SuperSQL を用いた RDB と LDAP の統合管理における差分更新手法

Incremental Updates in RDB and LDAP Integration using SuperSQL

古思 望† 遠山 元道‡

Nozomu KOSHI Motomichi TOYAMA

LDAP(Lightweight Directory Access Protocol)とは分散ディレクトリサービスの１つであり、様々な情報管理に利用されている。しかし、データに冗長性がある場合、管理が非効率的である。したがって、データに冗長性がない正規化された関係データベースとの統合利用が効率的である。SuperSQL によって、関係データベースから LDAP への変換は実装されているが、関係データベースでデータが更新された場合、LDAP サーバーを更新するには、再びディレクトリ全体を生成し直さなくてはならない。本研究では、関係データベースの更新に対し、LDAP サーバーの更新を最小化する差分更新の手法を提案する。

LDAP （Lightweight Directory Access Protocol） is distributed directory service and is used for many information management. Since the tree structured nature of LDAP has redundant data, it is more efficient to use a hybrid of a RDB and LDAP server. SuperSQL made it possible to integrate RDB and LDAP server with translation capability from former to latter. Updating on RDB, we have to recompute the whole LDAP directory. In this paper we propose incremental update for LDAP directory about updated tuples on RDB through SuperSQL system.

1. はじめに

近年インターネットの普及により、分散ディレクトリサービスが浸透してきている。その中で汎用的に利用されてきているのがLDAP(Lightweight Directory Access Protocol)[4]であり、DNSやNISの代用ができる。またLDAPスキーマの柔軟性によりそれぞれの情報の異種性を容易に扱え、組織内での各種の情報管理にも使用されている[3]。データに冗長性があり検索速度が早いという利点はあるが、更新時において複数のエントリーを更新するため管理が高コストであるという欠点を持っている。したがってデータに冗長性がなく管理が容易である関係データベースと統合利用した方が効率的である[1]。関係データベースとLDAPの統合利用はSuperSQLによって実現し、関係データベースに格納されているデータをLDAPディレクトリに変換するシステムが実装されている[7]。しかし、関係データベースが更新された場合、LDAPサーバーを更新するには、再びディレクトリ全体を生成し直さなくてはならない。そこで本研究では、関係データベースの更新に対し、LDAPサーバーの更新を最小化する差分更新の手法を提案する。以下2章でLDAP、3章でSuperSQLによるLDAPディレクトリの自動生成システムについて説明し、4章で差分更新の方法を示す。そして最後に結論と課題を述べる。

† 学生会員　慶應義塾大学大学院理工学研究科修士課程
koshi@db.ics.keio.ac.jp
‡ 正会員　慶應義塾大学理工学部情報工学科
toyama@ics.keio.ac.jp

2. LDAP

2.1 LDAP モデル

LDAP はディレクトリ構造をとり、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)に対応する。エントリーは名前、mail などの情報を属性(attribute)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 は特別な属性であり、そのエントリーが持つべき属性を決定する。またエントリーを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)が付いており、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下位のノードから表現する。例えば「cn=伊藤, ou=情報, o=理工」の識別名は理工学部情報工学科のサブツリーに存在する伊藤というエントリーを表す。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であればどれでもよい。

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。これにより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となる。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、理工学部にある情報工学科のエントリーの LDIF は以下のようになる。

```
dn: ou=情報、o=理工
objectclass: organizationalUnit
ou: 情報
telephoneNumber: 045223××××
```

3. SuperSQL による LDAP ディレクトリの自動生成

SuperSQL[5],[6]を用いて、関係データベースに格納されているデータを LDAP ディレクトリに変換することができる。この章では SuperSQL について説明し、SuperSQL の持つ演算子と装飾子の LDAP における対応について述べる。最後に LDAP ディレクトリを自動生成する実行例を示す。

3.1 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 、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。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い、*<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。本研究では SuperSQL により LDIF を生成し、LDAP サーバーに LDAP ディレクトリを格納する。

3.2 LDAP における SuperSQL 演算子

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。

- 水平連結子（,）、水平反復子（[ ],）

１つのエントリーが持つ属性を続けてLDIFに出力させる。

- 垂直連結子（!）、垂直反復子（[ ]!）

同じ親をもつ兄弟のエントリーを続けて LDIF に出力させる。

- 深度連結子（%）、深度反復子（[ ]%）

親子関係にあるエントリーを LDIF に親のエントリーの後に子のエントリーを出力させ、親の識別名を継承させる。

3.3 LDAP における SuperSQL 装飾子

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する。LDAP において SuperSQL の装飾子の対応の仕方について説明する。

- objectclass

エントリーが持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。

- path

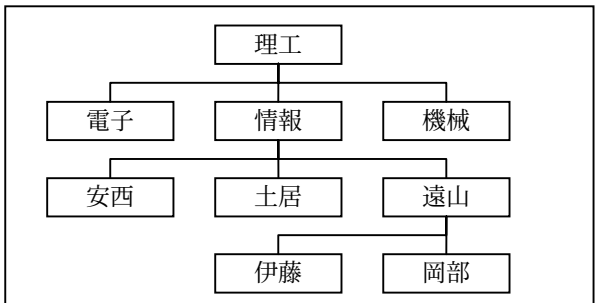
上位のテーブルが深度連結子で結合していないとき、この装飾子を用いることにより上位の識別名を継承する。

3.4 実行例

学部、学科、研究室、学生の関係データベースのスキーマを

R_1:Faculty(id, name)
R_2:Department(id, name, faculty_id)
R_3:Lab(id, name, tel, dept_id)
R_4:Student(id, name, tel, mail, lab_id)

とする。この関係データベースから生成する LDAP ディレクトリと SuperSQL 質問文を図１に示す。




```
GENERATE ldap [
  {f.name@{att=o}}
  @{objectclass=organization}%
  [{d.name@{att=ou}}
  @{objectclass=organizationalUnit}%
  [{l.name@{att=ou}}
     @{objectclass=organizationalUnit}%
  [{s.name1@{att=cn},s.name2@{att=sn}
                    ,s.mail@{att=mail}}
  @{objectclass=person
    ,objectclass=inetOrgPerson
    ,objectclass=organizationalPerson}
  ]!]!]!]!
FROM Faculty f, Department d,
                        Lab l, Student s
Where f.id = d.fculty_id and
        d.id = l.dept_id and l.id = s.lab_id;
```

図１ LDAP ディレクトリと SuperSQL 質問文
Fig.1 LDAP directory and SuperSQL query

4. 差分更新

SuperSQL により関係データベースから LDAP ディレクトリを自動生成することは可能になったが、現在のところ関係データベース側で更新されたデータを LDAP サーバー側で更新するとき、再びディレクトリ全体を生成して更新する方法しかない。よって関係データベース側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重要となる。その方法についてこの章で説明する。

4.1 １タプルの更新における差分更新

関係データベース側で更新された変化分をsource Δ（Δ_s）とする。その変化分に相当するLDAPサーバー側の変化分をtarget Δ（Δ_t）とする。差分更新の方法として関係データベース側で更新されたデータに関するサブツリーを生成するようにSuperSQL質問文を書き直せばよい[2]。テーブル$(R_1,R_2,\cdots\cdots,R_k)$から SuperSQL質問文(S)を使ってLDAPディレクトリ(T)を生成するとき以下のように表す。

$$T = S_{\text{Join の条件}}(R_1,R_2,\cdots\cdots,R_k)$$

$R_j(1 \leqq j \leqq k)$に挿入、削除、修正されたタプルをi_{Rj},d_{Rj},u_{Rj}とする。R_j にタプル $t_j(i_{Rj},d_{Rj},u_{Rj})$を適用し、新しくできたテーブルを$R_j$’とする。更新の結果新たにできるディレクトリT’は再び全体のディレクトリを再計算するならば

$$T = S_{\text{Join の条件}}(R_1,R_2,\cdots,R_j\text{’}\cdots,R_k)$$

である。しかし LDAP サーバー側の変化分Δ_t を求めた上で、元のディレクトリTに適応させる差分更新をすることでより効率的になる。このとき LDAP 側の変化分

$$\Delta_t = S_{\text{Join の条件}}(R_1,R_2,\cdots,t_j\cdots,R_k)$$

と表すことができる。これを元にディレクトリTに適用させることによりT’を求めることが可能になる。しかし求めたΔ_t が必ずしも更新する最小のサブツリーとは言えない。なぜならば、タプルを更新したとき、LDAP 上でそれに関する冗長なエントリーも生成してしまうからである。LDAP では冗長なエントリーも更新可能であり、このまま更新しても正しい結果が得られるが最小の LDAP ディレクトリを更新できるようにしたい。冗長な更新をしない更新を最小部分更新と定義する。

4.2 最小部分更新

最小部分更新とは RDB 側で更新された変化分に対して、LDAP 側で最小のサブツリーを更新する変化分である。これを $\Delta_t(\min)$と表す。$\Delta_t(\min)$を生成するためには SuperSQL 質問文をそのように書き直す必要がある。関係のないテーブルは削除し、上位のテーブルを削除すると識別名が継承できなくなるため、SuperSQL の装飾子 ”path ” を利用して継承できるようにする。前述した例の SuperSQL 質問文を書き直すと以下のようになる。

$$\Delta_t(\min) = S^{\text{path="上位の識別名"}}_{\text{joinの条件}}(R_1,\ldots t_j,\ldots R_m)$$

但し,$1 \leqq l \leqq k, 1 \leqq m \leqq k, l < m$

path は上位の継承すべき識別名を表す。本研究ではどのようなパターンの更新が関係データベース側で起こったとして

も、常に $\Delta_t(\min)$の LDAP ディレクトリを生成できるようにする。しかし $\Delta_t(\min)$が必ずしも LDAP ディレクトリを更新する最小コスト更新であるとは限らない。LDAP サーバーは$\Delta_t(\min)$ではなく Δ_t でも更新可能であり、$\Delta_t(\min)$を生成するのに関係データベースからあらかじめ上位の識別名を取得するなどコストもかかるが、最小部分更新を求める理由は以下の２つである。１つ目は図２のように LDAP サーバーを分散して管理する場合、 Δ_t では上位の冗長なエントリーも生成してしまいすべての LDAP サーバーを更新する必要があるが、 $\Delta_t(\min)$では冗長なエントリーを生成しないので、最小限の LDAP サーバーだけの更新で済む。２つ目は上位のテーブルを削除することにより無駄な Join をする必要がない。しかしその反面上位の識別名をあらかじめ関係データベースから取得し、SuperSQL 質問文を書き直す必要がある。

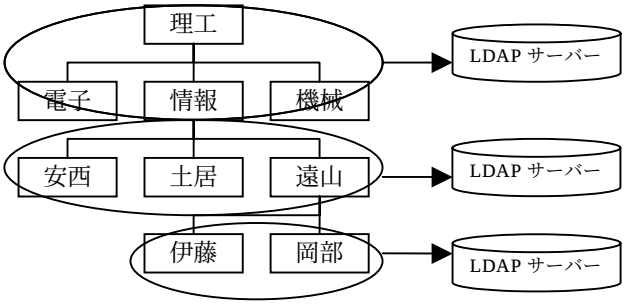


図２ LDAP サーバーの分散管理
Fig.2 Distributed Management for LDAP server

4.3 システム構成図

図３は本システムの構成図である。生成クエリーQ_Gによって生成されるディレクトリが LDAP サーバーに格納されているとする。このときユーザーが生成するクエリーQ_Gと関係データベースの変化分 Δ_S を差分更新コントローラーに与えると、システムが、更新に関する一時的なテーブルを作成し、最小部分更新クエリー$Q_U(\min)$を生成する。そのクエリーが SuperSQL システムを通して、元のテーブルと一時的なテーブルにアクセスし、最小部分更新の LDAP ディレクトリ（$\Delta_t(\min)$）を生成し、LDAP サーバーを更新する。

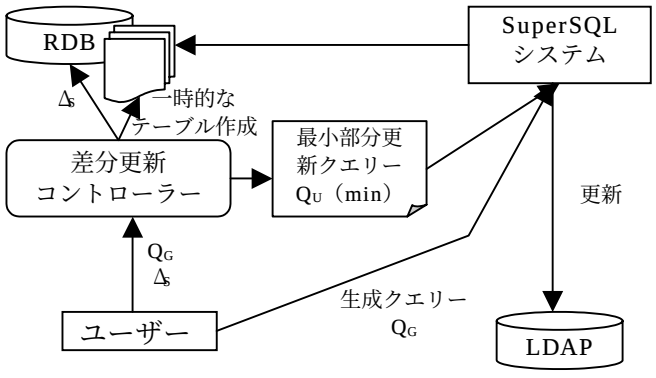


図３ LDAP サーバーの分散管理
Fig.3 System overview

4.4 １タプルの差分更新の例（挿入）

関係データベーススキーマを３章で述べたものとし、生成する LDAP ディレクトリ T と SuperSQL 質問文 S を図１とする。T は S と R_1、R_2、R_3、R_4 を使って以下のように表せる。

$$T = S_{R_1.id=R_2.faculty\_id \text{ and } R_2.id=R_3.dept\_id \text{ and } R_3.id=R_4.lab\_id}(R_1,R_2,R_3,R_4)$$

関係データベース側に清原研究室のタプルが挿入された場合、

$$\Delta_S = \text{insert into } R_3 \text{ values}(11, \text{`清原`}, 43\times\times\times, 1)$$

となる。このタプルに関するエントリー全体を生成する Δ_t は

$$\Delta_t = S_{11=R_4.lab\_id}(R_1,R_2,R_3,R_4)$$

である。しかしこれでは関係のないエントリー、すなわち研究室のエントリーより上位のエントリーも生成してしまう。したがって LDAP 側の最小の変化量である最小部分更新$\Delta_t(\min)$を生成するには上位のテーブルを削除し、SuperSQL の装飾子 ”path ” を用いて以下のようにする。

$$\Delta_t(\min) = S^{path="ou=情報, o=理工"}_{11=R_4.lab\_id}(i_{R_3}, R_4)$$

この式は清原研究室のエントリーを root とするサブツリーの生成であり、その最小部分更新 SuperSQL 質問文は図４のようになる。また生成される Δ_t と $\Delta_t(\min)$を図５に示す。

```
GENERATE ldap
  [{l.name@{att=ou}}
    @{objectclass=organizationalUnit}%
  [{s.name1@{att=cn}, s.name2@{att=sn},
    s.mail@{att=mail}}
  @{objectclass=person,
    objectclass=inetOrgPerson,
    objectclass=organizationalPerson}
   ]!]!@{path = "ou=情報, o=理工"}
FROM Lab l, Student s
Where 11 = s.lab_id;
```

図４ 最小部分更新 SuperSQL 質問文
Fig.4 Minimum update SuperSQL query

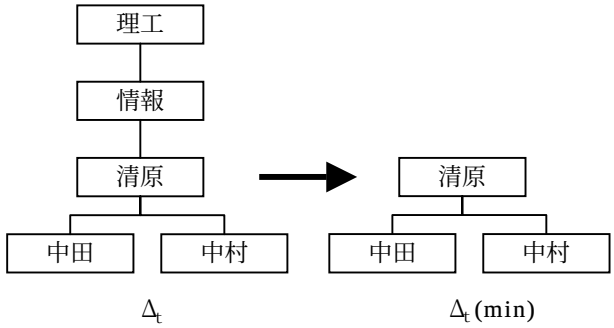


図５ Δ_t と $\Delta_t(\min)$
Fig.5 Δ_t and $\Delta_t(\min)$

このクエリーから生成された LDAP ディレクトリ $\Delta_t(\min)$を元の LDAP ディレクトリ T に加えることで、更新後の LDAP ディレクトリ T ’ を生成することができる。削除は挿入の逆であり、修正は生成した $\Delta_t(\min)$に対するサブツリーを元のディレクトリから削除してから$\Delta_t(\min)$を加える。

4.5 複数タプルの差分更新

これまでは、１つのタプルに関する更新を議論した。しかし実際には複数テーブルにタプルが挿入される、挿入と削除が同時に行われる、１つのテーブルに複数のタプルが更新されるなどいろいろな更新パターンがある。基本的には、挿入、削除は別々に考え、修正は削除してから挿入する。学科以下

のエントリーを生成する関係データベーススキーマ
R_1:Department(id, name)
R_2:Lab(id, name, tel, dept_id)
R_3:Student(id, name, tel, mail, lab_id)
を例として用い、生成される LDAP ディレクトリ T は

$$T = S^{path="o=理工"}_{R_1.id=R_2.dept\_id \text{ and } R_2.id=R_3.lab\_id}(R_1,R_2,R_3)$$

と表される。１つのテーブルに複数のタプルが更新される場合と複数のテーブルが更新される場合について考える。

- １つのテーブルに複数のタプルの挿入（削除・修正）

このとき同じ親を持つタプル同士は親までのパスを指定し、まとめて同じ SuperSQL 質問文で処理する。しかし親が違うならばパスの指定が異なるので、違う SuperSQL 質問文で処理しなくてはならない。例えば遠山研究室の学生が複数挿入されたとすると

$$\Delta_t(\min) = S^{path="ou=遠山, ou=情報, o=理工"}_{8=i_{R_3}.lab\_id}(i_{R_3})$$

の１つの質問文で処理することができるが遠山研究室と安西研究室の学生が複数挿入されると

$$\Delta_{t1}(\min) = S^{path="ou=遠山, ou=情報, o=理工"}_{8=i_{R_3}.lab\_id}(i_{R_3})$$

$$\Delta_{t2}(\min) = S^{path="ou=安西ou=情報, o=理工"}_{9=i_{R_3}.lab\_id}(i_{R_3})$$

の２つの質問文を実行することになる。削除、修正も同様である。

- 複数のテーブルに挿入（削除・修正）

このときそれぞれのテーブルに更新の有無を表す更新フラグテーブルを定義する。この関係にある $B_i(1 \leqq i \leqq k)$ は R_i に関する変数であり 0 か 1 を持ち、0 は更新されていないテーブル、1 は更新が起きたテーブルを表している。３つのテーブルに挿入された場合の更新フラグテーブルを表１に示す。すなわち、３つのテーブルに挿入されたとき表１の１行目以外の全ての SuperSQL 質問文を実行することになる。さらに最小部分更新にするには、それぞれの SuperSQL 質問文に対して上位のパスを指定して、冗長なテーブルを排除すればよい。

表１　更新フラグテーブル
Table.1　Update flag table

B1	B2	B3	LDAP ディレクトリ
0	0	0	$S(R_1, R_2, R_3)$
0	0	1	$S(R_1, R_2, i_{R3})$
0	1	0	$S(R_1, i_{R2}, R_3)$
0	1	1	$S(R_1, i_{R2}, i_{R3})$
1	0	0	$S(i_{R1}, R_2, R_3)$
1	0	1	$S(i_{R1}, R_2, i_{R3})$
1	1	0	$S(i_{R1}, i_{R2}, R_3)$
1	1	1	$S(i_{R1}, i_{R2}, i_{R3})$

例えば R_2 に新しい研究室(清原研究室)が挿入され、R_3 にその研究室の生徒が挿入されるとする。すなわち R_1 には挿入されてなく R_2、R_3 には挿入されているので、表１から R_1 が 0 でありかつ１行目以外である 2、3、4 行目の SuperSQL 質問文を実行する。最小部分更新も考慮に入れて、それぞれの質問文は以下のようになる。

$$\Delta_{t1}(\min) = S^{path="ou=情報, o=理工"}_{11=R_3.lab\_id}(i_{R_2}, R_3)$$

$$\Delta_{t2}(\min) = S^{path="ou=清原, ou=情報, o=理工"}_{11=i_{R_3}.lab\_id}(R_2, i_{R_3})$$

$$\Delta_{t3}(\min) = S^{path="ou=清原, ou=情報, o=理工"}_{11=i_{R_3}.lab\_id}(i_{R_2}, i_{R_3})$$

それぞれの target Δを元のディレクトリに適応すれば差分更新が実現される。今回は i_{R2}, i_{R3} が親子関係ではないので、２番目の質問文は実行しなくてよい。修正は挿入と削除に分け、挿入、削除それぞれに更新フラグテーブルを適用する。

5. まとめと今後の課題

関係データベース側で更新されたデータに対して、最小部分更新クエリー$Q_u(\min)$から SuperSQL を通して最小のサブツリーΔ(min)を生成し、差分更新を行う方法を提案した。これにより LDAP サーバーでディレクトリを更新する時間が短縮でき、また分散してサーバーを管理している場合でも他のサーバーに影響を与えることなく更新したいサーバーだけに更新可能である。さらにいろいろな更新パターンにおいても常に最小部分更新を行う方法を示した。今後はベンチマークに基く、性能の定量的評価などが必要と考えている。

[謝辞]

本研究の一部は, 文部科学省の世界的研究教育拠点の形成のための重点的支援 21 世紀 COE プログラム「アクセス網高度化光・電子デバイス技術」の支援によるものである。

[文献]

[1] Thierry Dclot, Pascal Dcchamboux, Bcatricc Financc, Yann Lcpctit and Gillcs LcBrun: LDAP Databases and Distributed Objects: Towards a Better Integration. In Databases in Telecommunications, 2001, pp140-154.

[2] Jose A Blakclcy, Pcr-Akc Larson, Frank Wm, and Tompa: Efficiently Updating Materialized Views. in ACM SICMOD, 1986, pp61-71.

[3] Sihem Amer-Yahia, H. V. Jagadish, Laks V. S. Lakshmanan, and Divesh Srivastava: On Bounding-Schemas for LDAP Directories, in International Conference on Extending Database Technology (EDBT) 2000, pp.287-301.

[4] Sophie Cluet, Olga Kapiskaia and Divesh Srivastava: Using LDAP Directory Caches, in Symposium on Principles of Database Systems (PODS), 1998, pp273-284.

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

[6] Motomichi. Toyama.: “SuperSQL: An Extended SQL for Database Publishing and Presentation,” Proceedings of ACM SIGMOD '98 International Conference on Management of Data, 1998, pp.584-586.

[7] 古思望、遠山元道「SuperSQL によるLDAP データの自動生成」情報処理学会研究報告, Vol.2002, No67 2002-DBS-128 pp319-326, 2002.

古思 望　Nozomu KOSHI

慶應義塾大学大学院理工学研究科修士課程在学中．2002 慶應義塾大学理工学部情報工学科卒業．データベースシステムの研究に従事．情報処理学会学生会員．日本データベース学会学生会員．

遠山 元道　Motomichi TOYAMA

慶應義塾大学理工学部情報工学科専任講師．博士（工学）．1984 慶應義塾大学大学院博士課程単位取得退学．主にデータベースシステムの研究に従事．IEEE Computer Society, ACM，情報処理学会、日本ソフトウェア科学会，電子情報通信学会，日本データベース学会会員．