データベースシステム 128−43
（２００２．７．１８）

SuperSQLによるLDAPデータの自動生成

古思 望† 遠山 元道‡

†慶應義塾大学大学院 理工学研究科 開放環境科学専攻
‡慶應義塾大学 理工学部 情報工学科
E-mail: † koshi@db.ics.keio.ac.jp, ‡ toyama@ics.keio.ac.jp

近年インターネットの爆発的な普及により DNS などの分散ディレクトリサービスが浸透し、組織の情報をディレクトリ構造に蓄えることが多くなってきている。そのディレクトリサービスの中でも最近汎用的に利用されてきているのが LDAP(Lightweight Directory Access Protocol) である。しかし、LDAP には関係データベースが行える大量の更新やトランザクション機構、SQL を用いたリレーショナル処理がサポートされていない。また情報に冗長性があり更新操作が複雑でデータの管理が難しい。一方正規化された関係データベースでは情報の冗長性はなく更新処理が局所的でデータを管理しやすい。そこで本研究では SuperSQL を拡張し関係データベースから LDAP データを自動生成することを提案する。

キーワード : SuperSQL 、 LDAP 、 更新

Automatic generation of LDAP data using SuperSQL

Nozomu KOSHI † Motomichi TOYAMA ‡

†School of Science for OPEN and Environmental Systems,
Faculty of Science and Technology, Keio University.
‡Department of Information and Computer Science, Faculty of Science and Technology,
Keio University.
E-mail : †koshi@db.ics.keio.ac.jp ‡toyama@ics.keio.ac.jp

Distributed directory services, such as DNS, permeate by the explosive spread of the Internet in recent years, and the information of an organization is stored in directory structure more often. LDAP (Lightweight DirectoryAccess Protocol) is used general-purpose also in the directory service recently. However, a lot of updating, transaction mechanisms or the relational processing using SQL which can perform a relational database are not supported by LDAP. Moreover, redundancy is in information, updating operation is complicated, and management of data is difficult. On the other hand in the normalized relational database, there is no informational redundancy, and its updating processing is local and it tends to manage data. Then, in this research, it propose s extending SuperSQL and generating LDAP data automatically from a relational database.

keyword : SupersQL , LDAP , Update

1 はじめに

近年インターネットの爆発的な普及により DNS などの分散ディレクトリサービスが浸透し、組織の情報をディレクトリ構造に蓄えることが多くなってきている。なぜならディレクトリは木構造であり、情報が冗長に存在し検索速度が早く、また分散してサーバーが配置できるので、各階層においてデータを管理できるからである。そのディレクトリサービスの中でも最近汎用的に利用されているのが LDAP(Lightweight Directory Access Protocol) である。しかし、LDAP には関係データベースが行える大量の更新やトランザクション機構、SQL を用いたリレーショナル処理がサポートされていない。また情報に冗長性があり更新操作が複雑でデータの管理が難しい。一方正規化された関係データベースでは情報の冗長性はなく更新処理が局所的でデータを管理しやすい。しかし LDAP を利用するための API(Application Programming Interface) の中でどれも関係データベースから LDAP データに変換してくれるものはない。そこで本研究では質問文の結果から様々な媒体への出力が可能な SuperSQL を用いて関係データベースから LDAP データを自動生成することを提案し、効率化をはかる。以下、2 章で LDAP 、3 章で、LDAP における SuperSQL の演算子、装飾子の対応と実行例について述べる。4 章で具体例を用いて LDAP 側ではなく関係データベース側でデータを管理することの効率の良さを評価する。最後に今後の展望・課題を 5 章で、結論を 6 章で述べる。

2 LDAP

LDAP はディレクトリサービスを汎用的に扱えるようにプロトコルを定義し、インターネット上で、サーバーを分散して管理できる。

2.1 LDAP モデル

LDAP は 4 つのモデルに分類され、特に以下の節で情報モデルとネーミングモデルについて詳しく説明する。

- 情報モデル——LDAP ディレクトリに、どういった種類の情報を格納できるかを定義する。
- ネーミングモデル——LDAP ディレクトリにある情報の構成の仕方や、参照する方法を定義する。
- 機能モデル——LDAP ディレクトリにある情報を用いて、どのようにアクセスしたり、更新、追加、削除したりできるのかを定義する。
- セキュリティモデル——LDAP ディレクトリにある情報を、権限を持たないアクセスや修正から、いかにして守るかを定義する。

2.1.1 情報モデル

LDAP はディレクトリ構造をとっていて、1 つ 1 つのエントリー (ノード) は現実の世界に存在するオブジェクト概念 (例えば人や組織など) であり、それらに関する名前、mail などの情報を保持している。エントリーは名前、mail などの情報を属性 (atrribute) として持ち、各属性には 1 つ以上の値 (value) がある。属性の中で objectclass は、エントリーがどのような属性を持たなくてはいけないか、どのような属性を持つことができるか、とういことを表すための特別の属性名である。

2.1.2 ネーミングモデル

エントリーを 1 つ 1 つ識別するためにそれぞれのエントリーには識別名 (DN) が付いてる。識別名は DNS と同じように階層構造のトップからツリーをたどるごとに後ろから表現していく。例えば「cn=ito,o=CompanyA」の識別名は CompanyA の下にいる伊藤を表す。識別名に用いられる属性名はそのエントリーの中にある属性ならどれでもよい。

2.2 LDIF

ディレクトリ中のエントリーの情報をテキスト形式で表現する記述方法を LDIF(LDAP Data Interchange Format) という。これでディレクトリ全体の情報をテキスト形式で表現することが可能になる。1 つのエントリーに対して必ず、識別名、オブジェクトクラス、属性を記述しなくてはならない。例えば、ある会社 CompanyA の支店である Kawasaki エントリーの LDIF は以下のようになる

```
dn:ou=Kawasaki,o=CompanyA
objectclass:organizationalUnit
obectclass:top
ou:Kawasaki
```

3 SuperSQL

SuperSQL は SQL を拡張したデータベース出版言語であり、HTML、Java、LaTeX 、Excel ワークシート、VRML、XML 等の様々な媒体上で多様なレイアウトのドキュメントを生成することが可能である。本研究では LDAP データをテキスト形式で表現する LDIF を生成する。

SuperSQL の質問文は、SQL の SELECT 句を $GENERATE < medium >< TFE >$ の構文をもつ GENERATE 句で置き換えたものである。$< medium >$ で出力媒体の指定を行う。今回は LDAP である。$< TFE >$ はターゲットリストの拡張である Target Form Expression を表し、結合子、反復子などのレイアウト指定演算子を持つ一種の式である。

3.1 LDAP の実現

本研究では SuperSQL の多彩なレイアウト表現によって LDAP の階層的なディレクトリ構造を、また装飾子によってエントリーの持つオブジェクトクラス、属性名を出力する。

3.2 SuperSQL 演算子の実装

SuperSQL 質問文では、水平、垂直、深度の各次元の結合子を「,」「!」「%」で表現している。またインスタンスがある限り対応する次元方向に繰り返しを反復子「[]」で表現する。LDAP において SuperSQL の演算子の対応の仕方について説明する。

3.2.1 LDAP における演算子の対応

- 水平連結子 (,)、水平反復子 ([],)
  1 つのエントリが持つ属性を続けて LDIF に出力させることができる。
- 垂直連結子 (!)、垂直反復子 ([]!)
  同じ親を持つ兄弟のエントリー同士を続けて LDIF に出力させることができる。
- 深度連結子 (%)、深度反復子 ([]%)
  親子関係にあるエントリーを続けて LDIF に出力させることができ、親の識別名を継承する。

3.3 SuperSQL 装飾子の実装

SuperSQL では、質問文の中で多彩な装飾指定を行うことができ、これは質問文の属性の後に装飾指定子「@装飾指定」を記述することで実現される。LDAP において SuperSQL の装飾子の対応の仕方について説明する。

3.3.1 LDAP における装飾子の対応

LDAP 生成用の装飾子は 3 つある。

- objectclass
  エントリーがもつオブジェクトクラスを指定する。
- att
  関係データベースの属性名を LDAP で指定された属性名に変換する。

- att-alias
  alias エントリーの場合は att-alias で実際のデータの持つエントリーがあるサブツリーを指定する。

3.4 実行例

図 1 のスキーマを持つデータベースがある。この関係データベースでは会社 CompanyA には支店があり、その支店には売場がある。そしてその売場には従業員が働いている。それらの情報が格納されている関係データベースから SuperSQL を用いて root ノードに CompanyA のエントリーがあり、その下に支店のエントリー、売場のエントリー、従業員のエントリーと階層を深くしていく LDAP ディレクトリを作成する。このとき、SuperSQL 質問文は図 3 のようになり、生成される LDIF を図 4 に、そしてそのディレクトリ構造を図 2 に示す

会　社

ID	会社名

支　店

ID	支店名	会社 ID	TEL

売　場

ID	売場名	支店 ID	TEL

従 業 員

ID	従業員名	売場 ID	会社 ID	TEL	Mail

図 1: サンプルデータベースのスキーマ

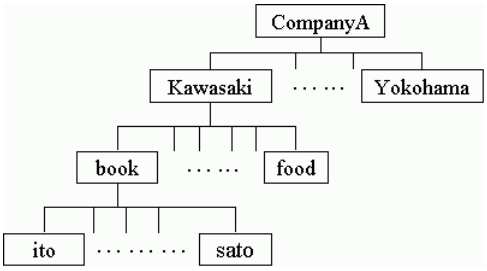

図 2: 生成されるディレクトリ構造

```
GENERATE ldap [
  {c.name@{att=o}}
  @{objectclass=organization
             ,objectclass=top}%
  [{s.city@{att=ou}}
     @{objectclass=organizationalUnit
                 ,objectclass=top}%
  [{d.name@{att=ou}}
    @{objectclass=organizationalUnit
                ,objectclass=top}%
  [{e.name@{att=cn},e.name@{att=sn}
                  ,e.mail@{att=mail}}
  @{objectclass=top,objectclass=person
    ,objectclass=inetorgperson
    ,objectclass=organizationalPerson}
   ]!]!]!]!

FROM company c,store s,dept d,employee e
Where s.company_id=c.id and
     d.store_id=s.id and e.dept_id=d.id;
```

図 3: SuperSQL 質問文

```
dn:o=CompanyA
objectclass:organization
objectclass:top
o:CompanyA

dn:ou=Kawasaki,o=CompanyA
objectclass:organizationalUnit
objectclass:top
ou:Kawasaki

dn:ou=book,ou=Kawasaki,o=CompanyA
objectclass:organizationalUnit
objectclass:top
ou:book

dn:cn=ito,ou=book,ou=Kawasaki,o=CompanyA
objectclass:top
objectclass:person
objectclass:inetorgperson
objectclass:organizationalPerson
cn:ito
sn:akinori
telephoneNumber:044977 ××××
mail:ito@company.co.jp
.....
```

図 4: 出力結果

4 評価

LDAP ディレクトリを更新した場合における本システムの有効性について評価をする。

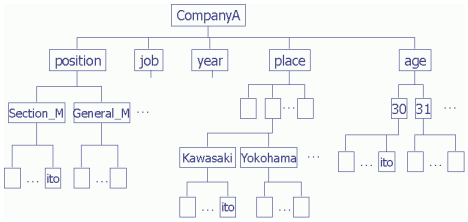

図 5: 例として用いるディレクトリ構造

今回例として用いる LDAP ディレクトリは図 5 であり、CompanyA の情報に関する LDAP ディレクトリである。地位 (position), 仕事 (job) , 入社年度 (year), 事業所 (place), 年齢 (age) で分類し、社員の情報を管理している。これを生成する関係データベースは図 1 のように設計し、図 3 のようにクエリーを記述すれば自動生成できるが、ここではクエリーが複雑になりすぎるので省略する。このときの更新、追加における本システムを用いる場合と用いない場合とで比較を行う。

4.1 更新における評価

例えば、社員 ito は課長 (Section_M) から部長 (General_M) に昇進し、Kawasaki から Yokohama に異動になり、年齢が 30 歳から 31 歳になったときの更新を考える。

- 本システムを用いない場合

1. ito エントリーを Section_M のサブツリーから General_M のサブツリーに移動させる。

   ・Section_M のサブツリーから ito エントリーを削除

```
$ldapdelete -D  "cn=Manager,o=CompanyA"
   -w secret "cn=ito,ou=Section_M,
                ou=Position,o=CompanyA"
```

   ・General_M のサブツリーに ito エントリーを追加

```
$ldapadd -D "cn=Manager,o=CompanyA"
            -w secret -f < addfile
```

```
dn:cn=ito,ou=General_M
                ,ou=Position,o=CompanyA
objectclass:top
objectclass:person
objectclass:inetorgPerson
cn:ito
mail:ito@companya.co.jp
telephoneNumber:045977 ××××
```

図 6: addfile の内容

2. ito エントリーを Kawasaki のサブツリーから Yokohama のサブツリーに移動させる。1 と同様に Kawasaki のサブツリーから $ldapdelete を用いて ito エントリーを削除し Yokohama のサブツリーに $ldapadd を用いて ito エントリーを追加する。

3. ito エントリーを 30 のサブツリーから 31 のサブツリーに移動させる。1 と同様に 30 のサブツリーから $ldapdelete を用いて ito エントリーを削除し、31 のサブツリーに $ldapadd を用いて ito エントリーを追加する。

- 本システムを用いる場合

1. 関係データベース側で ito タプルの Position 属性の Section_M 番号から Geberal_M 番号に、Place 属性を Kawasaki 番号から Yokohama 番号に、age 属性を 30 から 31 に変更する。

   図 7 は Position 属性の番号を更新する操作であり、これを同様に Place 属性と age 属性を更新する。

2. ディレクトリ構造は変わらないので最初に作成した SuperSQL 質問文を実行する。

```
UPDATE employee
SET position= 'Secion_M番号'
WHIRE position-'General_M番号';
SELECT *
FROM employee
WHERE name='ito';
```

図 7: Position 属性の更新

3. できた LDIF ファイルを LDAP サーバーに格納。

4.2 更新における評価のまとめ

2 つの更新処理に対する比較を行う。

- 本システムを用いない場合
  削除、追加の操作を繰り返すので更新処理が面倒で、ミスすることも多くディレクトリの一貫性が失われることがある。例えばSection_M から ito エントリーを削除し忘れたらディレクトリの一慣性を失う。
- 本システムを用いる場合
  関係データベース側で 1 人の社員にたいして 1 つのタプルの属性を変えるだけでよく更新が局所的でミスはなく常に一貫性のある LDAP ディレクトリが生成できる。

4.3 追加における評価

図 5 を見ると 1 人の社員に対して 5 つのエントリーを作らなくてはならない。このとき 2002 年に 10 人の新入社員が入社してくるときの更新を考える。

- 本システムを用いない場合
  図 8 のように社員 1 人に対して 5 エントリー

```
$ldapadd -D “cn=Manager,o=CompanyA” -
w secret < addfile
```

分のファイルを記述しなくてはならない。これが 10 人分のファイルがあり全部で 50 エントリー追加する。

```
dn:cn=goto,ou=common,ou=position,o=CompanyA
objectclass:top
objectlcass:person
objectclass:inetorgPerson
cn:goto
mail:goto@companya.co.jp
telephoneNumber:044955 ××××

dn:cn=goto,ou=22,ou=age,o=CompanyA
objectclass:top
objectlcass:person
objectclass:inetorgPerson
cn:goto
mail:goto@companya.co.jp
telephoneNumber:044955 ××××

                :
                :
```

図 8: 社員 1 人の addfile の内容

- 本システムを用いる

1. 図 9 のように関係データベース側で 10 人のタプルを加える。

```
insert into employee values(1020,'goto',
        1,110,201,goto@companya.co..jp',
                  '044955 ××××''22')
```

図 9: 1 人分のタプルの追加

2. ディレクトリ構造は変わらないので最初に作った SuperSQL 質問分を実行。このとき冗長である 50 エントリーが自動生成される。
3. できた LDIF ファイルを LDAP サーバーに格納

4.4 追加における評価のまとめ

正規化された関係データベースでは情報の冗長性はなく無駄なタプルを追加しないですむのでファイ

ルに書き込む記述量は少なくてすむ。2つの追加処理について比較する。

- 本システムを用いない場合
  冗長である 50 エントリー分の識別名、オブジェクトクラス、属性名をファイルに記述しなくてはならない。
- 本システムを用いる場合
  10 タプル分をファイルに記述し関係データベースに格納するだけでよい。冗長であるデータはシステムが自動的に生成する。

5 課題・展望

本研究によって関係データベースから LDAP データに変換できるようになったが現在のところ、関係データベースで更新されたデータのみを LDAP サーバー側で更新する差分更新ができていない。更新が行われても関係データベースにあるすべてのデータから LDAP データを生成するので一括生成となってしまう。差分更新を実現する方法として図 10 のようにすればよいと考えている。更新が行われたタプルのみの VIEW を作り、それに対して SuperSQL 処理系からアクセスし、修正ファイルを生成し更新されたデータのみを LDAP サーバーに格納する。このようにして差分更新ができるようになれば更新が行われたときに LDAP サーバーをロックする時間が短くなり効率がよくなる。

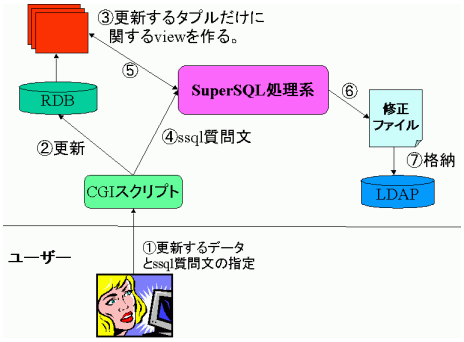

図 10: 差分更新

6 まとめ

本研究によって SuperSQL を利用して関係データベースから一貫性のある LDAP ディレクトリが生成されることが実現された。LDAP では、情報に冗長性があり検索する速度が早いが、情報の冗長性のため情報の管理が大変である。それに対し正規化された関係データベースでは情報に冗長性はなく、情報の管理がスマートである。このシステムにより情報の管理を LDAP 側ではなく関係データベース側で行うことができ、効率的となることを 4 章で証明した。今後ますます LDAP は普及していくと見られ、組織が大きくなるほど LDAP におけるデータの管理が複雑となり、また LDAP より関係データベースの方が親しみやすい点からこのシステムは重要さは増してくるであろう。

今後の課題としては 5 章で取り上げた差分更新の実現と SuperSQL 質問文の分割による SuperSQL 処理系の最適化があげられる。

参考文献

[1] SuperSQL: http://ssql.db.ics.keio.ac.jp/

[2] Motomichi Toyama, “SuperSQL: An extended SQL for Database Publishing and Presentation,” *ACM SIGMOD ’98* , pp. 584-586, 1998

[3] T. Seto, T. Nagafuji and M. Toyama, “Generating HTML Sources with TFE Enhanced SQL,” *Proc. ACM Symp. on Applied Computing(SAC ’97)*, pp. 96-105, 1997

[4] Sihem Amer-Yahia,H.V.Jagadish,Laks V.S. Lakshmanan and Divesh Srivastava “On Bounding-Schemas for LDAP Directories“ *EDBT2000*, pp.287-301, 2000

[5] 高畑 理、藤沼健太郎、石橋 玲、遠山元道「Magic Mirror Mailing: 個人情報データベースを利用する柔軟なメール配送システム」データ工学ワークショップ論文集、2001.