SuperSQLを用いたRDBとLDAPにおける統合管理の高度化

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るのがLDAP(Lightweight Directory Access Protocol)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、単一組織内での情報管理にも使用されてる。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。関係データベースとLDAPの統合利用はSuperSQLによって実現され、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を実装した。しかし、関係データベース側でデータが更新されたとき、LDAPディレクトリを更新するにはもう一度ディレクトリ全体を生成しなくてはならない。そこで本研究では関係データベース側で更新されたデータのみに関するLDAPディレクトリをSuperSQLシステムを通して生成し、差分更新をする方法を提案する。

キーワード：SuperSQL 、 関係データベース 、 LDAP 、 差分更新

The advancement of the integrated management in RDB and LDAP by using SuperSQL

Nozomu KOSHI † Motomichi TOYAMA ‡

†School of Science for OPEN and Environmental Systems,
Facul ty 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

Recently, distributed directory services have been popular with the growth of the Internet and to keep many kinds of informations in a directory structure. In particular LDAP(Lightweight Directory Access Protocol) is adopted as common, which replaces NIS or DNS. The heterogeneity of the information can be easily treated according to the flexibility of a LDAP schema, and it is used also for information management in an organization. Although there is an advantage that reference speed is fast since the tree structured nature of LDAP have redundant data, there is disadvantage that management is high cost. Therefore, 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. When updating on RDB, we have to recompute the whole directory. In this paper, we propose incremental update by LDAP directory about updated tuples from RDB throuth SuperSQL system.

keyword : SupersQL , RDB , LDAP , Incremental Update

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、単一組織内での情報管理にも使用されてる [3, 8]。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 によって実現され、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を実装した。[9]。しかし、関係データベース側でデータが更新されたとき、LDAP ディレクトリを更新するにはもう一度ディレクトリ全体を生成しなくてはならない。そこで本研究では関係データベース側で更新されたデータのみに関する LDAP ディレクトリを SuperSQL システムを通して生成し、差分更新をする方法を提案する。以下 2 章で LDAP、3 章で SuperSQL を用いて関係データベースから LDAP ディレクトリを自動生成するシステムについて説明し、4 章で差分更新の方法、5 章で結論を述べる。

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) である。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下位のノードから表現し、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, 7] 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う。$< TFE >$ 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 ([ ]!)
  同じ親を持つ兄弟のエントリー同士を続けて 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。
- path
  上位のエントリーの識別名を継承する。

3.4 実行例

学部、学科、研究室、生徒の関係データベースのスキーマを

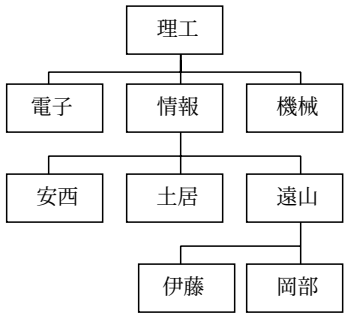
R_1:Faculty(id, name)

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}}
 @{objectcalss=organizationalUnit}%
 [l.name@{att=ou}}
 @{objectclass=organizationalUnit}%
 [{s.name@{att=cn}, e.mail@{att=mail}}
 @{objectclass=person,
    objectclass=inetorgPerson,
    objectclass=organizationalPerson}
 ]! ]! ]!] !
FROM Faculty f,Department d,
                          Lab l, Student s
WHERE f.id = d.faculty_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

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ディレクトリ全体を生成してしまう方法しかない。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 ディレクトリ側の変化分を 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 $(R_1, R_2, ..., R_k)$ 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。

$$T = S_{Join\ の条件}(R_1, R_2, ..., R_k) \qquad (1)$$

$R_j(1 \leq j \leq k)$ に挿入、削除、更新されたタプルを $i_{R_j}, d_{R_j}, u_{R_j}$ とする。R_j にタプル $t_j(i_{R_j}, d_{R_j}, u_{R_j})$ を適応し、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

$$T' = S_{Join\ の条件}(R_1, R_2, .., R'_j, .., R_k)$$

である。しかし LDAP 側の変化分 Δ_t を求めた上で、元のディレクトリ T に適応させることにより効率的になる。このとき

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

と表すことができる。これを元のディレクトリ T に適応させることにより T ’を求めることが可能である。しかし求めた Δ_t は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よいが最小の LDAP ディレクトリを更新できるようにする。それ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、上位のエントリーの識別名を継承できるようにする。そのためには SuperSQL の装飾子 “path” を利用して前述した例の SuperSQL 質問文を書き直すと以下のようになる。

$$\Delta_t(min) = S^{path=\text{”上位の識別名”}}_{Join\ の条件}(R_l, ..t_j, .., R_m)$$
$$但し、1 \leq l \leq k, 1 \leq m \leq k, l < m$$

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 では $\Delta_t(min)$ ではなく Δ_t でも更新可能であり、$\Delta_t(min)$ では関係データベースから上位の識別名を取得するなどコストもかかる。しかし最小部分更新にする理由は図 2 のように LDAP サーバーを分散して管理する場合、Δ_t では冗長なエントリーを生成してしまい全ての LDAP サーバーを更新する必要性あるが、$\Delta_t(min)$ では最小のサーバーで更新可能だからである。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。

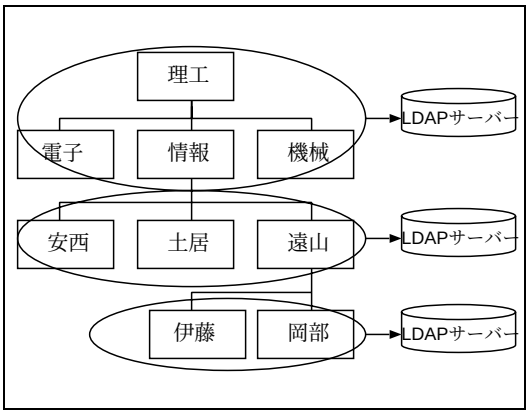


図 2: LDAP サーバーの分散管理

4.3 システム全体図

図 3 は本システムの全体図である。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

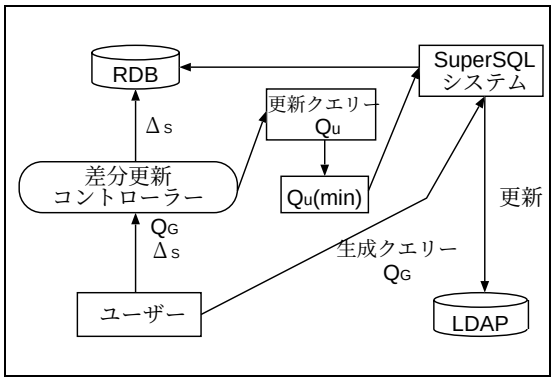


図 3: 本システムの全体図

バーに格納されているとする。このときユーザーが生成クエリー Q_G と関係データベース側の変化分 Δ_s を差分更新コントローラーに与えると、システムが関係データベースを更新し、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 システムを通して最小の LDAP ディレクトリ ($\Delta_t(\min)$) を生成し更新することができる。更新クエリー Q_u は関係データベース側で更新されたタプルに関する全てのエントリーを生成するクエリーであり、$Q_u(\min)$ よりも冗長なエントリーが生成される。

4.4 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。

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

関係データベース側に寺岡研究室のタプルが挿入された場合、

$$\Delta_s = insert\ into\ R_2\ values(11, '寺岡', 43 \times\times\times, 1)$$

はとなる。このタプルに関するエントリー全体を生成する Δ_t は

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

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 SuperSQL の装飾子"path" を用いて以下のようにする。

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

この式は寺岡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、具体的な SuperSQL 質問文は以下のようになる。また生成される Δ_t と $\Delta_t(\min)$ を図 4 に示す。

```
GENERATE ldap
  [{l.name@{att=ou}}
     @{objectclass=organizationalUnit}%
  [{s.name1@{att=cn}, e.mail@{att=mail}}
  @{objectclass=person
    ,objectclass=inetorgperson
    ,objectclass=organizationalPerson}
   ]![]!@{path = "ou=ics, o=rikou"}
FROM Lab l, Student s
Where 11 = s.lab_id;
```

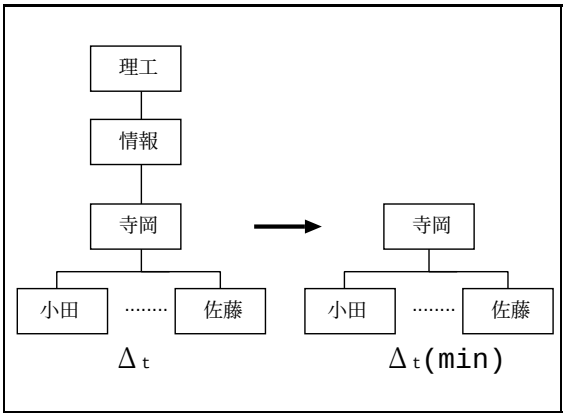


図 4: Δ_t と $\Delta_t(\min)$

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、更新される全体の 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、更新は生成した $\Delta_t(min)$ 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 差分更新パターン

これまでは、一つのタプルに関する更新しか考慮してなかった。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挿入されるなどいろいろな更新パターンがあり、さらに階層構造であるため親子関係についても関連してくる。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, faculty_id)

R_2:Lab(id, name, tel, dept_id)

R_3:Student(id, name, tel, mail, lab_id)

を例として用いる。このとき生成される LDAP ディレクトリ T は

$$T = S_{R_1.id=R_2.dept\_idand\ R_2.id=R_3.lab\_id}(R_1, R_2, R_3)$$

と表され、それぞれのテーブルに以下ののような更新操作があった場合を考える。

- 1つのテーブルに複数のタプルの挿入 (削除・更新) がある (同じ操作)。
  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生徒が複数挿入されたとすると

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

  の1つの質問文で処理できるが遠山研究室と土居研究室の生徒が挿入されると

  $$\Delta_t(min) = S^{path=\text{"}ou=\text{遠山},ou=\text{情報},o=\text{理工}\text{"}}_{8=R_3.lab\_id}(i_{R_3})$$
  $$\Delta_t(min) = S^{path=\text{"}ou=\text{土居},ou=\text{情報},o=\text{理工}\text{"}}_{10=R_3.lab\_id}(i_{R_3})$$

  の2つの質問文を実行することになる。

- 1つのテーブルの複数の更新操作 (挿入・削除・更新)
  更新操作に応じて別々に処理する必要がある。例えば遠山研究室の生徒が挿入され、別のタプルが削除されたら別々の SuperSQL 質問文で処理をする。

- 全てのテーブルで更新操作が同じである。
  $B_i(1 \leq i \leq k)$ 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。この関係を更新フラグテーブルと呼ぶことにする。3つのテーブルに挿入のされた場合の更新フラグテーブルは表 1 になる。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。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

B_1	B_2	B_3	VIEW
0	0	0	$\mathrm{S}(R_1, R_2, R_3)$
0	0	1	$\mathrm{S}(R_1, R_2, i_{R_3})$
0	1	0	$\mathrm{S}(R_1, i_{R_2}, R_3)$
0	1	1	$\mathrm{S}(R_1, i_{R_2}, i_{R_3})$
1	0	0	$\mathrm{S}(i_{R_1}, R_2, R_3)$
1	0	1	$\mathrm{S}(i_{R_1}, R_2, i_{R_3})$
1	1	0	$\mathrm{S}(i_{R_1}, i_{R_2}, R_3)$
1	1	1	$\mathrm{S}(i_{R_1}, i_{R_2}, i_{R_3})$

例えば R_2 に新しい研究室 (寺岡研究室) 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。

$$\Delta_{t1}(min) = S^{path=\text{"}ou=\text{情報},o=\text{理工}\text{"}}_{Join\ \text{の条件}}(i_{R_2}, R_3)$$
$$\Delta_{t2}(min) = S^{path=\text{"}ou=\text{寺岡},ou=\text{情報},o=\text{理工}\text{"}}_{Join\ \text{の条件}}(R_2, i_{R_3})$$
$$\Delta_{t3}(min) = S^{path=\text{"}ou=\text{情報},o=\text{理工}\text{"}}_{Join\ \text{の条件}}(i_{R_2}, i_{R_3})$$

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。また i_{R_2}, i_{R_3} が親子関係でなかったら 3 番目の質問文は実行しなくてよい。

- 挿入、削除、更新がある。
  3つの更新操作が混合していると更新フラグ

テーブルで無視できる質問文がでてくる。2つのテーブルにおける (挿入・削除)、(挿入・更新)、(削除・更新) の関連性について示す。以下のケースは (挿入・削除) についてである。
Case1 $\Delta_t \in S(i_{R_1}, i_{R_2})$ の LDAP ディレクトリの VIEW は元のディレクトリに挿入される。
Case2 $\Delta_t \in S(i_{R_1}, d_{R_2})$ の LDAP ディレクトリの VIEW は元のディレクトリに影響はない。
Case3 $\Delta_t \in S(i_{R_1}, R_2)$ の LDAP ディレクトリの VIEW は元のディレクトリに挿入される。
Case4 $\Delta_t \in S(d_{R_1}, d_{R_2})$ の LDAP ディレクトリの VIEW は元のディレクトリから削除される。
Case5 $\Delta_t \in S(d_{R_1}, R_2)$ の LDAP ディレクトリの VIEW は元のディレクトリから削除される。
Case6 $\Delta_t \in S(R_1, R_2)$ の LDAP ディレクトリの VIEW は元のディレクトリである。
以上より表 2 のようにまとめられる。

表 2: 挿入・削除

R_1	R_2	S(R_1, R_2)
挿入	挿入	挿入
挿入	削除	無視
挿入	無操作	挿入
削除	挿入	無視
削除	削除	削除
削除	無操作	削除
無操作	挿入	挿入
無操作	削除	削除
無操作	無操作	無操作

すなわち R_1 と R_2 が親子関係で関連があるとしても、削除されたタプルと挿入されたタプルとの SuperSQL 質問文の実行結果は無視できるので、実行しなくてもよい。(更新・削除) については (挿入・削除) と同じ関係であり、(挿入・更新) は (挿入・挿入) と考えてよい。

4.6 差分更新の手順

先述した５つのパターンに応じて処理していく。生成されるディレクトリは式 1 とする。それぞれのテーブルにおいて挿入・削除・更新が起こるとする。差分更新の処理手順は以下のようになる。

1. それぞれのテーブルにおいて更新操作に応じてタプルを分ける。
2. 1. で同じ親を持つタプル同士を１つにまとめる。
3. 1. と 2. で生成したそれぞれのグループに対して更新フラグテーブルより SuperSQL 質問文を実行する。実行しない SuperSQL 質問文は先述した通りである。

例えばそれぞれのテーブルでできたグループ数が g_1,g_2,.....,g_k ならば実行する SuperSQL 質問文の回数は最悪 $g_1 \times g_2 \times ..... \times g_k$ である。

4.7 最小部分更新と最小コスト更新の関連性

最小部分更新が必ずしも最小コスト更新になるとは限らない。その例としては同じテーブルかつ同じサブツリー (親が同じ) に多くの複数のタプルが挿入されたとき、１つ１つに最小部分更新の SuperSQL 質問文を生成し、更新を行うのではなく１つ上位のエントリー以下のサブツリー全体を生成した方がコストが低くなる可能性がある。例えば 1 において情報工学科に多くの生徒が挿入された場合、最小部分更新にするならばそれぞれの研究室ごとで質問文を実行することになる。

$$\Delta_t(min) = S^{path=\text{"}ou=遠山,ou=情報,o=理工\text{"}}_{8=R_4.lab\_id}(i_{R_4})$$
$$\Delta_t(min) = S^{path=\text{"}ou=安西,ou=情報,o=理工\text{"}}_{9=R_4.lab\_id}(i_{R_4})$$
$$\Delta_t(min) = S^{path=\text{"}ou=土居,ou=情報,o=理工\text{"}}_{10=R_4.lab\_id}(i_{R_4})$$

全ての研究室ごとで質問文を実行するよりは１つ上位の情報工学科をルートとするサブツリーを生成する SuperSQL 質問文を実行した方がよい場合がある。

$$\Delta_t(min) = S^{path=\text{"}ou=情報,o=理工\text{"}}_{1=R_3.dept\_id,R_4.lab\_id}(R_2, R_3, i_{R_4})$$

本研究では、コストにかかわらず最小部分更新を行えるようにする。最小部分更新は大抵、最小コスト更新になるからである。

5 まとめ

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 $\Delta_t(min)$ 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ときにおいて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える方法を示した。

参考文献

[1] Thierry Delot, Pascal Dechamboux, Beatrice Finance, Yann Lepetit and Gilles LeBrun: LDAP, Databases and Distributed Objects: Towards a Bett er Integration. in *Databases in Telecommunications*, 2001, pp140-1 54.

[2] Jose A Blakeley, Per-Ake Larson, Frank Wm, Tompa: Efficiently Updating Materialized Views. in *ACM SIGMOD* , 1986, pp61-71.

[3] Sihem Amer-Yahia, H. V. Jagadish, Laks V. S. La kshmanan, 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 Kapitskaia and Divesh Srivastava: Using LDAP Directory Caches. in *Symposium on Principles of Database Systems(POD S)*, 1998, pp273-284.

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

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

[7] 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

[8] 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.

[9] 古思 望、遠山元道「SuperSQL による LDAP データの自動生成」データベースワークショップ, pp319-328, 2002.