SYSTEMS AND METHODS FOR PLATFORM-INDEPENDENT DATA FILE TRANSFERS

BACKGROUND OF THE INVENTION

[0001] This invention relates generally to systems and methods for transferring data files and, more particularly, to a system and method for performing platform-independent data file transfers within a client-server network.

[0002] During operation of a client-server network, it is often desirable to transfer data files between network nodes, such as clients and/or servers. In at least some such networks, a database, or a server having a database, may store an entire data file within a database table for a particular record. For example, a database may include a table that holds, for each record, a record identifier and a data file associated with the identified record. As the number of records grows, however, the database size increases quickly due to the additional data files. This may greatly impact the speed in which queries are processed within the database, as well as the speed of adding new records to the database.

[0003] Moreover, at least some databases may store entire data files of binary data, such as images, audio, and/or other multimedia objects, as Binary Large Objects (BLOBs). For example, a database may include a table that holds, for each record, a record identifier and a BLOB associated with the identified record. Such a database is subject to the same performance issues described above. Accordingly, a system that enables data file transfer into and out of a database, independent of the operating system transferring the data file, is needed.

BRIEF DESCRIPTION OF THE INVENTION

[0004] In one aspect, a method for transferring a data file between a client system and a server system is provided. The client executes a first computer operating system and the server executes a second computer operating system, includes a file system, and is coupled to a database. The method includes receiving,

by the database, a request to store the data file, wherein the request is transmitted to the database by a software application associated with the client. The method also includes receiving the data file from the client, storing the data file in a directory in the server file system, storing a path for the directory in the database, and outputting one of a success code and a failure code to the client.

[0005] In another aspect, a network-based system for performing platform independent data file transfers is provided. The system includes a plurality of client systems including a first client and a second client, wherein the first client executes a first computer operating system and is associated with a software application written in a first programming language, and the second client executes a second computer operating system and is associated with a software application written in a second programming language. The system also includes a server system including a file system and a database, wherein the server executes a third computer operating system. The server is configured to be coupled to the clients, and is further configured to receive, from the first client, a request to store a data file, receive the data file from the first client, store the data file in a directory in the server file system, store a path for the directory in the database, and output one of a success code and a failure code to the first client.

[0006] In another aspect, a computer coupled to a database for performing platform independent data file transfers is provided. The computer includes a file system and is in communication with a client, wherein the client executes a first computer operating system and the computer executes a second computer operating system. The computer is programmed to receive, from the client, a request to store a data file, receive the data file from the client, store the data file in a directory in the file system, store a path for the directory in the database, and output one of a success code and a failure code to the client.

[0007] In yet another aspect, a computer program embodied on a computer readable medium for performing platform independent data file transfers between a client system and a server system is provided. The server system includes

a file system and is coupled to a database. The client executes a first computer operating system and the server operates a second computer operating system. The program includes at least one code segment that receives a request to store a data file, the request transmitted by a software application associated with the client, receives the data file from the client, stores the data file in a directory in the server file system, stores a path for the directory in the database, and outputs one of a success code and a failure code to the client.

BRIEF DESCRIPTION OF THE DRAWINGS

[0008] Figure 1 is a simplified block diagram of an exemplary embodiment of an architecture of a system in accordance with one embodiment of the present invention.

[0009] Figure 2 is an expanded block diagram of an exemplary embodiment of an architecture of a system in accordance with one embodiment of the present invention

[0010] Figure 3 is a flow chart illustrating an exemplary method utilized by the system shown in Figure 2.

DETAILED DESCRIPTION OF THE INVENTION

[0011] As used herein, a processor may include any programmable system including systems and microcontrollers, reduced instruction set circuits (RISC), application specific integrated circuits (ASIC), programmable logic circuits (PLC), and any other circuit or processor capable of executing the functions described herein. The above examples are exemplary only, and thus are not intended to limit in any way the definition and/or meaning of the term processor.

[0012] As used herein, a database may include any collection of data including hierarchical databases, relational databases, flat file databases, object-relational databases, object oriented databases, and any other structured collection of records or data that is stored in a computer system. The above examples are

exemplary only, and thus are not intended to limit in any way the definition and/or meaning of the term database. Examples of databases include, but are not limited to only including, Oracle® Database, MySQL, IBM® DB2, Microsoft® SQL Server, Sybase®, and PostgreSQL. However, any database may be used that enables the systems and methods described herein. (Oracle is a registered trademark of Oracle Corporation, Redwood Shores, California; IBM is a registered trademark of International Business Machines Corporation, Armonk, New York; Microsoft is a registered trademark of Microsoft Corporation, Redmond, Washington; and Sybase is a registered trademark of Sybase, Dublin, California.)

[0013] As used herein, a Character Large Object (CLOB) is a collection of character data, such as text, that may be stored in a database system. The CLOB variable type may include a relatively large amount of data. For example, a CLOB may include up to approximately 2 gigabytes (GB) of data.

[0014] Described in detail herein are exemplary embodiments of systems and methods that facilitate performing platform-independent data file transfers within a client-server network using a database. The systems and methods facilitate, for example, storing a data file in a file system of a server that is coupled to the database, and storing a path of the data file in the database. Moreover, the systems and methods facilitate reading the data file from the file system by initially querying the database for the file system path of the data file, and then reading the data file from the file system according to the returned path. As a result, the systems and methods described herein enable a software application written in any programming language to write data files to, and read data files from, a file system of a server that is coupled to a database, regardless of the operating system of the client and regardless of the operating system of the server.

[0015] Technical effects of the systems and methods described herein include at least one of (a) creating a plurality of directories in a file system of a server coupled to a database, wherein the server uses, or executes, any operating system; (b) receiving a data file from a client, wherein the client uses any operating

system and transmits the data file using a client software application that is written in any programming language; (c) storing the data file in a memory of the server; (d) declaring a CLOB variable; (e) sequentially reading each line of the data file from the server memory into the CLOB variable; (f) copying each line from the CLOB variable to a directory in the server file system; and (g) storing a path for the directory in the database. Additional technical effects of the systems and methods described herein include at least one of (g) receiving a request from the client to retrieve a stored data file; (h) declaring a CLOB variable; (i) sequentially reading each line of the data file from the directory into the CLOB variable; (j) copying each line from the CLOB variable to the server memory; and (k) transmitting the data file to the client.

[0016] In one embodiment, a computer program is provided. The computer program is embodied on a computer readable medium and utilizes a Structured Query Language (SQL) for communications between an end user device, such as a client system using, or executing, any operating system, and a server system using any operating system and coupled to communicate with a database. In one embodiment, such a system is used within a business-entity intranet, such as a payment transaction network. In an alternative embodiment, such a system is used within a combination of one or more business-entity intranets as well as on the Internet.

[0017] The systems and methods described herein are not limited to the specific embodiments described herein. For example, components of each system and/or steps of each method may be used and/or practiced independent and separately from other components and/or steps described herein. In addition, each component and/or step may also be used and/or practiced with other assembly packages and methods.

[0018] Figure 1 is a simplified block diagram of an exemplary system 100 in accordance with one embodiment of the present invention. In one embodiment, system 100 is a financial transaction card payment system. More specifically, in the exemplary embodiment, system 100 includes a server system 102,

and a plurality of client sub-systems, also referred to as client systems 104, connected to server system 102. In one embodiment, client systems 104 are computers including a web browser and/or a client software application, such that server system 102 is accessible to client systems 104 over a network, such as the Internet and/or an intranet. Client systems 104 are interconnected to the Internet through many interfaces including a network, such as a local area network (LAN), a wide area network (WAN), dial-in-connections, cable modems, wireless modems, and/or special high-speed ISDN lines. Client systems 104 could be any device capable of interconnecting to the Internet including a web-based phone, personal digital assistant (PDA), or other web-based connectable equipment. A database server 106 is connected to a database 108 containing information on a variety of matters, such as financial transaction card payment data. In one embodiment, centralized database 108 is stored on server system 102 and is accessed by potential users at one of client systems 104 by logging onto server system 102 through one of client systems 104. In an alternative embodiment, database 108 is stored remotely from server system 102 and may be non-centralized.

[0019] Figure 2 is an expanded block diagram of an exemplary embodiment of a server architecture of a system 200 in accordance with one embodiment of the present invention. Components in system 200, identical to components of system 100 (shown in Figure 1), are identified in Figure 2 using the same reference numerals as used in Figure 1. System 200 includes server system 102 and client systems 104. Server system 102 further includes database server 106, an application server 202, a web server 204, a fax server 206, a directory server 208, and a mail server 210. A disk storage unit 212 is coupled to database server 106 and directory server 208. Examples of disk storage unit 212 include, but are not limited to including, a Network Attached Storage (NAS) device and a Storage Area Network (SAN) device. Database server 106 is also coupled to database 108. Servers 106, 202, 204, 206, 208, and 210 are coupled in a local area network (LAN) 214. In addition, a system administrator's workstation 216, a user workstation 218, and a supervisor's workstation 220 are coupled to LAN 214. Alternatively, workstations

216, 218, and 220 are coupled to LAN 214 using an Internet link or are connected through an intranet.

[0020] Each of client systems 104 and workstations 216, 218, and 220 is a personal computer having a web browser and/or a client application. Although the functions performed at client systems 104 and workstations 216, 218, and 220 are typically illustrated as being performed at respective client systems 104 and/or workstations 216, 218, and 220, such functions may instead be performed at one of many personal computers coupled to LAN 214. Client systems 104 and workstations 216, 218, and 220 are illustrated as being associated with separate functions only to facilitate an understanding of the different types of functions that can be performed by individuals having access to LAN 214.

[0021] Server system 102 is configured to be communicatively coupled to client systems 104 and workstations 216, 218, and 220 to enable server system 102 to be accessed using an ISP Internet connection 222. The communication in the exemplary embodiment is illustrated as being performed using the Internet, however, any other wide area network (WAN) type communication can be utilized in other embodiments, i.e., the systems and processes are not limited to being practiced using the Internet. In addition, and rather than WAN 224, local area network 214 could be used in place of WAN 224. Furthermore, fax server 206 communicates with remotely located client systems 104 and workstations 216, 218, and 220 using a telephone link.

[0022] In the exemplary embodiment, client systems 104 may use, or execute, any of a plurality of computer operating systems. Examples of possible computer operating systems include, but are not limited to only including, Microsoft® Windows, Mac OS X®, and a Linux distribution, such as Canonical Ltd. Ubuntu®, Novell® SUSE Linux, and Debian®. (Microsoft is a registered trademark of Microsoft Corporation, Redmond, Washington; Mac OS X is a registered trademark of Apple, Inc., Cupertino, California; Canonical Ltd. Ubuntu is a registered trademark of Canonical Ltd., Isle of Mann; Novell is a registered trademark of Novell, Inc.,

Waltham, Massachusetts; and Debian is a registered trademark of Software in the Public Interest, Inc., Indianapolis, Indiana.) However, any computer operating system may be used that enables client systems 104 to function as described herein. Each client system 104 interacts with server system 102 and/or accesses database 108 through database server 106 using a client application. In the exemplary embodiment, the client software application may be written in any of a plurality of programming languages. Examples of possible programming languages include, but are not limited to only including, C#, C++, Java, and/or Perl. However, any programming language may be used that enables the software application used by client systems 104 to function as described herein.

[0023] Moreover, in the exemplary embodiment, database server 106 may use, or execute, any of a plurality of computer operating systems. Examples of possible computer operating systems include, but are not limited to only including, Microsoft® Windows Server 2003, FreeBSD®, UNIX®, and a Linux distribution, such as those discussed above. (Microsoft is a registered trademark of Microsoft Corporation, Redmond, Washington; FreeBSD is a registered trademark of The FreeBSD Foundation, Boulder, Colorado; and UNIX is a registered trademark of The Open Group, San Francisco, California.) However, any computer operating system may be used that enables database server 106 to function as described herein. Further, any server communicatively coupled to database 108, such as directory server 208, for example, may use, or run, any of the plurality of computer operating systems discussed above.

[0024] Figure 3 is a flowchart illustrating an exemplary method 300 utilized by system 200 (shown in Figure 2). In the exemplary embodiment, in order to transfer data files within system 200, a plurality of directories must be created 302. Each directory is created in a file system of a server coupled to database 108 (shown in Figure 2), such as database server 106 (shown in Figure 2). In an alternative embodiment, directories may be created on any server within server system 102 (shown in Figure 2) having a file system accessible by database 108. A plurality of directory types and directory names are also created within database 108. In one

embodiment, SQL commands are used to instruct database 108 to create the directory types and to associate directory names with the directory types.

[0025] After directories have been created within a file system and associated with directory types in database 108, client system 104 (shown in Figure 2) may request, through the client application, that a data file be stored or read. Database server 106 receives 304 the request and determines whether the request is a store request or a read request.

[0026] In the exemplary embodiment, and when the request is a store request, database server 106 receives 306 the data file and a directory from client system 104. In an alternative embodiment, and when the directory is not received from client system 104, database server 106 selects a default directory. Database 108 then stores 308 the data file into a memory (not shown) of a server, such as database server 106 or directory server 208 (shown in Figure 2). The server memory may include Random Access Memory (RAM) modules and/or a swap space that has been created in the server file system. Database 108 declares 310 a Character Large Object (CLOB) variable, which is used to copy data from the data file stored in the server memory to the destination directory. Database 108 then reads 312 a first line of characters from the server memory. More specifically, database 108 sets the CLOB variable equal to the first line of characters. In the exemplary embodiment, database 108 then opens the directory, which was created as described above, and stores 314 the CLOB variable value into the directory. Alternatively, if the directory has not already been created, database 108 creates 302 the directory as described above. Database 108 then determines 316 whether there are remaining lines of characters in the data file stored in the server memory. The end of the data file may be represented by a flag, such as an End Of File (EOF) flag. If there are lines of characters remaining in the data file, database 108 repeats the steps of reading 312 the next line of characters from the server memory, setting the CLOB variable equal to the next line of characters, and storing 314 the CLOB variable value into the directory. Database 108 then stores 318 a path of the directory and/or the file name in a particular database table. The database table includes multiple columns of data

and/or identifiers, such as a first column that includes an identifier for each database record and a second column that includes the path of the directory to which the data file has been stored. When the data file has been stored, database 108 outputs 320 to client system 104 one of a success code and a failure code. The success code or failure code may be displayed to a user via an audible signal and/or a visual signal on a client display.

[0027] In the exemplary embodiment, and when the request is a read request, database server 106 receives 322 the file name and/or an associated database record from client system 104. Database 108 declares 324 a Character Large Object (CLOB) variable, and searches 326 for the directory of the server file system associated with the data file. Database 108 may search a table, such as a table described above for storing the directory paths and associated record identifiers, using an identifier for the database record associated with the data file being requested. Database 108 opens the data file within the directory and reads 328 the first line of the data file. Database 108 sets the CLOB variable to equal the first line of the data file. The CLOB variable value is then stored 330 in the server memory. Database 108 then determines 332 whether there are remaining lines of characters in the data file stored in the directory. The end of the data file may be represented by a flag, such as an End Of File (EOF) flag. If there are lines of characters remaining in the directory, database 108 repeats the steps of reading 328 the next line of characters from the directory, setting the CLOB variable equal to the next line of characters, and storing 330 the CLOB variable value into the server memory. The data file is then transferred 334 to client system 104. When the data file has been transferred, database 108 outputs 320 to client system 104 one of a success code and a failure code. The success code or failure code may be displayed to a user via an audible signal and/or a visual signal on a client display. Alternatively, the data file may be displayed to the user of client system 104 using the client display.

[0028] The systems and methods described herein facilitate performing platform-independent data file transfers in a client-server network that includes a database. Moreover, the systems and methods enable a software

application written in any programming language, and run by a client using any computer operating system, to write data files to and read data files from a file system of a server that is coupled to a database, regardless of the computer operating system used by the server.

[0029] Although the systems and methods described herein are described in the context of platform-independent data file transfers using a database, it is understood that the systems and methods are not limited to only such systems and/or methods. Likewise, the system components illustrated are not limited to the specific embodiments herein, but rather, components of the system may be utilized independently and separately from other components described herein.

[0030] While the invention has been described in terms of various specific embodiments, those skilled in the art will recognize that the invention can be practiced with modification within the spirit and scope of the claims.

WHAT IS CLAIMED IS:

1. A method for transferring a data file between a client system and a server system that includes a file system and is coupled to a database, wherein the client executes a first computer operating system and the server executes a second computer operating system, said method comprising:

receiving, by the database, a request to store the data file, the request transmitted by a software application associated with the client;

receiving the data file from the client;

storing the data file in a directory in the server file system;

storing a path for the directory in the database; and

outputting one of a success code and a failure code to the client.

2. A method in accordance with Claim 1 further comprising creating a plurality of directories in the server file system, wherein the directory in which the data file is stored is one of a selected directory and a default directory.

3. A method in accordance with Claim 1 wherein storing the data file in a directory in the server file system comprises storing the received data file in a memory of the server.

4. A method in accordance with Claim 3 wherein storing the data file in a directory of the server file system further comprises declaring, by the database, a Character Large Object (CLOB) variable.

5. A method in accordance with Claim 4 wherein the data file includes a plurality of character lines, storing the data file in a directory of the server file system further comprises sequentially storing each line of the plurality of character lines into the directory by:

setting the CLOB variable equal to each line;

copying the CLOB variable contents into the directory; and

determining whether the data file contains additional lines.

6. A method in accordance with Claim 1 further comprising:

receiving, by the database, a request to transmit the data file to the client;

storing the data file in a memory of the server; and

transmitting the data file to the client.

7. A method in accordance with Claim 6 wherein the data file includes a plurality of character lines, storing the data file in a memory of the server comprises sequentially storing each line of the plurality of character lines into the memory by:

accessing the data file stored in the directory;

declaring a Character Large Object (CLOB) variable

setting the CLOB variable equal to each line;

copying the CLOB variable contents into the memory; and

determining whether the data file contains additional lines.

8. A method in accordance with Claim 1 wherein storing the data file in a directory of the server file system comprises storing the data file in the server file system, wherein the second computer operating system executed by the server is different than the first computer operating system executed by the client.

9. A method in accordance with Claim 6 wherein the client is a first client of a plurality of client systems, transmitting the data file to the client comprises transmitting the data file to a second client of the plurality of client systems, wherein the second client executes a third computer operating system that is different than the first computer operating system executed by the first client.

10. A method in accordance with Claim 9 wherein transmitting the data file to a second client comprises transmitting the data file to the second client, wherein the second client is associated with a software application that is written in a different programming language than the language of the software application associated with the first client.

11. A network-based system for performing platform independent data file transfers, said system comprising:

a plurality of client systems comprising a first client and a second client, wherein said first client executes a first computer operating system and is associated with a software application written in a first programming language, said second client executes a second computer operating system and is associated with a software application written in a second programming language; and

a server system comprising a file system and a database, said server executes a third computer operating system, said server configured to be coupled to said plurality of clients, said server further configured to:

receive, from said first client, a request to store a data file;

receive the data file from said first client;

store the data file in a directory in said server file system;

store a path for the directory in said database; and

output one of a success code and a failure code to said first client.

12. A system in accordance with Claim 11 wherein said server is further configured to create a plurality of directories in said server file system, wherein the directory in which the data file is stored is one of a selected directory and a default directory.

13. A system in accordance with Claim 11 wherein said server is further configured to:

declare a Character Large Object (CLOB) variable in said database;

store the received data file in a memory of said server; and

store the data file in the directory in said server file system using the CLOB variable.

14. A system in accordance with Claim 11 wherein the data file includes a plurality of character lines, said server is further configured to:

set the CLOB variable equal to each line;

copy the CLOB variable contents into the directory; and

determine whether the data file contains additional lines.

15. A system in accordance with Claim 11 wherein said server is further configured to:

receive a request to transmit the data file to a client of said plurality of clients;

access the data file stored in the directory;

declare a Character Large Object (CLOB) variable in said database;

set the CLOB variable equal to each line;

copy the CLOB variable contents into said server memory;

determine whether the data file contains additional lines; and

transmit the data file to said client from said server memory.

16. A system in accordance with Claim 11 wherein the first computer operating system executed by said first client is different than at least one of the computer operating system executed by said second client and the third operating system executed by said server.

17. A system in accordance with Claim 11 wherein the first programming language used to write the software application associated with said first client is different than the second programming language used to write the software application associated with said second client.

18. A computer coupled to a database for performing platform independent data file transfers, said computer comprising a file system, said computer in communication with a client, wherein the client executes a first computer operating system and said computer executes a second computer operating system, said computer programmed to:

receive, from the client, a request to store a data file;

receive the data file from the client;

store the data file in a directory in the file system;

store a path for the directory in the database; and

output one of a success code and a failure code to the client.

19. A computer in accordance with Claim 18 wherein said computer is further programmed to:

store the data file in a memory of said computer;

declare, in the database, a Character Large Object (CLOB) variable;

set the CLOB variable equal to a first line of characters in the data file stored in said memory;

copy the CLOB variable contents into the directory; and

determine whether the data file contains additional lines.

20. A computer in accordance with Claim 18 wherein said computer is further programmed to:

receive a request to transmit the data file to the client;

access the data file stored in the directory;

declare a Character Large Object (CLOB) variable in the database;

set the CLOB variable equal to each line;

copy the CLOB variable contents into a memory of the computer;

determine whether the data file contains additional lines; and

transmit the data file to the client from the memory.

21. A computer in accordance with Claim 20 wherein the client is a first client of a plurality of client systems, said computer is configured to transmit the data file to a second client of the plurality of client systems, wherein the first client executes the first computer operating system that is different from the second computer operating system executed by said computer, and wherein the second client executes a third operating system.

22. A computer program embodied on a computer readable medium for performing platform independent data file transfers between a client system and a server system that includes a file system and is coupled to a database, wherein the client executes a first computer operating system and the server operates a second computer operating system, said program comprising at least one code segment that:

receives a request to store a data file, the request transmitted by a software application associated with the client;

receives the data file from the client;

stores the data file in a directory in the server file system;

stores a path for the directory in the database; and

outputs one of a success code and a failure code to the client.

23. A computer program in accordance with Claim 22 further comprising at least one code segment that:

stores the data file in a memory of the server;

declares, in the database, a Character Large Object (CLOB) variable;

sets the CLOB variable equal to a first line of characters in the data file stored in the server memory;

copies the CLOB variable contents into the directory; and

determines whether the data file contains additional lines.

24. A computer program in accordance with Claim 22 further comprising at least one code segment that:

receives a request to transmit the data file to the client;

accesses the data file stored in the directory;

declares a Character Large Object (CLOB) variable in the database;

sets the CLOB variable equal to each line;

copies the CLOB variable contents into a memory of the server;

determines whether the data file contains additional lines; and

transmits the data file to the client from the server memory.

25. A computer program in accordance with Claim 22 wherein the client is a first client of a plurality of client systems, said program further comprising at least one code segment that transmits the data file to a second client of the plurality of client systems, wherein the first client executes the first computer operating system that is different from the second computer operating system executed by the server, and wherein the second client executes a third operating system