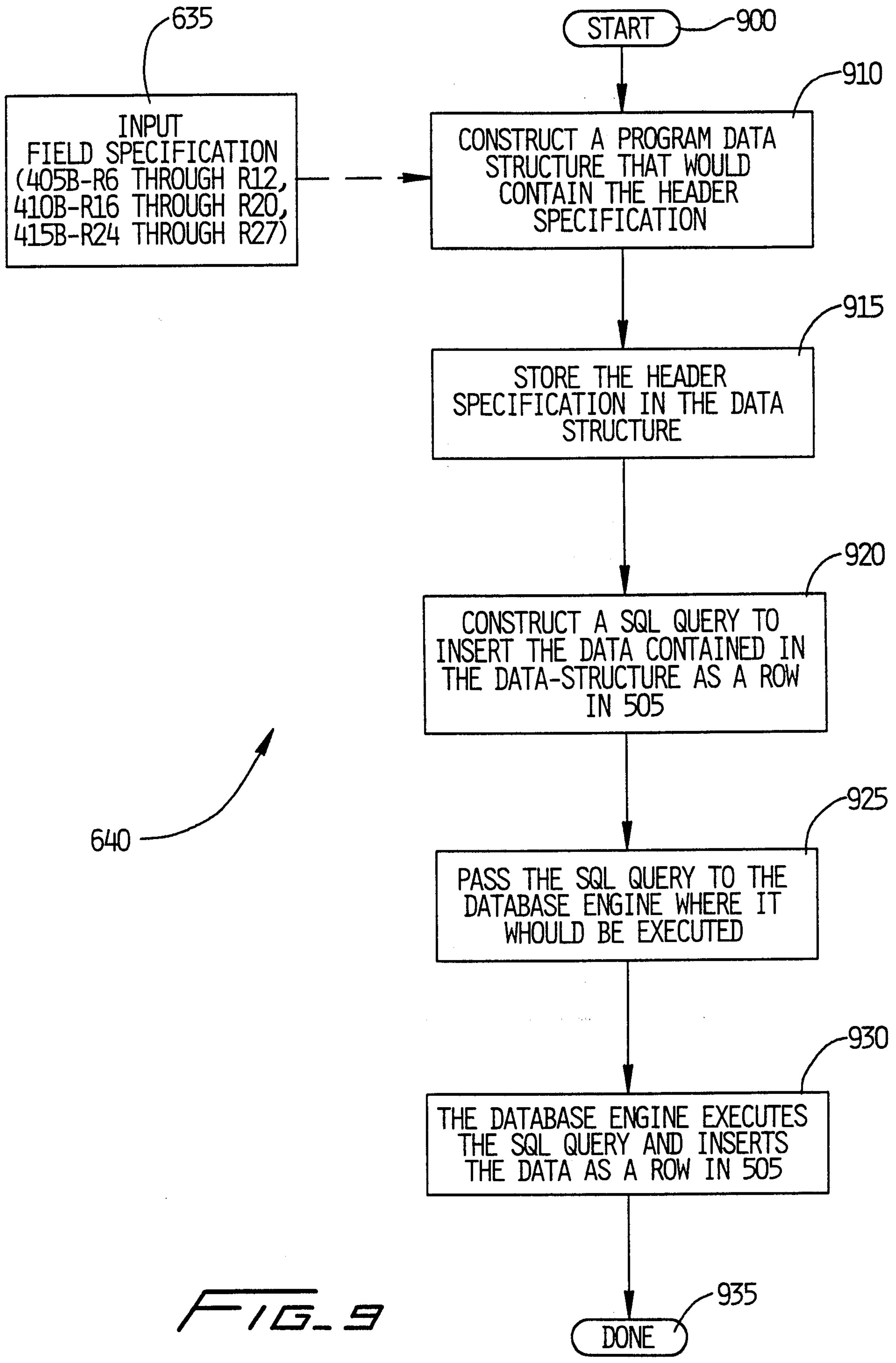

635
INPUT FIELD SPECIFICATION (405B-R6 THROUGH R12, 410B-R16 THROUGH R20, 415B-R24 THROUGH R27)
START
900
910
CONSTRUCT A PROGRAM DATA STRUCTURE THAT WOULD CONTAIN THE HEADER SPECIFICATION
915
STORE THE HEADER SPECIFICATION IN THE DATA STRUCTURE
920
CONSTRUCT A SQL QUERY TO INSERT THE DATA CONTAINED IN THE DATA-STRUCTURE AS A ROW IN 505
640
925
PASS THE SQL QUERY TO THE DATABASE ENGINE WHERE IT WHOULD BE EXECUTED
930
THE DATABASE ENGINE EXECUTES THE SQL QUERY AND INSERTS THE DATA AS A ROW IN 505
935
DONE

FIG_9

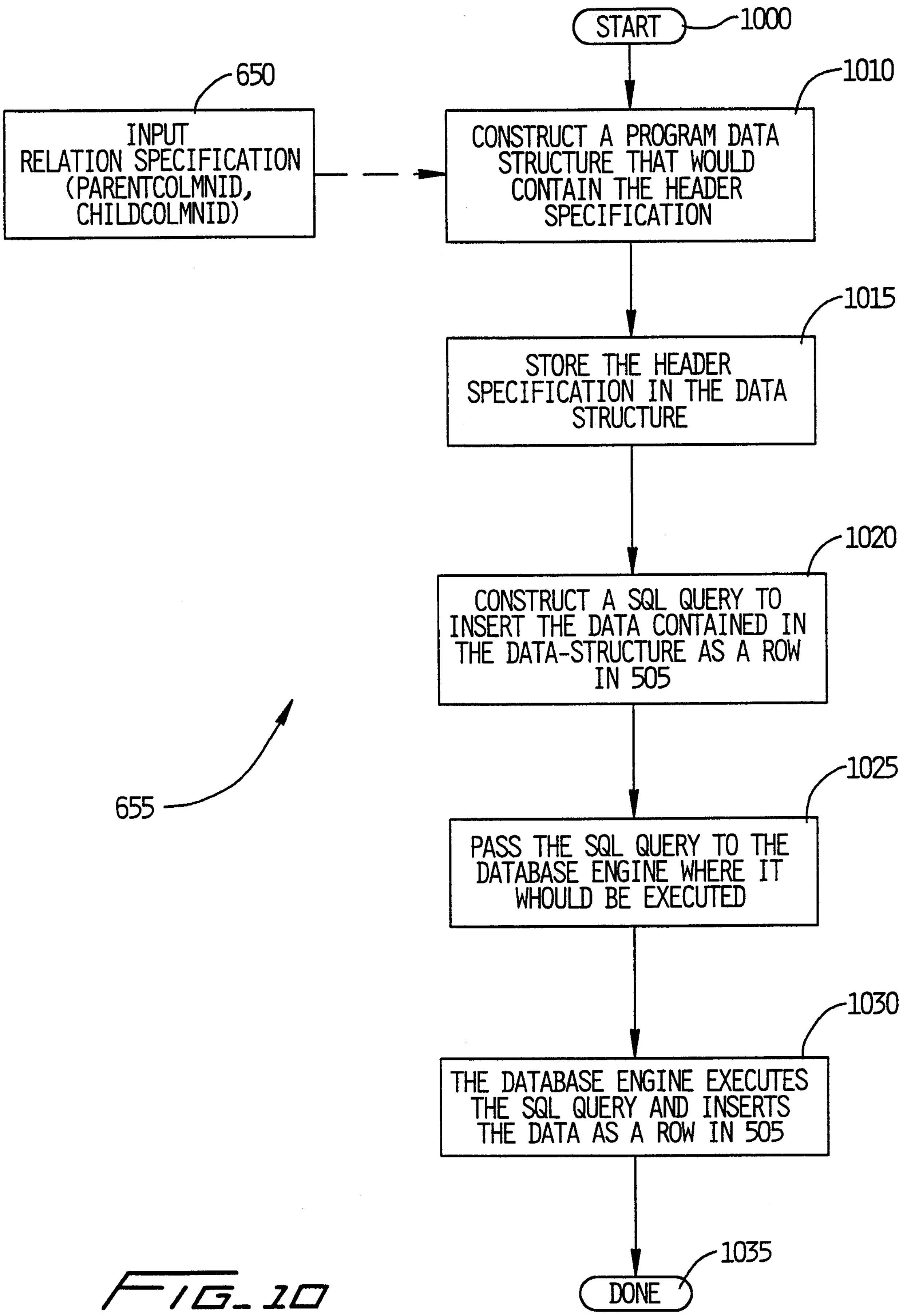

START
1000
650
INPUT RELATION SPECIFICATION (PARENTCOLMNID, CHILDCOLMNID)
1010
CONSTRUCT A PROGRAM DATA STRUCTURE THAT WOULD CONTAIN THE HEADER SPECIFICATION
1015
STORE THE HEADER SPECIFICATION IN THE DATA STRUCTURE
1020
CONSTRUCT A SQL QUERY TO INSERT THE DATA CONTAINED IN THE DATA-STRUCTURE AS A ROW IN 505
655
1025
PASS THE SQL QUERY TO THE DATABASE ENGINE WHERE IT WHOULD BE EXECUTED
1030
THE DATABASE ENGINE EXECUTES THE SQL QUERY AND INSERTS THE DATA AS A ROW IN 505
1035
DONE

FIG_10