LDAP TO SQL DATABASE PROXY SYSTEM AND METHOD

[0001] The present matter relates generally to computer system integration and particularly to facilitating database activity between otherwise incompatible computer system components.

[0002] Configuring a computer system to perform specific functions (for example, electronic commerce, web services, business transaction processing, electronic data communications, etc.) often includes the integration of disparate hardware and software from various sources. A particular component for such a system may be configured for operation with one or more other components in accordance with a first protocol. However, a systems integrator may wish to have the particular component operate with a specific other component that does not support that first protocol.

[0003] For example, it is often desired to integrate a data store component comprising a directory or other database with another component of a system. Data stores are typically defined in accordance with specific schemas and protocols for accessing information in the data store. One such protocol is the Structured Query Language (SQL) for accessing relational database management system (RDBMS) databases and another is Lightweight Directory Access Protocol (LDAP) for providing access to directory services, specifically, X.500-based directory services optimized for reading, searching and browsing information. Each has particular advantages or other characteristics which may make its use desirable in a computer system. For example, LDAP is adapted as a network protocol for accessing information in a directory and can be easy to implement in a client component for accessing information stored remotely. It is often used for email clients to access directory information (e.g. an address book). SQL-based databases are available widely from different vendors and provide complex functionality including transactional support and roll-back schemes not typically found with LDAP directories.

[0004] Sometimes a data store in a computer system is to be shared by multiple components that are configured for different protocols. Thus there are times when a component and a data store are incompatible, for example, with the data store configured for SQL operation and the component configured for LDAP operation.

[0005] One implementation of LDAP is OpenLDAP, an open source community developed software data store coordinated by the OpenLDAP Foundation. One component of this software is a LDAP server for providing LDAP access to data stored in one or more backends. OpenLDAP provides access to various backends or storage types

(i.e. types of databases) through specific modules. OpenLDAP is configured to provide backends of three general categories, namely those that 1) store data, 2) proxy otherwise stored data or 3) generate data on the fly. One proxy backend is back-sql for mapping an SQL-based RDBMS to a directory service agent (DSA). Examples of supported RDBMSes are DB2 from IBM, MySQL from MySQL AB and PostgreSQL from PostgreSQL Global Development Group.

[0006] However, as back-sql proxies an SQL-database for multiple purposes and the schema is predefined by the needs of OpenLDAP, it performs more work than may be necessary and incurs more overhead (storage and other resources) than may be desired such that its performance characteristics are not suitable. Back-sql is configured to use its own proprietary database schema rather than an external database schema even if an existing SQL database is defined in accordance with its respective database schema.

[0007] Accordingly, there is a need for a simple LDAP to SQL database proxy.

[0007a] IBM Systems Journal, Vol 39, no2, "An enterprise directory solution with DB2", by Shi et al, discloses a system by which an SQL database is defined, through the database schema, by the LDAP server into which it is plugged. Similarly, US2003/0191757 discloses a system which maps LDAP requests into DA database responses and communicates the DA response into an LDAP compatible response.

[0008] There is preferably provided an LDAP to SQL proxy to adapt a relational database as a directory service for one or more LDAP clients typically served by an LDAP server. The proxy comprises LDAP to SQL query and response mappings preferably but not necessarily for a reduced set of LDAP operations. The proxy may comprise a mechanism to maintain a plurality of persistent database connections to reduce query processing overhead. The proxy may be further adapted to maintain a plurality of persistent connections to each of at least two alternative databases for executing queries to provide enhanced failsafe operation. A plurality of LDAP to SQL proxied databases may be configured for sharing query loads to enhance scalability and performance. The proxy may be configured as a module for providing a backend relational database to an LDAP server. Advantageously, the database may be defined in accordance with a database schema that is external to the LDAP server to facilitate integration of an existing database to an LDAP environment and avoid duplication.

[0009] In accordance with a first aspect, there is preferably provided a method of providing directory services in accordance with an lightweight data access protocol (LDAP) to an LDAP client comprising: providing an LDAP server for receiving and responding to directory service requests in accordance with LDAP, the LDAP server comprising an LDAP to SQL proxy module defined to map LDAP service requests to SQL queries, execute the SQL queries against a database to obtain SQL query results and map the SQL query results to LDAP responses; and wherein the database is defined in accordance with a database schema external to the LDAP server. The proxy module may be configured to map a reduced set of types of LDAP service requests to minimize LDAP server resource consumption, for example, only those sufficient to facilitate information retrieval from the database. Preferably, the LDAP server is configured as an OpenLDAP server. The LDAP to SQL proxy module may be defined to maintain a pool of persistent connections to at least one database such that the connections provide access to execute the SQL queries.

[0010] In accordance with another aspect there is preferably provided a computer readable medium comprising instructions, which, when executed by a computing device, adapt an LDAP server for providing directory services to connect a database configured for SQL operation to a client configured for LDAP operation. The instructions comprising mappings to transform LDAP service requests to SQL queries and SQL responses to LDAP responses and the database is defined in accordance with a database schema which is external to the LDAP server.

[0011] In yet a further aspect there is preferably a computer system comprising: an LDAP server for providing directory services to an LDAP client; an LDAP to SQL proxy adapting the LDAP server to connect a database accessible by SQL to the LDAP client, the proxy transforming LDAP service requests to SQL queries and SQL responses to LDAP responses; and a database accessible using SQL, said database defined in accordance with a database schema external to the LDAP server.

[0012] In one embodiment, the computer system preferably comprises a like plurality of each of the LDAP servers, LDAP to SQL proxies and databases, one of each server, proxy and database associated to define a cell; and said computer system further comprising a

master database accessible by each cell. The computer system may further comprise a redirector server to share directory service requests among the cells. Each proxy in a particular cell is preferably configured to execute directory service requests against the master database if the database of the particular cell is unavailable.

BRIEF DESCRIPTION OF THE DRAWINGS

[0013] An embodiment or embodiments will now be described by way of example only with reference to the following drawings in which:

Figure 1 is a block diagram of an LDAP server with an LDAP to SQL proxy module in accordance with an embodiment;

Figure 2 is a block diagram of message processing in accordance with an embodiment;

Figure 3 is a block diagram of an computer system architecture providing a plurality of SQL databases to an LDAP client via LDAP to SQL proxies in accordance with an embodiment; and

Figure 4 is a flow chart of operations in accordance with the embodiment of Figure 3.

[0014] For convenience, like numerals in the description refer to like structures in the drawings.

DESCRIPTION OF PREFERRED EMBODIMENT

[0015] Figure 1 illustrates a representative computer system 100 in which a first component 102 comprises a LDAP client for accessing directory services provided by a LDAP server 104 over a communication coupling 106. However, the backend data store (second component) is an RDBMS 118 accessible via SQL. In the present embodiment, the LDAP server is configured as an OpenLDAP server (*slapd* daemon) for a LINUX operating system environment having a plurality of standard modules for various backends such as but not limited to bdb 108, sql 112 and monitor 114. LDAP server 104 is adapted with a backend proxy module 110 for communicating SQL queries via communication coupling 116 to database 118, which in this embodiment is an exemplary PostgreSQL database defined in accordance with a database schema maintained externally of the LDAP server 104. Couplings 106 and 116 may be public or private networks, local or

wide area networks etc. or other communication channels. Other SQL-based databases or LDAP servers may be used rather than those shown and described. Some OpenLDAP modules need not be provisioned if not needed, for example, back-sql, back-bdb, etc.

[0016] Requests from LDAP client 102 for directory services (typically queries for obtaining information stored to the backend 118) are communicated in accordance with an LDAP protocol to LDAP server 104. Server 104 invokes proxy 110 which transforms the LDAP queries to SQL for communicating to database 118. Responses from database 118 are received in accordance with the SQL protocol and transformed by proxy module 110 to LDAP protocol responses for sending back to LDAP client 102 via server 104.

[0017] Figure 2 illustrates an example of "message" operations between server 104, proxy module 110 and database 118. It is understood that "message" herein means information communicated or otherwise passed from one component to another and need not be restricted to data communicated in a defined message form via a communication network. Message 1 comprises a supply of credentials for server 104 and client 102. Proxy module 110 transforms the message to provide the client credentials to the database 118 (message 2) and authenticates the server credentials locally, providing a result (message 3). Upon receipt of the SQL result of the authentication request (message 4), proxy module 110 provides a transformed LDAP result (message 5) to server 104. Message 5 is thus dependent on the results of the two authentications. Message 6 comprises an LDAP query which is transformed and sent as a SQL query (message 7). The SQL response (message 8) is received (e.g. a set of records from the external database) and transformed to an LDAP response (message 9).

[0018] Proxy module 110 may be configured with knowledge of the requirements of the LDAP client and the specific directory services to be provided to it. As a result, a specific set of LDAP to SQL and vice versa transformations may be determined and implemented to provide reduced LDAP directory services and operations. Proxy module 110 thus maps incoming message formats to outgoing formats efficiently without necessarily implementing maps for all types of LDAP service requests and SQL queries and responses. For example, if LDAP client 102 is limited to retrieving (i.e. searching and reading) information from a directory, LDAP to SQL proxy module 110 need not be configured to facilitate LDAP requests to modify or store information. As such, the proxy

module will consume fewer resources than a fuller SQL proxy implementation such as back-sql. Additionally and importantly, the proxy permits use or integration of an existing database particularly when it is desirable not to change the existing database (i.e. define a new schema). Thus an SQL database with its existing schema may be efficiently integrated with components configured for using LDAP. Duplicate tables etc. need not be created as is the result of using a second schema for the same data.

[0019] In a preferred implementation, proxy module 110 (back-prox) is configured in C programming language for OpenLDAP as set forth in Table 1:

File Name	Purpose
back-prox.h	Defines structures and constants that are required by all of the files in the module. The structures are used to pass information between functions.
connectionpool.h	This is a header file for the connection pool.
connectionpool.c	This file contains the functions for initializing, maintaining and cleaning up the connection pool.
external.h	This header defines the functions for the external interface to the rest of OpenLDAP. This file is required by each module.
init.h	Header file used to initialize the module.
init.c	
search.h	Header file for the searching functions. Also defines structs for converting LDAP attribute names to SQL column names.
search.c	When a search is performed the functions defined in this file are used to translate the incoming query, send it to a Postgres backend, and then package the result into an LDAP response.
bind.c	This file contains the functions that are used when a user attempts to bind to a node in OpenLDAP. This is where the authentication against both the server and user credentials takes place.

Table 1

[0020] To reduce transactional overhead for handling queries, a pool of persistent connections is maintained for reusably connecting to the external database. The connections may be shared in a round robin or other sharing fashion as is known to those of ordinary skill in the art. Further operations are described herein below with reference to Figure 4 wherein a preferred proxy maintains a connection pool for each of two alternative databases (a local copy and a remote master) to query.

[0021] The following functions map the function pointers from OpenLDAP 102 to the actual implementation in the proxy module 110 (i.e. back-prox in the preferred

embodiment). Functions that are set to 0 have no implementation and the 0 is the default return value to indicate success.

```
bi->bi_open = 0;
bi->bi_config = 0;
bi->bi_close = 0;
bi->bi_destroy = 0;
bi->bi_db_init = backprox_db_init;
bi->bi_db_config = backprox_db_config;
bi->bi_db_open = backprox_db_open;
bi->bi_db_close = backprox_db_close;
bi->bi_db_destroy = backprox_db_destroy;
bi->bi_op_abandon = 0;
bi->bi_op_compare = 0;
bi->bi_op_bind = backprox_bind;
bi->bi_op_unbind = backprox_unbind;
bi->bi_op_search = backprox_search;
bi->bi_op_modify = 0;
bi->bi_op_modrdn = 0;
bi->bi_op_add = 0;
bi->bi_op_delete = 0;
bi->bi_chk_referrals = 0;
bi->bi_operational = 0;
bi->bi_connection_init = 0;
bi->bi_connection_destroy = 0;
```

[0022] Functions that do not have an implementation were either not necessary or were redundant and not used because the implementation was done in a different function. The corresponding functions can be found throughout the rest of the module. The BackendInfo structure that is being referred to by the variable bi is defined in slap.h and is used to pass information, function pointers and semaphores between the various functions in the module. A brief description of the implementations of the functions in the modules follows in Table 2. Each of the *_db_* descriptions comes from the slap.h file.

Function	Description
bi_db_init	Called to initialize each database, called upon reading 'database <type>' called only from backend_db_init()
bi_db_config	Called to configure each database, called per database to handle per database options called only from read_config()
bi_db_open	Called to open each database called once per database immediately AFTER bi_open() calls but before daemon startup. Called only by backend_startup()

bi_db_close	Called to close each database called once per database during shutdown but BEFORE any bi_close call. Called only by backend_shutdown()
bi_db_destroy	Called to destroy each database called once per database during shutdown AFTER all bi_close calls but before bi_destory calls. Called only by backend_destory()
bi_op_bind	Called on a bind operation.
bi_op_unbind	Called on an unbind operation.
bi_op_search	Called on a search operation.

Table 2

[0023] The standard configuration file for OpenLDAP is called the slapd.conf file and can often be found at /etc/openldap on a Linux machine for a standard installation. The back-prox module comprises additional parameters that are read on startup from the backprox_db_config() function in the init.c file. Table 3 lists the new parameters and their roles.

Parameter	**Description**
cell_dbuser	The login name of the cell database user.
cell_dbpasswd	The password of the cell database user.
cell_dbname	The name of the database instance to log into.
cell_dbhost	The host machine of the cell database.
primary_dbuser	The login name of the primary database user.
primary_dbpasswd	The password of the primary database user.
primary_dbname	The name of the database instance to log into.
primary_dbhost	The host machine of the primary database.
connection_pool_size	The number of persistent connections in each of the connection pools. There is one pool for the local copy and one for the master database.
reconnect_interval	When a database goes down, this is the polling frequency that the reconnection thread will use.

Table 3

[0024] Mapping of LDAP attributes to SQL and vice versa may be performed in a variety of ways including table look-ups etc. as will be understood to persons of ordinary skill in the art.

[0025] Figure 3 illustrates a block diagram of a computer system 300 for providing scalable and redundant directory services to an LDAP client. In system 300, there is a master database 324 which is replicated to a plurality of cells 306, 308 and 310 each cell comprising a cell database (318, 320 and 322) fronted by a respective LDAP to SQL

proxy component (312, 314 and 316) for providing the directory services to the LDAP client 302. A query from the LDAP client 302 is directed to a particular one of the n cells 306, 308 and 310 by a redirector component 304 (typically a server configured for load sharing) to share load among the cells and provide for scalable and redundant service. LDAP to SQL proxy component 312, 314 and 316 may comprise an OpenLDAP server with a LDAP to SQL proxy module as previously described and further described below.

[0026] A request for directory services to a particular cell may be serviced from that cell's database, a local copy of the master database, or from the master database should the local copy database not be available. Figure 4 illustrates a flow chart of operations 400 for a connection pool of a cell (e.g. of the LDAP to SQL proxy) to determine which of the cell database and master database to direct the transformed LDAP to SQL query. In order to decrease processing, a flag may be set to indicate to the proxy not to attempt to use any of its connections in the cell connection pool. A background thread may be executed to re-establish connections to the cell database when it becomes available, periodically retrying as necessary until successful. The thread can rebuild the connections and reset the flag before terminating.

[0027] Operations 400 commence following receipt of a query to service at start step 402. The cell down flag is checked at step 404. If it is not down, via No branch to step 406, the cell connection is tested and a result determined (step 408). If the cell connection is OK, via Yes branch to step 410, the transformed LDAP to SQL query is executed against the cell's local database. The connection is returned to the pool (step 411) and operations for that query end (step 412). If the connection did not test OK at step 408 or the cell flag was set at step 404, via respective branches to step 414, cell reconnection is spawned and operations attempt to query the master database.

[0028] Steps 416-423 are performed relative to the master database and are similar to steps 404-410 as performed relative to the cell database. If a query cannot be completed against either the cell database or the master database, at step 424 an error is returned and operations finish (step 412).

[0029] In a preferred implementation, system 300 may comprise a portion of an electronic mail system. LDAP client 302 may comprise an electronic mail server for providing electronic mail via web access to users of the mail service such as is available from Mirapoint, Inc. Such a server may be integrated with other user provisioning services etc. and require access to external databases storing such user information etc.

[0029] Although an embodiment or embodiments, sometimes preferred, have been described herein, it will be understood by those skilled in the art that variations may be made thereto – the scope of the invention is defined by the appended claims.

Claims

1. A method of providing directory services in accordance with a lightweight data access protocol (LDAP) to an LDAP client comprising:
providing at least one LDAP server for receiving and responding to directory service requests in accordance with LDAP, each LDAP server comprising an LDAP to SQL proxy module defined to map LDAP service requests to SQL queries, execute the SQL queries against a database to obtain SQL query results and map the SQL query results to LDAP responses, wherein a database schema, which is maintained externally to and separate from the LDAP server, is operable to integrate the database of SQL queries and results into the LDAP server.

2. The method according to claim 1 wherein each said proxy module is configured to only map a reduced set of types of LDAP service requests.

3. The method according to claim 2 wherein the LDAP to SQL proxy module maps LDAP service requests sufficient to facilitate only information retrieval from the database.

4. The method according to any one of claims 1 to 3 wherein the LDAP server is configured as an OpenLDAP server.

5. The method of any one of claims 1 to 4 wherein the each group of LDAP server, proxy and database defines a cell, the method further comprising the provision of a master database accessible to each cell.

6. The method according to any one of claims 1 to 5 wherein the LDAP to SQL proxy module maintains a pool of persistent connections to at least one database, said connections providing access to execute said SQL queries.

7. The method of any one of claims 1 to 6 further comprising the provision of a redirector server for sharing directory service requests among the cells.

8. The method of any one of claims 1 to 7 wherein each proxy in a particular cell is configured to execute directory service requests against the master database if the database of the particular cell is unavailable.

9. A computer readable medium comprising instructions, which, when executed by a computing device, adapt an LDAP server for providing directory services to connect a database configured for SQL operation to a client configured for LDAP operation to implement the method of any one of claims 1 to 8.

10. A computer system comprising:
at least one LDAP server for providing directory services to an LDAP client; each of the at least one servers comprising
an LDAP to SQL proxy for adapting the LDAP server to connect a database accessible by SQL to the LDAP client, the proxy being arranged to map LDAP service requests to SQL queries and SQL responses to LDAP responses; and
a database accessible using SQL, wherein a database schema, which is maintained externally to and separate from the LDAP server, is operable to integrate the database of SQL queries and responses into the LDAP server.

11. The computer system according to claim 10 wherein the proxy is arranged to transform a reduced set of types of LDAP service requests.

12. The system according to claims 10 or 11 wherein the LDAP server is an OpenLDAP server.

13. The computer system according to any one of the previous claims wherein each group of LDAP server, LDAP to SQL proxy and database defines a cell and said computer system further comprising a master database accessible by each cell.

14. The computer system according to claim 13 further comprising a redirector server for sharing directory service requests among the cells.

15. The computer system according to claim 13 or 14 wherein each proxy in a particular cell is configured to execute directory service requests against the master database if the database of the particular cell is unavailable.

16. The computer system according to any one of claims 13 to 16 wherein each proxy is arranged to maintain a plurality of persistent connections to each of the database of the proxy's cell and the master database, said connections providing access to execute said SQL queries to enhance failsafe operations.