SoftMillennium

LDAP Directories And JDBC
A White Paper

Table of Contents

Executive Summary

New, innovative uses of the LDAP technology are surfacing daily. LDAP directories are implemented at businesses of all sizes and are proving to be the preferred way of storing data about people, employees, organizational units, computers, system accounts, access rights and many other categories of information found in today's enterprises. However, as of now, there is no convenient method of ad-hoc querying of the directory data and displaying it in a presentable format. Companies providing Java enabled Business Intelligence solutions have been concentrating on the RDBMS based data stores and mostly ignoring vast amounts of valuable data stored in LDAP directories. LDAP data could be cross referenced, validated and analysed using the advanced, graphical tools from vendors in the new Virtual Database field of technology. It could be, but not easily, not without the SQL based JDBC driver that is. Likewise, companies needing an ad-hoc reporting solution for LDAP must instead use expensive, inflexible "canned" reports. Enter a JDBC driver for LDAP.

Developers of data-driven applications have learned to depend on JDBC as the universal data access protocol used by J2EE application servers and many other types of data driven Java applications. This white paper will focus on benefits of accessing LDAP data using a Type 4 JDBC driver for LDAP.

JDBC Driver for LDAP addresses these important issues:

1. It provides an abstraction layer for Java applications using JDBC as the prefered data access protocol.
2. It provides a standard, popular, well understood and well supported way of accessing LDAP directories.
3. It provides a solution to ad-hoc reporting off LDAP data as JDBC is the only protocol accepted by all Java based report designers and BI applications.

Data Abstraction

Sun Microsystems'Entity EJB specification is a very good attempt to provide a standard for data abstraction model in enterprise applications. In all released versions of the EJB specification, a preference has been given to the SQL based, relational datasources. More features using the SQL standard are added to the Entity EJB specification with each new release.

Entity Beans with Container Based Persistence were a remarkable idea. They gave developers freedom of swapping databases in their applications without having to change any of the java code. With version 2.0 of the EJB specification, a solid effort was made to provide a powerful data abstraction mechanism through SQLJ.

What stands out is that the single, most important prerequisite to use all these great features described above is the required use of SQL statements through a JDBC driver.

JDBC specification has been one of the most successful java technologies of all times. Included since version 1.0 of the java specification, it had already gone through several revisions, each adding a new subset of APIs. JDBC is proving to be the mainstay of enterprise data access and one couldn't imagine a J2EE implementation without JDBC being at its core.

There is a JDBC driver written for every major RDBMS system and other types of data stores shouldn't be left out. LDAP directory is an example of a data store that hasn't fully benefitted from the features offered by the java application servers. JDBC driver for LDAP can change that.

A lot of existing enterprise applications need to look up the information about a person, her accounts, access rights, and roles in the organization. Typically, to be usefull, such information needs to be stored in a relational database and retrieved using a JDBC driver. However, LDAP directories are optimized for reading such data, are much better suited for storing it. Their inherent hierarchical layout of information makes sense when defining a structure of an organization. With the recent push for a single point of authentication and authorization, LDAP directories again were choosen to be the prefered means of achieving these goals. With the use of LDAP on the rise, it makes sense to investigate a standard way of accessing directory data by using a JDBC driver.

Existing applications currently accessing RDBMS systems through a JDBC driver may be easily rewritten to use a JDBC driver for LDAP and let the directory become the intended central repository of organizational data. By using a JDBC driver, the authors of existing applications already have taken the first most important step in achieving the abstraction layer between the application and the underlying datastore. With JDBC driver for LDAP, the transition to use LDAP directory as the single datasource containing the organizational information and available to many applications, may be as simple as remapping of table names to objectclass names. Authors of new applications should definitely consider an LDAP directory and the JDBC driver for LDAP as the mechanism of looking up information about people's access rights. Consequently,

application designers currently considering the use of JNDI API to access directories may want to research accessing directories through JDBC and for example take advantage of the built-in features of an CMP EJB.

Benefits Of Using A Mature Standard

There are many important reasons to use a JDBC driver to connect to an LDAP directory:

- Ability to access numerous directories from all vendors supporting LDAP v.2 or v.3 protocol in the same way as any other JDBC enabled database.
- Large number of developers who understand the benefits and have experience using the JDBC set of APIs.
- Support for JDBC in third party tools like application servers and report designers, which ranslates into immediate benefits of being able to easily access directory data using already written JDBC code.
- Large amount of easily available information including books, examples and documentation.

Leveraging development efforts with proven technologies is the key to success in all software projects. Experienced system architects and software engineers tend to use already written code and avoid “reinventing the wheel”. Estimates of JDBC based development efforts are usually accurate and the efficiency of the development team is apparent in most well managed projects.

JDBC driver for LDAP is one Java component which will allow software designers to access a goldmine of yet undiscovered riches, which is directory data, by using a mature set of Java APIs.

Ad-Hoc Reporting Using LDAP Directories

Raw, unorganized data offers very little in terms of analytical benefits. In modern enterprises, realtime access and analysis of ever changing data presents one of the bigger challenges. Of course there are tools which make this job easier, but most of them don't have capabilities of accessing LDAP based data and being able to report off it. Point in case would be Java based Business Intelligence tools, which so many large companies depend on for making "on the spot" business decisions. Lack of the ability to conveniently cross reference data from LDAP directories with data from other sources is obvious. Even to create the simplest of reports, without a JDBC driver for LDAP, one almost always needs to hard code the formatting of data and offer reports in a "canned", inflexible format. All vendors in the BI, Virtual Database and reporting spectra offer applications and tools with access to data sources through a JDBC protocol. With an addition of a JDBC driver for LDAP, a user of such applications gains an immediate ability to design and run professionally looking, real-time, ad-hoc reports and analysis of LDAP data - a treat simply not available without the JDBC driver.

By being able to use the combination of the powerful SQL language, multi datasource joins and custom data formatting functions, which most of the report tools offer, one has virtually unlimited ability to query and cross-reference LDAP data without having to write a single line of code. Many java reporting vendors offer server aplications with an embedded report engine and features allowing the end user to define custom Access Control Lists. Such features and JDBC driver for LDAP make it possible to roll out a multiuser, enterprise wide LDAP reporting solution where each user may design publish and view reports using only this part directory's schema and data to which she was given access by the administrators of the reporting server.

Conclusion

There are alternative ways to access LDAP data. Behind the scenes, every X.500 directory supporting LDAP must persist its data in some kind of data store. There is no standard for the type of database to be used in directories. Each directory server vendor must implement its own way of persisting the data and it varies widely from one vendor to another. Many times it is a highly indexed relational database, for which there already might be a JDBC driver available on the market. One might wonder if it would be more efficient and easier to access these backend directory databases directly. The answer is negative. Implementations of directory backend data storage are not published and are subject to change without a warning from one product release to another. There are many directory vendors in the market and it wouldn't be practical or even wise having to guess the undocumented features of a backend datastore implementation and then having to keep up with the changes. One should also mention the easily imaginable breaches of data access security possible with such approach.

Other Java protocols such as JNDI are not widely used in SQL based data mining and reporting applications and therefore do not offer the benefits of the wealth of third party tools using JDBC.

A new JDO (Java Data Objects) Java specification from Sun Microsystems offers a promise by attempting to further diminish the differences between numerous methods with which multisource data may be accessed. The specification however is still at its early stages, and it looks like it will be difficult to implement and will take time to be widely accepted in the marketplace.

The various ways in which a Java application may use a JDBC driver are countless, but the major benefits are always the same: low implementation cost, data abstraction, reliability.

Therefore, accessing directory data through the JDBC-LDAP bridge offers at the moment a viable solution with many immediately available benefits.

JDBC may very well be the bearer of the Universal Data Access title. When a Java developer needs to access a new type of data store, first thing she does is checking the availability of a JDBC driver for it. From import/export routines through data abstraction to reporting, one cannot imagine these tasks done in Java applications without JDBC drivers. With the addition of a JDBC driver for LDAP, end users gain access to rich sets of data from a growing myriad of LDAP directory implementations throughout the world.

Appendix

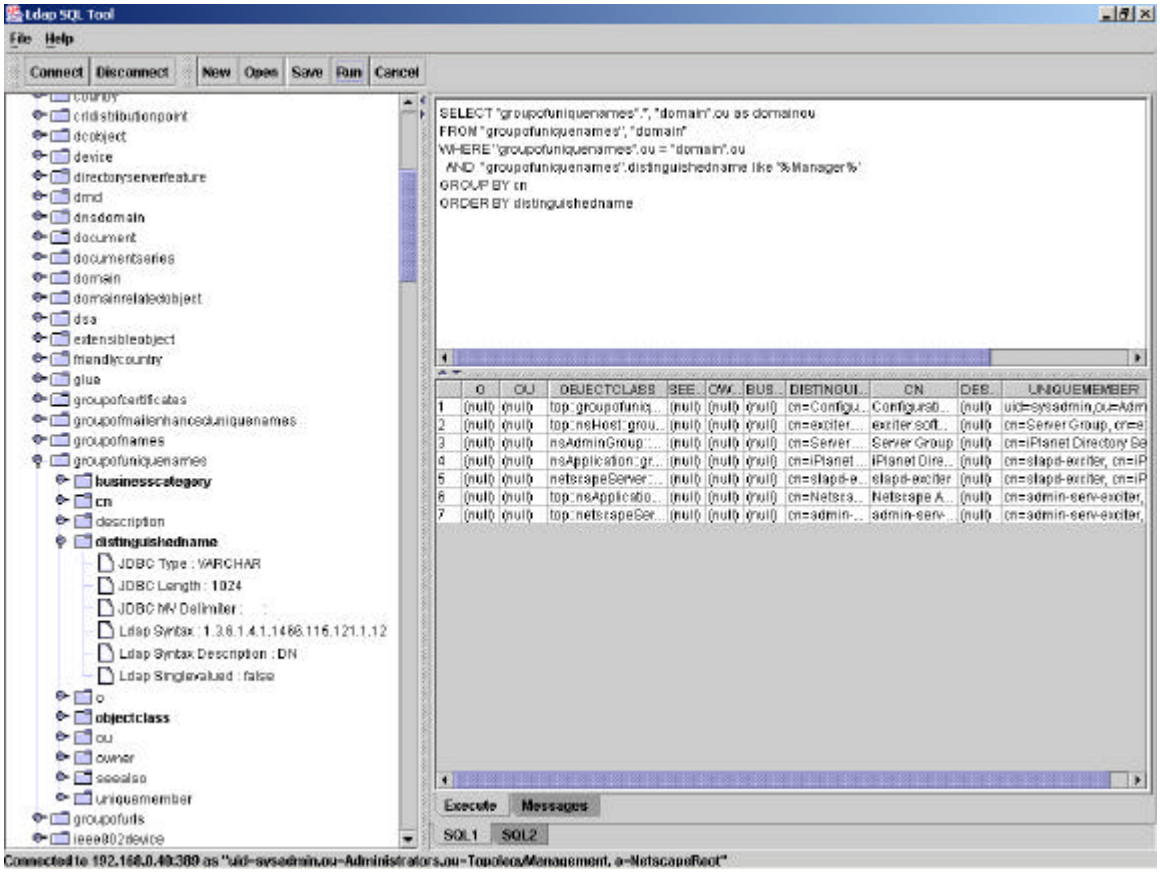
How Does JDBC Driver For LDAP Work?

Mapping of the LDAP object classes to relational tables is straight forward. Each objectclass becomes a table with columns corresponding to objectclass' attributes. When attribute is multivalued, all values are represented as a single, user delimited value in a column. All inherited objectclass attributes are also present as columns in the table's schema. Furthermore, mapping between the data types is also possible in a configuration file read by the driver at initialization time.

One important issue facing designers of JDBC drivers for LDAP is the unpredictably large number of objects retuned from wildcard directory queries. Although LDAP directories are optimized for read operations, concurrent requests for large amounts of data may cause delays and even overwhelm the resources of a host machine using the driver. JDBC driver for LDAP may offer user configureable, in-memory cacheing of large segments of commonly used and mostly static data.

With configureable cache refresh time intervals, the data is never stale for too long and applications needing it get it very fast without creating a disruption in network traffic.

Screen 1. LdapSQLTool provided with SoftMillennium's JDBC driver for LDAP



Screen 2. Example of LDAP schema information displayed in Actuate's e.Report Designer