RDBMSによるOSIディレクトリ実現方法の検討*

3N−5

岸本 康成　窪田 光裕　空 一弘
NTT情報通信網研究所

1. はじめに

テキスト通信では，メールシステムの相互接続，通信メディアの多様化等の動向を受けて，通信対象の持つ情報を統一的に管理するディレクトリシステムの開発が進んでいる。ディレクトリ[1]の実現には，ディレクトリで管理する情報であるDIB (Directory Information Base) の構築が必要である。

現在,我々はリレーショナルデータベース管理システム (RDBMS) を用いたDIB構築方法[2]について研究を進めている。本稿ではDIBを実現するにあたって，DIT (Directory Information Tree) やエントリといったディレクトリ特有のデータ構造のテーブルへの対応付け，参照系ディレクトリ操作の中で最も処理が複雑なSEARCH操作のデータベース操作へのマッピング方法について検討結果を示す。

2. DIBのRDBMSによるテーブル設計

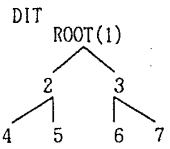
2.1 テーブル設計の基本方針

テーブルの設計にあたっては以下の方針で望んだ。

(1)実現するDITの構造を制限をしない。
(2)更新より読み出し系の操作，特に複雑な処理のため検索時間が他の操作に較べかかると予想されるSEARCH操作の検索時間を最小にすることを優先的に考慮する。

2.2 木構造のマッピング

DIT上の各エントリには，システム内部で定めた一意の識別番号であるENTRYNOを付与する。DIT上の各エントリのENTRYNOを格納するENTRYNOカラムと，その直接上位であるエントリのENTRYNOを格納するSUPERIORENTRYNOカラムを持つENTRY_Tテーブルを作成し，木構造をテーブルにマッピングした。(図1)



ENTRY_Tテーブル

ENTRYNO	SUPERIORENTRYNO
1	
2	1
3	1
4	2
5	2
6	3
7	3

図1 木構造のマッピング

2.3 属性値の格納方法

属性値の格納方法としては，以下に示す3つの方法を検討した。

案1
ENTRY_Tテーブルに全属性型の属性値を格納するカラムを作成する。(図2(a))

案2
ENTRY_Tテーブルとは別に，オブジェクトクラス毎にテーブルを作成し，1つのローには1つのエントリを対応させ，ENTRYNOカラムとオブジェクトクラスが持つ属性型の属性値を格納するカラムを作成する。(図2(b))

案3
ENTRY_Tテーブルとは別に，属性毎にテーブルを作成し，1つのローには1つの属性値を対応させ，その属性値とその属性値を持つエントリのENTRYNOを格納する。(図2(c))

3つの案のうち，案3は属性毎にテーブルを作成するので，テーブルの数は増大する。しかし，以下に示す2つ理由から属性ごとにテーブルを作成する案3を採用する。

・ディレクトリでは一般に1つの属性が複数の値を取ることを認めている。DITに制限を与えないため，1つの属性に対して任意の数の属性値を格納した際，案3は他の案に較べデータ格納効率が良い。
・SEARCH操作では属性値に対する条件に該当するエントリを検索するので，属性単位でテーブルを作成した方が検索効率が良い。

(a)案1
ENTRY_Tテーブル

ENTRYNO	SUPERIOR ENTRYNO	COUNTRY NAME	ORGANIZATIONAL UNIT NAME	COMMON NAME	SEX	·····
1						
2	1					
⋮	⋮					

存在する全属性

(b)案2
PERSONクラスのテーブル
(他のオブジェクトクラスのテーブルも同様)

ENTRYNO	COMMON NAME	SEX	·····
4			
5			
⋮			

オブジェクトクラスの持つ属性

(c)案3
COMMON NAME属性テーブル
(他の属性テーブルも同様)

ENTRYNO	COMMON NAME
4	
5	
⋮	

図2 属性値の格納方法

3. SEARCH操作の処理方式

ディレクトリ操作を実現する場合，使用頻度の高い参照系操作の中でも，特に処理の複雑なSEARCH操作の応答時間の短縮を最優先に考えなければならない。SEARCH操作では，属性値に対する条件，検索範囲条件を各々filter，subsetというの2つのパラメータで指定する。ここでは，filter及びsubsetという2つの条件のSQL文を用いた実現方法について検討した。

3.1 filterの実現方式

filterでは複数の属性値に対する条件を論理演算 (AND,OR,NOT) で結ぶ指定ができる。このとき，論理演算の実現方式案として以下の2つを検討した。

案1
全エントリに対して共通のテーブルであるENTRY_Tテーブルと各属性テーブルをJOINする方式 (JOIN方式)

案2
各属性値条件毎の検索を行うSQL文に対して集合演算子を適用する方式 (集合演算方式)

例えば，filterが2つの属性値条件のANDの場合の上記各案におけるSQL文例を以下に示す。

JOIN方式 (案1)

```
SELECT DISTINCT ENTRYNO
FROM ENTRY_T,属性テーブル1,属性テーブル2
WHERE (ENTRY_T.ENTRYNO = 属性テーブル1.ENTRYNO  AND
       属性値条件1) AND
      (ENTRY_T.ENTRYNO = 属性テーブル2.ENTRYNO AND
       属性値条件2)
```

* Implementation Methods of OSI Directory using Relational Database Management System
Yasunari KISIMOTO,Teruhiro KUBOTA,Kazuhiro SORA
NTT Network Information Systems Laboratories

集合演算方式（案2）
(SELECT DISTINCT ENTRYNO FROM 属性テーブル1
WHERE 属性値条件1)
INTERSECT
(SELECT DISTINCT ENTRYNO FROM 属性テーブル2
WHERE 属性値条件2)

各案の比較結果を表2に示す。ここでは，以下の2つのケースに対して処理時間の比較を行った。

ケース1
エントリが1つの属性に対して複数の属性値を持つ場合
（エントリ数200，1つの属性に対して10個の属性値）

ケース2
複数のエントリが同じ属性値を持つ場合
（エントリ数2000，属性値200種類）

表2より，案1は，多くのエントリが，ある属性に対して同一の属性値を持つ場合，索引使用時の実行時間が極端に遅くなる。しかし，実際のシステムではSEX属性のように多くのエントリが同一の属性値を持つ場合もある。一方，同じケースで案2の場合，処理時間は案1の場合に比べ極端に遅くなるような場合はなく，それ以外のケースでも同じ程度である。よって，論理演算の実現方式としては案2を採用する。

表2 論理演算実現方式案の比較

案		処理時間（案1のケース1,索引無しの場合を1とした時の比率）	
		索引無し	索引有り
案1 JOIN方式	ケース1	1.0	0.1
	ケース2	1.2	22.0
案2 集合演算方式	ケース1	0.9	0.1
	ケース2	0.9	0.1

3.2 subsetの実現方式

subsetの指定の中で最も実現が困難なsubset=wholeSubtreeを満たすエントリの検索方式を検討した。方式案として以下の２つを検討した。

案1
非末端エントリのROOTからの階層を表すデータ（上位エントリのENTRYNOをルートから順にTOP,LVL2,LVL3・・・・の各カラムに保持）を格納したテーブル（ENTRYSEARCH）を新たに作成し，そのテーブルにANYを用いてwholeSubtreeにある非末端エントリを求める。各属性テーブルにSUPERIORENTRYNOカラムを付加し，先に求めたエントリの下位エントリでfilter条件に合致するエントリを属性テーブルから検索する。（図3(a)）

案2
各属性テーブルにエントリのROOTからの階層を表すデータを格納するカラム（ENTRY_LEVEL）を作り，各属性テーブルごとに属性値条件を満たし,wholeSubtreeにあるエントリを検索するSQL文を作成する。（図3(b)）

各案で検索に用いるSQL文は，例えばBaseObjectのENTRYNOが3の場合，以下のようになる。

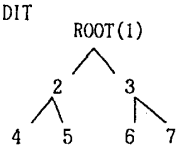
（案1）
```
SELECT DISTINCT ENTRYNO FROM 属性テーブル
WHERE 属性テーブルのSUPERIORENTRYNO = ANY
(SELECT ENTRYNO FROM ENTRYSEARCH WHERE LVL2 = 3)
```

（案2）
```
SELECT DISTINCT ENTRYNO FROM 属性テーブル
WHERE 属性値カラム=属性値  AND (ENTRY_LEVEL LIKE '1,3'%)
```

(a)案1
ENTRYSEARCHテーブル

ENTRYNO	TOP	LVL2	LVL3	・・
1	1			
2	1	2		
3	1	3		

DIT



属性テーブル

ENTRYNO	SUPERIORENTRYNO	属性値カラム	・・・・
4	2		
5	2		
6	3		
7	3		

(b)案2
属性テーブル

ENTRYNO	ENTRY_LEVEL	属性値カラム
4	1,2,4	
5	1,2,5	
6	1,3,6	
7	1,3,7	

ENTRY_LEVELカラムに格納される文字列
＝ENTRYNO(1)+","+ENTRYNO(2)+","+ENTRYNO(3)+・・・・+ENTRYNO(N)
（ENTRYNO(k)は上位エントリのROOTからk番目に対応するエントリのENTRYNOを示す文字列）

図3 検索範囲条件実現方式各案のテーブル設計

2案の比較を表3に示す。案2は，属性テーブルごとにエントリのROOTからの階層を表すデータを格納しなければならず,DISK資源を大量に使用する。しかし，大規模DITを想定しても,ディスク使用量は案1の1.6倍にすぎない。一方，数十万ものエントリを持つ実用化システムを考えた場合，案1では処理時間が検索範囲内のエントリ数の増加に伴って増大するため，実用的な処理時間が得られない可能性が高い。従って，各SQL文でANYを使用せず1つの属性テーブルに閉じた，より高速な検索が期待できる案2を採用する。

表3 検索範囲にあるエントリの検索方式案の比較

項目 ＼ 案	案1	案2
検索に必要なテーブル	属性テーブル ENTRYSEARCH	属性テーブル
処理時間の検索範囲内エントリ数依存性	大（表4）	小
ディスク使用量(案1を1とした時の比率)	1.0	1.6

表4 案1で用いるSQL文の評価

検索範囲内エントリ数	検索時間（検索範囲内エントリ数200,索引有りの場合を1とした時の比率）	
	索引有り	索引無し
200	1.0	1.8
400	2.2	3.0

4. おわりに

今回，RDBMSによるDIBの実現方法について検討した。テーブル設計では，木構造のマッピング方法，属性値の格納方法について検討した。また，ディレクトリ操作の中で応答時間に対する要求条件の実現が困難なSEARCH操作において，filter及びsubsetを満たすエントリを検索するSQL文の構築方法を検討した。今後は，今回の検討の結果採用したSEARCH処理方式を実装し，応答時間，処理能力についての実測による評価を行う。

参考文献

1)CCITT勧告 X.500シリーズ／ISO 9594
2)小花貞夫，西山智，鈴木健二：リレーショナルアプローチによるOSIディレクトリのDIB（ディレクトリ情報ベース）の実装と評価，情報処理学会論文誌，Vol.32 No.11 pp.1488-1497 (1991)