4R-6 グラフデータを扱うデータベース・システム（グラフデータ操作言語　ＧＯＬ－Ｉ）

関　義長　　遠山　元道　　浦　昭二
慶應義塾大学

１．はじめに

コンピュータと通信の発達にともない、多様かつ複雑なデータを扱えるデータベース・システムが必要になってきた。グラフデータは、そのようなデータの一種である。例えば、道路網や電気回路、化学反応の遷移図、意味ネットワーク、その他ネットワークで表現できる様々なものもそうである。

しかし、関係データモデルでは、グラフデータ（頂点と辺の集合として表現される）を扱うのに十分な機能を持たない。そこで現状では、アプリケーション毎に処理することが多い。また関係モデルに推移閉包を求める演算を含める研究等も行われている。

本研究では、グラフデータのデータ抽象化をはかり、グラフデータを直接表現し操作できるデータベース・システムの設計・試作を行う。

以下では、グラフデータの利点、欠点をふまえた上で、どのようなデータベース・システムを設計、試作していくのか、ということについて述べる。

２．グラフデータ操作言語（ＧＯＬ－Ｉ）

グラフデータを表現し操作するための言語である。Graph-data Operation Languageの頭文字を取ってＧＯＬ－Ｉと名付ける。以下には、ＧＯＬ－Ｉの構文を記す。

２－１．スキーマ定義

グラフデータのスキーマ定義には、a)頂点と辺の属性の定義、b)頂点の型の定義、c)辺の型の定義、d)グラフの型の定義、が必要である。それらの構文は、以下の通りである。

```
schm_exp  ::= DEFINE_FIELD(att_name domain) |
          DEFINE_NODE_TYPE(ntype_name(att_name_list)) |
          DEFINE_EDGE_TYPE(etype_name((att_name_list)
                                (ntype-ntype))) |
          DEFINE_GRAPH_TYPE(gtype_name((ntype_list)
                                      (etype_list))
```

２－２．グラフ演算

グラフの検索と生成を行うのが、グラフ演算である。

a）グラフ検索演算

この演算は、値としてグラフを返すものとパスの集合をかえすものとがある。

```
α（g）   －＞  g'          g   ：グラフ
β（G）   －＞  g           G   ：グラフの集合
ρ（g）   －＞  P           P   ：パスの集合
graph_search_op  ::=  α_op  |  β_op  |  ρ_op
α_op  ::=  MAKE_SUBGRAPH((graph)(new_graph(n_e))
                    (var_list)(condition_clause)) |
           MAKE_CLOSURE((graph)(tc_name node(n-e))
                    (var_list)(condition_clause))
β_op  ::=  UNION_GRAPH(graph_list)  |
           INTERSECT_GRAPH(graph_list)
ρ_op  ::=  MAKE_SIMPL_PATH((graph)(path_name
         node(n-e)node)(var_list)(condition_clause))
n-e    ::=  ＊ | etype | ntype | etype  n-e | ntype  n-e
var_list  ::=  φ |  var_type  |  var_type  var_list
var_type  ::=  n_var/n_type  |  e_var/e_type
condition_clause  ::=  φ | var.attri  comp  var.attri |
        var.attri  comp  constant |
    ALLva(condition_clause) | EXSva(condition_clause) |
        (condition_clause  AND  condtion_clause)  |
        (condition_clause  OR  condtion_clause)
```

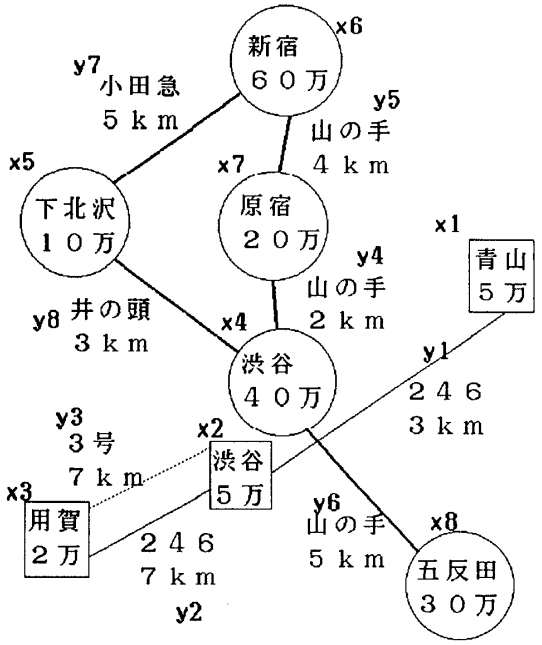
b）グラフ生グラフ生成演算は、頂点の集合と辺の集合から１つのグラフを生成するための演算である。

```
η（Ｎ＊Ｅ）  －＞  g          N  ：頂点の集合
                              E  ：辺の集合
graph_create_op  ::=  CREATE_GRAPH(gtype_name  gname
                        (CREATE_NODE(node_list))
                        (CREATE_EDGE(edge_list)))
node_list  ::=  (ntype_name(val_list))
edge_list  ::=  (etype_name((val_list)(node-node)))
```

３．ＧＯＬ－Ｉの使用例

道路網と鉄道網をグラフ化したデータを作成、検索するのに、ＧＯＬ－Ｉをどのよう用いていくかを例示する。

グラフデータの例
地図・東京



Ｄａｔａｂａｓｅ　Ｓｙｓｔｅｍ　Ｄｅａｌｉｎｇ　Ｗｉｔｈ　Ｇｒａｐｈ－ｄａｔａ
（Ｇｒａｐｈ－ｄａｔａ　Ｏｐｅｒａｔｉｏｎ　Ｌａｎｇｕａｇｅ　ＧＯＬ－Ｉ）
Ｙｏｓｈｉｎａｇａ　ＳＥＫＩ，　Ｍｏｔｏｍｉｃｈｉ　ＴＯＹＡＭＡ，　Ｓｈｏｊｉ　ＵＲＡ
ＫＥＩＯ　Ｕｎｉｖｅｒｓｉｔｙ　３－１．グラフ

データのスキーマ定義

DEFINE文を用いてスキーマの定義を行う。属性、頂点の型、辺の型、グラフの型の順に定義を行う。

前頁のグラフは、道路網と鉄道網に関する地図という型のグラフである。頂点は、交差点と駅であり、辺は、道路と首都高と鉄道である。たとえば交差点の属性は交差点名と通過台数であり、通過台数のドメインは6桁の整数である。定義は以下の通りである。

```
DEFINE_FIELD(交差点名  char10)
DEFINE_FIELD(通過台数  int6)
DEFINE_FIELD(駅名  char10)
DEFINE_FIELD(道路名  char10)
DEFINE_FIELD(距離  int3)
DEFINE_FIELD(利用者数  int7)
DEFINE_FIELD(路線名  char10)
DEFINE_NODE_TYPE(交差点(交差点名  通過台数))
DEFINE_NODE_TYPE(駅(駅名  利用者数))
DEFINE_EDGE_TYPE(道路((道路名  距離)(交差点－交差点)))
DEFINE_EDGE_TYPE(首都高((道路名 距離)(交差点-交差点)))
DEFINE_EDGE_TYPE(鉄道((路線名  距離)(駅－駅)))
DEFINE_GRAPH_TYPE(地図((交差点 駅)(道路 首都高 鉄道)))
```

3－2．グラフデータの生成

CREATE_GRAPH演算を用いて、頂点と辺にデータを挿入し、グラフを生成する。

ｘｉとｙｉは、インスタンス記号といい、頂点と辺を一意に識別するためのキーとなっている。これは、データ挿入の時に必ず付けなければならない。

```
CREATE_GRAPH(地図、東京)
      (CREATE_NODE
            (交差点 (青山  5万  x1)
                    (渋谷  5万  x2)
                    (用賀  2万  x3))
            (駅     (渋谷  40万  x4)
                    (下北沢  10万  x5)
                    (新宿  60万  x6)
                    (原宿  20万  x7)
                    (五反田  30万  x8)))
      (CREATE_EDGE
            (道路   (246  3ｋｍ  (x1-x2)  y1)
                    (246  7ｋｍ  (x2-x3)  y2))
            (首都高 (3号  7ｋｍ  (x2-x3)  y3))
            (鉄道   (山の手  2ｋｍ  (x4-x7)
                    (山の手  4ｋｍ  (x6-x7)  y5)
                    (山の手  5ｋｍ  (x4-x8)  y6)
                    (小田急  5ｋｍ  (x6-x5)  y7)
                  (井の頭  3ｋｍ  (x4-x5)  y8))))
```

以上の演算により前頁のようなグラフデータが生成される。

3－3．グラフデータの検索

a）α_opのMAKE_SUBGRAPHを使ってグラフ地図・東京から路線名が”山の手”である辺を持つサブグラフを求める演算を行う。

```
MAKE_SUBGRAPH((東京)(山の手線(ｎ1  ｅ1))
              (ｎ1／駅  ｅ1／鉄道)
              (ｅ1．路線名 ＝ ”山の手” AND
              ｅ1．ｎｏｄｅ ＝ ｎ1))
```

この演算の意味は、以下のとおりである。

```
g’ ＝  {ｎ1  ｅ1 |
        ｎ1∈N of 東京  ∧
        ｅ1∈E of 東京  ∧
        ｅ1．路線名 ＝ ”山の手” ∧
        ｅ1．node ＝ ｎ1}
```

結果として　g’ ＝ {山の手線 N’ E’}

N’ ＝ {x4 x6 x7 x8}

E’ ＝ {y4 y5 y6}　が求められる。

b）ρ_opのMAKE_SIMPL_PATHを使って新宿と渋谷の間の単純経路となる鉄道の経路を求める。

```
MAKE_SIMPL_PATH((東京)
              (新宿－渋谷  ｎ1((鉄道  駅)＊)ｎｍ)
              (ｎ1/駅  ｎｍ/駅)
              (ｎ1．駅名 ＝ ’新宿’  AND
              ｎｍ．駅名 ＝ ’渋谷’))
```

演算の意味は以下のとおりである。

```
P ＝  {p |
    p＝ｎ1ｅ1・ｎｉｅｉ・ｎｊｅｊ・ｅｍ－1ｎｍ∧
      ｎｉ≠ｎｊ  ∧
      ｎ1, ｎｍ, ｎｉ, ｎｊ ∈ 駅 ｏｆ 東京 ∧
      ｅｉ, ｅｊ ∈ 鉄道 ｏｆ 東京 ∧
      ｎ1．駅名 ＝’新宿’∧ｎｍ．駅名 ＝’渋谷’}
```

結果として

P ＝ {x4y8x5y7x6 , x4y4x7y5x6}　が求められる。

4．検討と考察

以上、ＧＯＬ－Ｉの構文の基本的な部分についてのみ記した。その他の構文のうちで重要と思われるものについて簡単に述べる。

a）データの削除に関する構文(DELETE)

頂点の削除、辺の削除、グラフの削除がある。ＧＯＬ－Ｉの操作で注意が必要なのは、頂点の削除である。グラフにおいて、辺は頂点と頂点の直積の部分集合として現れる。よって頂点を削除すると、それに接続している辺も削除される。

b）グラフの階層化に関する構文(MAKE_HYPER_NODE)

グラフデータが大きくなると、そのままでは扱いにくくなる。そこでグラフを階層化するために、ハイパー・ノードというものを考える。ハイパー・ノードは、いくつかの頂点とそれらの間の辺を集めたものである。全体のグラフとハイパー・ノードとの間を接続している辺のインスタンスも持っている。

c）集約関数

パスの検索に関して、パスを求めるだけではなく、最短経路を求めたり、パス中の頂点や辺の属性値を取り出すために、いくつかの集約演算が用意してある。

システムのインプリメントは、現在行っている途中である。現在のＧＯＬ－Ｉは、アプリケーション・プログラムインターフェイスが中心である。今後は、ユーザ・インターフェイスのサポートにより、使い易いものにしたい。例えばグラフィック・インターフェイスがサポートできれば良いと思う。

5．結論

関係モデルでは、表現しにくいか、もしくは、表現できないデータや演算を、グラフデータと操作言語の導入により表現できるようにした。

参考文献

Cruz87.”A Graphical Query Language Supporting Recursion”Isabel F.Cruz Alberto O.Mendelzon,Peter T.Wood, Proc.ACM SIGMOD Conf. 87.

Banc88.”Object-Oriented Database Systems” Francois Bancilhon, POAS88, pp152-162

伊理87.”計算幾何学と地理情報処理” 伊理　正夫, 腰塚　武志, 「ｂｉｔ」 臨時増刊号